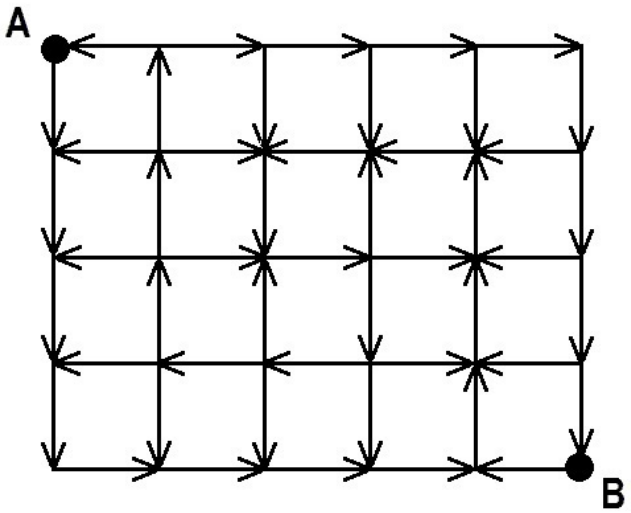
Первый отборочный этап

Задачи первого этапа. Информатика

Первая попытка. Задачи 8–11 класса

Задача I.1.1.1. Город перекрестков (20 баллов)



Вы разрабатываете навигатор для одного города. Этот город разбит улицами на квадратные кварталы, причем движение по любому из отрезков улицы в пределах каждого квартала строго одностороннее. С каждого перекрестка можно выехать только в разрешенных знаками направлениях. Требуется по прилагаемой карте города с указанными на ней разрешенными направлениями перемещения проложить самый короткий маршрут из точки A в точку B.

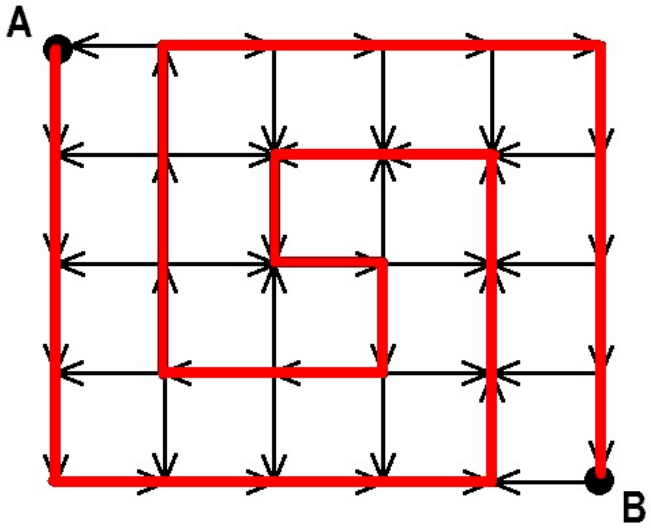
Формат входных данных

На вход подается карта перекрестков города. В первой строке содержатся два числа N — число кварталов с севера на юг и M — число кварталов с запада на восток ($1 \leqslant n \leqslant 50$). Точка A самая северо-западная, точка B самая юго-восточная. Далее в $2 \cdot N + 1$ строках содержится описание разрешенных направлений движения. Улицы города запад-восток описаны в нечетных строках. В каждой такой строке содержится по M символов без пробела, указывающих разрешенное движение на соответствующем участке. В четных строках содержится описание улиц север-юг. В этих строках содержится по $M + 1$ символов, указывающих возможное движение по отрезкам улиц север-юг. Движение на север, юг, запад, восток обозначается буквами n, s, w, e соответственно.

Формат выходных данных

В первую строку вывести число отрезков улиц в самом коротком маршруте из точки A в точку B. Во вторую строку нужно выдать описание этого маршрута в виде последовательности символов n, s, w, e без пробелов. Если кратчайших маршрутов несколько, выдать самый первый среди них по алфавитному порядку. Гарантируется, что из точки A можно попасть в точку B.

Пояснения к примеру



Примеры

Пример №1

Стандартный ввод
4 5 weeee snssss wewww snsnns weeew snnsns wwwew ssssns eeeew

Стандартный вывод
29 sssseeeennnwwseswwnnneeeessss

Пример программы-решения

Ниже представлено решение на языке C++.

```cpp
//Алгоритм Дейкстры

#include<bits/stdc++.h>

#define pb push_back
#define mp make_pair
#define f first
#define s second

#define all(x) (x).begin(), (x).end()
#define sz(a) (int)(a).size()
#define for0(i, n) for (int i = 0; i < (int)(n); i++)
#define for1(i, n) for (int i = 1; i <= (int)(n); i++)
#define int long long

using namespace std;
typedef pair<int, int> pii;
typedef long double ld;

int dx[4] = {-1, 0, 1, 0},
    dy[4] = {0, 1, 0, -1};
char z[4] = {'n', 'e', 's', 'w'};

signed main(){

   ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

   int n, m;
   cin >> n >> m;
   vector<vector<vector<int> > > G(n+1, vector<vector<int> >(m+1));

   string s;
   for0(i, 2*n+1){
    cin >> s;
    if(i%2){
            int tx = i/2;
        for0(j, sz(s))
            if(s[j] == 's')
                G[tx][j].pb(2);
            else
                G[tx+1][j].pb(0);

    }
    else{
        int tx = i/2;
        for0(j, sz(s))
          if(s[j] == 'w')
            G[tx][j+1].pb(3);
          else
            G[tx][j].pb(1);
    }
   }

    string inf;
    inf.resize(6000, '#');
    vector<vector<string> > d(n+1, vector<string>(m+1, inf));
    vector<vector<int> > mark(n+1, vector<int>(m+1, 0));
    d[0][0] = "";

    for0(u, (n+1)*(m+1)){
```

```
61      int tx, ty, mn = 1e9;
62         for0(i, n+1) for0(j, m+1) if(mark[i][j] == 0 && sz(d[i][j]) < mn){
63             tx = i;
64             ty = j;
65             mn = sz(d[i][j]);
66         }
67
68         mark[tx][ty] = 1;
69
70         for0(i, sz(G[tx][ty])){
71             int tdir = G[tx][ty][i];
72             int nx = tx + dx[tdir];
73             int ny = ty + dy[tdir];
74             string ts = d[tx][ty] + z[tdir];
75
76             if(sz(ts) < sz(d[nx][ny]) || (sz(ts) == sz(d[nx][ny]) && ts < d[nx][ny]))
77                 d[nx][ny] = ts;
78         }
79     }
80
81     cout<<sz(d[n][m])<<endl;
82     cout<<d[n][m];
83 }
```

Пример программы-решения

Ниже представлено решение на языке C++.

```
1  //Обход в ширину
2
3  #include<bits/stdc++.h>
4
5  #define pb push_back
6  #define mp make_pair
7  #define x first
8  #define y second
9
10 #define all(x) (x).begin(), (x).end()
11 #define sz(a) (int)(a).size()
12 #define for0(i, n) for (int i = 0; i < (int)(n); i++)
13 #define for1(i, n) for (int i = 1; i <= (int)(n); i++)
14 #define int long long
15
16 using namespace std;
17 typedef pair<int, int> pii;
18 typedef long double ld;
19
20    int n, m;
21 int dx[4] = {0, -1, 1, 0},
22     dy[4] = {1, 0, 0, -1};
23 char z[4] = {'e', 'n', 's', 'w'};
24
25 bool in_sq(int x, int y){
26
27     return (x >= 0 && y >= 0 && x <= 2*n && y <= 2*m);
28 }
29
30 signed main(){
31     ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);
```

```
32
33      string ans[200][200];
34      cin >> n >> m;
35      vector<string> G;
36      string s;
37      for0(i, 2*n+1){
38          cin >> s;
39          string z;
40          if(i%2 == 0){
41              for(auto q : s) z = z + "0" + q;
42                 z += "0";
43          }
44          if(i%2 == 1){
45              for(auto q : s) z = z + q + " ";
46          }
47          G.pb(z);
48      }
49      deque<pii> och;
50      och.pb({0, 0});
51      G[0][0] = '#';
52      ans[0][0] = "";
53
54      while(sz(och) > 0){
55          pii tp = och[0];
56          och.pop_front();
57
58          for0(i, 4){
59              int nx = tp.x + dx[i];
60              int ny = tp.y + dy[i];
61              if(in_sq(nx, ny) && G[nx][ny] == z[i] &&  G[nx + dx[i]][ny + dy[i]] == '0'
                ↪  ){
62                  G[nx + dx[i]][ny + dy[i]] = '#';
63                  ans[nx + dx[i]][ny + dy[i]] = ans[tp.x][tp.y] + z[i];
64                  och.pb({nx + dx[i], ny + dy[i]});
65              }
66          }
67      }
68
69      cout<<sz(ans[2*n][2*m])<<endl;
70      cout<<ans[2*n][2*m];
71  }
```

Задача I.1.1.2. WALL-E и кубики (20 баллов)

Робот WALL-E пытается упорядочить груду кубиков. В зоне его ответственности есть n расположенных друг за другом столбиков, каждый состоит из кубиков, стоящих друг на друге (в i-м столбике содержится h_i кубиков, столбики нумеруются слева направо). Робот действует по следующему алгоритму: он ищет самый левый столбик с номером i, в котором кубиков больше, чем в предыдущем, то есть $h_{i-1} < h_i$. Далее он берет верхний кубик столбика номер i и сбрасывает его на столбик номер $i-1$. Если существует столбик с номером $i-2$ и в нем теперь меньше чем в $i-1$-м столбике, то данный кубик перемещается далее на $i-2$-й столбик. И так до тех пор, пока этот кубик, либо не дойдет до верха первого столбика либо не упрется в более высокий столбик слева. Далее эта последовательность операций повторяется до тех пор, пока есть два рядом стоящих столбика таких, что $h_{i-1} < h_i$. По значениям высот исходных n столбиков вывести высоты упорядоченных n столбиков.

Формат входных данных

В первой строке содержатся число N — количество столбиков. В следующей строке находятся N чисел $h_1, h_2, \ldots, h_N$ через пробел, соответствующие высотам соответствующих столбиков. $1 \leqslant N \leqslant 10^5$, $1 \leqslant h_i \leqslant 2 \cdot 10^9$.

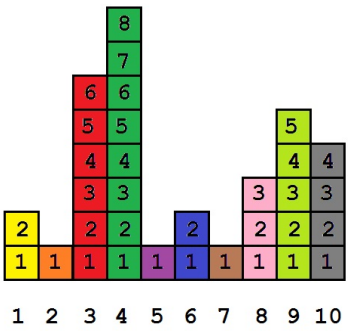
Формат выходных данных

Вывести N чисел через пробел, соответствующих высотам столбиков, после того как WALL-E закончит работу.

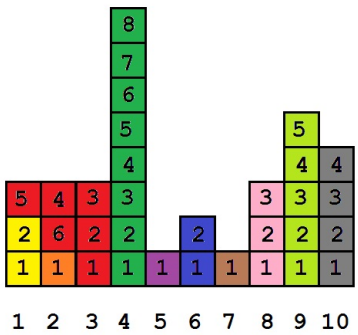
Пояснения к примеру

Следующая серия иллюстраций показывает, куда и какие кубики попали в процессе упорядочивания столбиков:

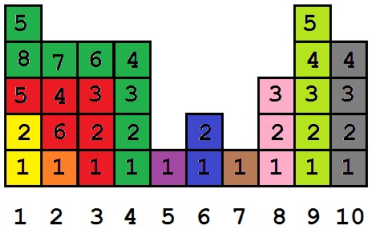
1. Стартовая ситуация



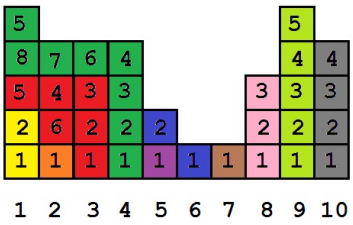
2. Упорядочили столбик 3



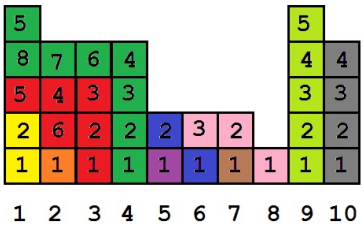
3. Упорядочили столбик 4



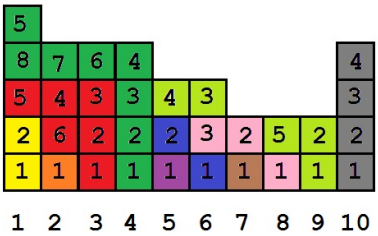
4. Упорядочили столбик 6



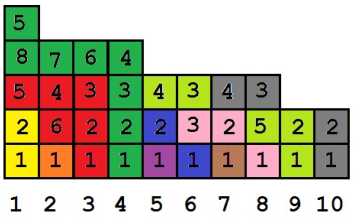
5. Упорядочили столбик 8



6. Упорядочили столбик 9



7. Упорядочили столбик 10, итоговое расположение кубиков



Примеры

Пример №1

Стандартный ввод
10 2 1 6 8 1 2 1 3 5 4
Стандартный вывод
5 4 4 4 3 3 3 3 2 2

Пример программы-решения

Ниже представлено решение на языке C++.

```cpp
//Стек пар
#include<bits/stdc++.h>

#define pb push_back
#define mp make_pair
#define x first
#define y second

#define all(x) (x).begin(), (x).end()
#define sz(a) (int)(a).size()
#define for0(i, n) for (int i = 0; i < (int)(n); i++)
#define for1(i, n) for (int i = 1; i <= (int)(n); i++)
#define int long long

using namespace std;
typedef pair<int, int> pii;
typedef long double ld;

signed main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    int n;
    cin >> n;
    vector<int> h(n);
    for0(i, n) cin >> h[i];

    vector<pii> st;
    st.pb({2e9, 1});
```

```
29
30      for0(i, n){
31
32          if(h[i] < st.back().x) st.pb({h[i], 1});
33          else
34          if(h[i] == st.back().x) st.back().y++;
35          else
36          if(h[i] > st.back().x){
37              int thb = st.back().x;
38              int ost = h[i] - thb;
39              int d = 1;
40              int v = 0;
41              int sump = 0;
42
43              bool ok = 0;
44              while(!ok){
45                  int sp = d * v - sump;
46
47                  if(ost <= sp){
48                          ok = 1;
49                          int tr = ost + sump;
50                          int c = tr / d;
51                          int o = tr % d;
52
53                          if(o > 0){
54                              int thh = thb + c + 1;
55
56                              if(thh ==st.back().x) st.back().y += o;
57                              else st.pb({thh, o});
58                          }
59
60                          int thh = thb + c;
61
62                          if(thh == st.back().x) st.back().y += (d - o);
63                          else st.pb({thh, d - o});
64                  }
65                  else{
66                      d += st.back().y;
67                      v = st[sz(st)-2].x - thb;
68                      sump += (st.back().x - thb) * st.back().y;
69                      st.pop_back();
70                  }
71              }
72          }
73      }
74
75      for(int i = 1; i < sz(st); i++)
76          for0(j, st[i].y) cout<<st[i].x<<' ';
77  }
```

Задача I.1.1.3. Послание внеземного разума (10 баллов)

Сенсация! Известный специалист по UFO профессор Персиков получил послание от внеземного разума. Во всяком случае он так считает. Послание представляет собой непрерывную последовательность из сигналов нескольких типов, каждый из этих типов для определенности можно обозначить малой буквой латинского алфавита. Профессор считает, что доказательством искусственности происхождения сигнала служит его периодичность. При этом период должен быть строго равен «константе

Персикова» — числу P. Такую последовательность назовем P — периодичной.

К сожалению, некоторые сигналы из-за прохождения сквозь бездны пространства (а может даже и времени) были утеряны. Их заменили на знак вопроса. Теперь профессору для дальнейшей работы требуется выяснить, можно ли как-то заменить утерянные сигналы на буквы так, чтобы последовательность стала P — периодичной. Для чистоты эксперимента полученная из космоса последовательность была размещена среди себе подобных. Не подведите, в ваших руках — будущее межгалактических взаимоотношений!

Формат входных данных

В первой строке содержится два числа через пробел: N — общее количество строк, которые вам нужно проверить на P-периодичность и число P — константа Персикова ($1 \leqslant P \leqslant 5 \cdot 10^3$). Далее содержится N непустых строк, состоящих из малых букв латиницы и знаков вопроса. $1 \leqslant N \leqslant 5 \cdot 10^4$, суммарная длина всех строк не превосходит $2 \cdot 10^5$.

Формат выходных данных

Вывести N строк, в каждой из которых вывести либо «YES», если соответствующую строку можно сделать P-периодичной путем замены всех знаков вопроса на некоторые буквы, и «NO» в противном случае. Будем считать последовательность P-периодичной, если для любых двух символов, расстояние между которыми кратно P верно, что они совпадают.

Примеры

Пример №1

Стандартный ввод
8 4 abacabaca abracadabra aa?aaa?aaaaaaaaaa q ??????? q?er?w? q?erw?? q?er?wer?werw?er

Стандартный вывод
YES NO YES YES YES YES NO NO

Пример программы-решения

Ниже представлено решение на языке C++.

```cpp
//Метод карманов
#include<bits/stdc++.h>
#define pb push_back
#define mp make_pair
#define x first
#define y second

#define all(x) (x).begin(), (x).end()
#define sz(a) (int)(a).size()
#define for0(i, n) for (int i = 0; i < (int)(n); i++)
#define for1(i, n) for (int i = 1; i <= (int)(n); i++)
#define int long long

using namespace std;
typedef pair<int, int> pii;
typedef long double ld;

signed main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    int u, p;
    cin >> u >> p;
        while(u--){
            string s;
            cin >> s;
            vector<set<char> > v(p);
            for0(i, sz(s))
              if(s[i] != '?') v[i%p].insert(s[i]);

              bool ok = 1;
            for(auto q : v)
                if(sz(q) > 1)
                    ok = 0;

                 cout<<  ((ok)? "YES\n" : "NO\n");
        }
}
```

Пример программы-решения

Ниже представлено решение на языке C++.

```cpp
//Эффективный перебор
#include<bits/stdc++.h>

#define pb push_back
#define mp make_pair
#define x first
#define y second

#define all(x) (x).begin(), (x).end()
#define sz(a) (int)(a).size()
#define for0(i, n) for (int i = 0; i < (int)(n); i++)
#define for1(i, n) for (int i = 1; i <= (int)(n); i++)
```

```
#define int long long

using namespace std;
typedef pair<int, int> pii;
typedef long double ld;

signed main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    int u, p;
    cin >> u >> p;
        while(u--){
            string s;
            cin >> s;
                bool ok = 1;

                for0(i, p){
                    char c = '#';
                for(int j = 0; i+j < sz(s); j += p) {
                    if(s[i+j] != '?')
                        if(c == '#') c = s[i+j];
                    else
                        if(c != s[i+j]) ok = 0;
                }
                }

                 cout<<  ((ok)? "YES\n" : "NO\n");
        }

}
```

Пример программы-решения

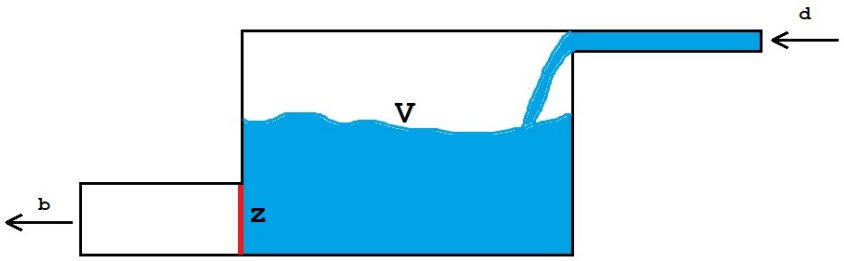
Ниже представлено решение на языке Python3.

```
n, p = map(int, input().split())
lines = []
f = open('text.txt', 'w')
for i in range(n):
    lines.append(input())
for i in range(n):
    line = lines[i]
    q = True
    for j in range(p):
        c = ''
        k = j
        while k < len(line):
            if line[k] != '?':
                if c == '':
                    c = line[k]
                else:
                    if line[k] != c:
                        q = False
                        break
            k += p
        if not q:
            break


```

```
25      if q:
26          f.write("YES" + '\n')
27      else:
28          f.write("NO" + '\n')
29  f.close()
```

Задача I.1.1.4. Водопровод с резервной емкостью (20 баллов)



Рассмотрим следующую модель водопровода. Имеется резервная емкость объемом V литров. По входной трубе постоянно поступает вода со скоростью d литров в минуту. По выходной трубе жидкость выливается со скоростью b литров в минуту. Выполняется условие, что $d < b$. В начальный момент времени выход Z из емкости перекрыт. В тот момент, когда емкость наполняется, выход открывается и вода поступает на выпуск. Так как $d < b$, то в какой-то момент емкость полностью опустеет, и заслонка Z снова закрывается, после чего процесс повторяется снова.

Очевидно, что при этом в работе водопровода случаются паузы, в течение которых происходит заполнение резервной емкости. К ним относится и первая пауза, служащая для первичного наполнения емкости водой. По техническим причинам требуется, чтобы длина каждой такой паузы не превышала величины p. При этом нас интересует суммарное время t, в течение которого выходная труба была открыта. Количество пауз n при этом должно быть минимальным возможным.

В рассматриваемой модели используются сколь угодно малые доли времени и объема. Считается, что заслонка Z срабатывает мгновенно.

По заданным величинам d, b, t, p требуется найти такой целый объем V, при котором число пауз n было минимально возможным для заданного времени подачи воды t, а среди всех объемов, обеспечивающих такое время работы подачи воды и такое количество пауз найти самый маленький.

Формат входных данных

В первой строке содержится четыре целых числа d, b, t, p через пробел: $1 \leqslant d < b \leqslant 2 \cdot 10^9$, $1 \leqslant t \leqslant 15000$, $1 \leqslant p \leqslant 5000$.

Формат выходных данных

Вывести одно целое число — объем V резервной емкости, обеспечивающей необходимые ограничения на подачу воды. Если таких объемов несколько, вывести минимальный.

Пояснения к примеру

Рассмотрим первый пример из условия. Скорость подачи d равна 5 литров в минуту, скорость выпуска b равна 10 литров в минуту, требуется обеспечить суммарную работу выпуска в течение $t = 32$ минут, при этом максимальный размер любой паузы не должен превышать $p = 8$ минут.

Если мы установим объем резервной емкости 40 литров, то:

- ровно за 8 минут (что разрешено условием) она наполнится со скоростью 5 литров в минуту;
- далее пойдет процесс выпуска: 40 литров будут выпущены за 4 минуты со скоростью выпуска 10 литров в минуту, но за это время в емкость попадет $4 \cdot 5 = 20$ литров новой воды. Она будет выпущена за 2 минуты, но за это время поступит $2 \cdot 5 = 10$ литров новой воды, которая будет выпущена за 1 минуту и так далее. Так как доли времени и объема могут быть сколь угодно малыми, получим при подсчете времени работы выпуска следующую сумму: $4 + 2 + 1 + 0,5 + 0,25 + 0,125 + ...$, которая в итоге равна 8. То есть через 8 минут емкость полностью опустеет, и заслонка закроется. Для того, чтобы время работы выпуска было равно 32 минуты, потребуется $32/8 = 4$ таких цикла, а значит и 4 паузы. За меньшее число пауз работу выпуска в течение 32 минут при существующих ограничениях обеспечить не получится.
  Очевидно, что $V = 40$ литров — максимально возможный разрешенный объем, иначе первая же пауза будет больше допустимой. С другой стороны, если мы попробуем уменьшить объем до 39 литров, то получим:
- первоначальное заполнение будет произведено за $39/5 = 7.8$ минуты. Далее процесс выпуска будет работать $39/10 + 39/20 + 39/40 + ... = 7,8$ минуты. Тогда за 4 цикла выпуск будет работать 31.2 минуты и для обеспечения 32 минут работы выпуска потребуется пятая пауза.

Примеры

Пример №1

Стандартный ввод
`5 10 32 8`

Стандартный вывод
`40`

Пример программы-решения

Ниже представлено решение на языке C++.

```
1  //Бинарный поиск
2  #include<bits/stdc++.h>
3
4  #define pb push_back
5  #define mp make_pair
6  #define x first
7  #define y second
8
9  #define all(x) (x).begin(), (x).end()
10 #define sz(a) (int)(a).size()
```

```
#define for0(i, n) for (int i = 0; i < (int)(n); i++)
#define for1(i, n) for (int i = 1; i <= (int)(n); i++)
#define int long long

using namespace std;
typedef pair<int, int> pii;
typedef long double ld;


signed main(){
  ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

  int d, b, t, p;

  cin >> d >> b >> t >> p;

  int y = t * (b-d);
  int z = d * p;

  int a = y / z + (y % z != 0);

  int L = 0, R = d * p;

  while(R - L > 1){
    int M = (R + L) / 2;

    if( M * a >= t * (b-d)) R = M; else L = M;

  }

  cout << R;
}
```

Задача I.1.1.5. Разбиения на различные множители (30 баллов)

Эта задача формулируется предельно кратко.

Дано натуральное число N. Требуется найти число способов представить его в виде произведения попарно различных множителей больших 1.

Формат входных данных

В первой строке содержится одно натуральное число $2 \leqslant N \leqslant 10^{12}$.

Формат выходных данных

Вывести одно число — количество способов представить число N в виде произведения попарно различных множителей больших 1.

Пояснения к примеру

Имеется 7 различных способов представить число 48 в виде произведения (в том числе и вырожденного) попарно различных множителей больших 1: 48, $2 \cdot 24$, $3 \cdot 16$,

$4 \cdot 12$, $6 \cdot 8$, $2 \cdot 3 \cdot 8$, $2 \cdot 4 \cdot 6$.

Примеры

Пример №1

Стандартный ввод
48
Стандартный вывод
7

Пример программы-решения

Ниже представлено решение на языке C++.

```cpp
//Динамическое программирование
#include <bits/stdc++.h>

#define pb push_back
#define mp make_pair
#define x first
#define y second

#define all(x) (x).begin(), (x).end()
#define sz(a) (int)(a).size()
#define for0(i, n) for (int i = 0; i < (int)(n); i++)
#define for1(i, n) for (int i = 1; i <= (int)(n); i++)
#define int long long

using namespace std;
typedef pair<int, int> pii;
typedef long double ld;

vector<pii> raz;
set<int> del;
vector<int> rdel;

signed main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    int n, dn;
    cin >> n;
    dn = n;

    for(int i = 2; i <= 1e6+1 && dn > 1; i++) if(dn % i == 0){
        int t = 0;
            while(dn % i == 0){
                t++;
                dn /= i;
            }
        raz.pb({i, t});
    }

    if(dn > 1) raz.pb({dn, 1});

    del.insert(1);

    for(auto q : raz){
```

```
        for0(i, q.y){
            rdel.resize(0);
            for(auto w : del) rdel.pb(w * q.x);
            for(auto w : rdel) del.insert(w);
        }
    }

    rdel.resize(0);
    for(auto q : del) rdel.pb(q);
    int r = sz(rdel);

    map<int, int> rev;
    for0(i, r) rev[rdel[i]] = i;

    vector<vector<int> > dp(r, vector<int>(r, 0));

    dp[0][0] = 1;
    for(int i = 1; i < r; i++) dp[i][0] += dp[i-1][0];

    for(int j = 1; j < r; j++){
        for(int i = 1; i < r; i++) if(rdel[j] % rdel[i] == 0)
            dp[i][j] = dp[i-1][rev[rdel[j] / rdel[i]]];
        for(int i = 1; i < r; i++) dp[i][j] += dp[i-1][j];

    }
    cout<<dp[r-1][r-1];
}
```

Пример программы-решения

Ниже представлено решение на языке Python3.

```
from bisect import bisect_left

def binary_search(a, x):
    i = bisect_left(a, x)
    if i != len(a) and a[i] == x:
        return i
    else:
        return -1

n = int(input())
dividers = []
for i in range(1, math.ceil(math.sqrt(n))):
    if n % i == 0:
        dividers.append(i)
divs = dividers.copy()
if int(math.sqrt(n)) ** 2 == n:
    dividers.append(int(math.sqrt(n)))
for x in divs[::-1]:
    dividers.append(n // x)
dp = [[0 for i in range(len(dividers))] for j in range(len(dividers))]
dp[0][0] = 1
for a in range(len(dividers) - 1):
    pref_sum = 0
    for i in range(len(dividers) - 1):
        pref_sum += dp[a][i]
        dp[a][i] = pref_sum
        if dividers[a + 1] % dividers[i + 1] == 0:
```

```
            bs = binary_search(dividers, dividers[a + 1] // dividers[i + 1])
            if bs != -1:
                dp[a + 1][i + 1] = dp[bs][i]
print(sum(dp[-1]))
```

Вторая попытка. Задачи 8–11 класса

Задача I.1.2.1. Распознавание деталей (15 баллов)

Ваша задача — написать программу для распознавания деталей заданного вида на конвейере. Изображение нужной детали вводится при начале распознавания. Далее вводится изображение части конвейера, на котором могут находиться детали различных видов. Требуется распознать и выделить все изображения заданной детали.

Формат входных данных

Сначала приведено изображение искомой детали. Оно имеет размер 5×5. В 5 строках содержится по 5 символов «.» и «#», где решетки соответствуют детали, а точки — фону. Гарантируется, что изображение детали является 4-х связной фигурой. После изображения детали идет изображение текущего состояния конвейера. Это изображение имеет размер 10×20. В последующих 10 строках содержится по 20 символов. Каждая деталь на конвейере 4-х связная и имеет свой цвет, обозначенный малой буквой латинского алфавита. Разные детали обозначены разными буквами. Таким образом изображение конвейера может содержать буквы от «a» до «z» и символ «.», по прежнему обозначающий фон. На конвейере находится не более 26 деталей. Следует учитывать, что требуемые детали могут быть повернуты на угол кратный 90 градусов и/или лежать обратной стороной.

Формат выходных данных

Требуется вывести изображение конвейера, на котором все вхождения заданной детали выделены соответствующими буквами, но переведенными в верхний регистр, остальные символы должны остаться без изменений.

Примеры

Пример №1

Стандартный ввод
....##. ...## ...#. ....a..... ....aaac.....gg... ....a.ccc.....gg... ....b.c....d..... ....bbb...ddd..... .....b.....dhh..... ....f.....hh...e... ...ff.....h.eee... ....ff.....ee.. .....mmmmmmmmmmm

Стандартный вывод
.....A..... ....AAAC.....gg... ....A.CCC.....gg... ....B.C....d..... ....BBB...ddd..... .....B.....dHH..... ....F.....HH...e... ...FF.....H.eee... ....FF.....ee.. .....mmmmmmmmmmm

Пример программы-решения

Ниже представлено решение на языке C++.

```
1  //Сравнение с шаблоном, моделирование
2
3  #include <bits/stdc++.h>
4
5  #define pb push_back
6  #define mp make_pair
7  #define x first
8  #define y second
9
10 #define all(x) (x).begin(), (x).end()
11 #define sz(a) (int)(a).size()
12 #define for0(i, n) for (int i = 0; i < (int)(n); i++)
13 #define for1(i, n) for (int i = 1; i <= (int)(n); i++)
14 #define int long long
15
16 using namespace std;
17 typedef pair<int, int> pii;
18 typedef long double ld;
19 typedef vector<string> vs;
```

```
20
21  vs knv;
22  vector<vs > obr(8);
23  string s;
24  set<char> ans;
25
26  vs r_90(vs &a){
27      vs r = a;
28      for0(i, sz(a))
29          for0(j, sz(a[i])) r[j][sz(a) - i - 1] = a[i][j];
30      return r;
31  }
32
33  vs mirr(vs &a){
34      vs r = a;
35      for0(i, sz(r)) reverse(all(r[i]));
36      return r;
37  }
38
39  vs cut(int x, int y, char z){
40      vs r;
41      for0(i, 5) r.pb(knv[x+i].substr(y, 5));
42          for0(i, 5) for0(j, 5) if(r[i][j] != z) r[i][j] = '.'; else r[i][j] = '#';
43
44      return r;
45  }
46
47  int toko(vs &a, char z){
48      int r = 0;
49      for(auto q : a) for(auto w : q) r += (w == z);
50      return r;
51  }
52
53  signed main(){
54      ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);
55
56      for0(i, 5){
57          cin >> s;
58          obr[0].pb(s);
59      }
60
61      for1(i, 3) obr[i] = r_90(obr[i-1]);
62
63      obr[4] = mirr(obr[0]);
64
65      for(int i = 5; i < 8; i++) obr[i] = r_90(obr[i-1]);
66
67      int ko = toko(obr[0], '#');
68
69      string z;
70      for0(i, 30) z += ".";
71      for0(i, 5) knv.pb(z);
72      for0(i, 10) {
73          cin >> s;
74          for0(j, 5) s = "." + s;
75          for0(j, 5) s += ".";
76          knv.pb(s);
77      }
78      for0(i, 5) knv.pb(z);
79
```

```
80      map<char, int> msk;
81      for(auto q : knv)
82          for(auto w : q) msk[w]++;
83
84      for(auto q : msk) if(q.y == ko){
85              bool ok = 0;
86          for0(i, 14) for0(j, 24) {
87              vs d = cut(i, j, q.x);
88
89                  if(toko(d, '#') == ko){
90                      bool ok1;
91                      for0(j, 8){
92                          if(d == obr[j]){
93                                  ans.insert(q.x);
94                          }
95                      }
96                  }
97          }
98      }
99
100     for(int i = 5; i < 15; i++){
101         for(int j = 5; j < 25; j++)
102         if(ans.find(knv[i][j]) != ans.end()) cout<<(char)(knv[i][j] - 'a' + 'A');
103         else cout<<knv[i][j];
104         cout<<endl;
105     }
106 }
```

Задача I.1.2.2. Мерцающие звезды (22 баллов)

Современных звездных путешественников очень трудно удивить. Однако фирма Amazing Star Travel хочет предложить нечто новое: наблюдения за мерцающими звездами. Это очень эффектное явление, возникающее в тот момент, когда мощную звезду заслоняет планета. Для этого разработан маршрут между двумя точками A и B. Специалисты фирмы выделили N наиболее ярких звезд в видимой части космоса и отметили M крупных планет. Осталось подсчитать, сколько раз за время путешествия по отрезку AB путешественники насладятся видом мерцающей звезды.

Формат входных данных

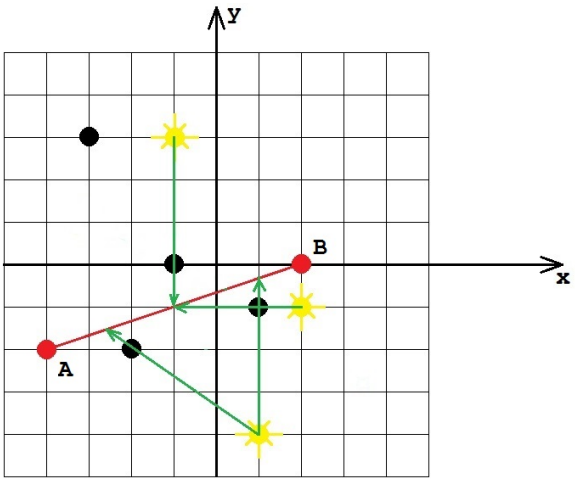
В первой строке содержится четыре целых числа через пробел XA, YA, XB, YB — координаты точек A и B. Во второй строке содержатся числа N и M, разделенные пробелом ($0 \leqslant N, M \leqslant 100$) — количество звезд и количество планет соответственно. В каждой из следующих N строк содержатся координаты очередной звезды. Далее в каждой из следующих M строк содержатся координаты очередной планеты. Все координаты целые, по модулю не превосходят 1000. Гарантируется, что никакие три точки из всех вышеперечисленных не находятся на одной прямой.

Формат выходных данных

В ответе нужно выдать одно число — количество случаев, когда при движении по отрезку из точки A в точку B какая-либо звезда будет заслонена от наблюдателя планетой. Если какие-либо две звезды мерцают одновременно, то это считается как

два независимых случая. Все упомянутые объекты считаем материальными точками, для упрощения вычислений все рассматриваем на плоскости. Помимо этого, согласно теории относительности, путешествие с точки зрения внешнего наблюдателя, совершается мгновенно, то есть положение звезд и планет за время путешествия не изменяется, однако для путешественников оно достаточно длительное, чтобы насладиться захватывающими видами.

Пояснения к примеру



Примеры

Пример №1

Стандартный ввод
-4 -2 2 0 3 4 -1 3 2 -1 1 -4 -3 3 -1 0 -2 -2 1 -1

Стандартный вывод
4

Пример программы-решения

Ниже представлено решение на языке C++.

```
1  //Целочисленно, сравнение площадей
2  #include <bits/stdc++.h>
3
4  #define pb push_back
5  #define mp make_pair
6  #define x first
```

```
#define y second

#define all(x) (x).begin(), (x).end()
#define sz(a) (int)(a).size()
#define for0(i, n) for (int i = 0; i < (int)(n); i++)
#define for1(i, n) for (int i = 1; i <= (int)(n); i++)
#define int long long

using namespace std;
typedef long double ld;
typedef pair<int, int> pii;

int sqt2(pii a, pii b, pii c){
    pii v1 = {b.x - a.x, b.y - a.y};
    pii v2 = {c.x - a.x, c.y - a.y};

    return abs(v1.x*v2.y - v1.y*v2.x);
}

signed main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    pii A, B;
    cin >> A.x >> A.y >> B.x >> B.y;

    int n, m;
    cin >> n >> m;
    vector<pii> s(n), p(m);
    for0(i, n) cin >> s[i].x >> s[i].y;
    for0(i, m) cin >> p[i].x >> p[i].y;

    int ans = 0;

    for0(i, n){
        int S = sqt2(A, B, s[i]);
        for0(j, m){
           int s1 = sqt2(A, B, p[j]);
           int s2 = sqt2(A, p[j], s[i]);
           int s3 = sqt2(B, p[j], s[i]);
           if(s1 + s2 + s3 == S) ans++;
        }
    }
    cout << ans;
}
```

Пример программы-решения

Ниже представлено решение на языке C++.

```
//вещественно, бинпоиском точку пересечения
#include <bits/stdc++.h>

#define pb push_back
#define mp make_pair
#define x first
#define y second

#define all(x) (x).begin(), (x).end()
#define sz(a) (int)(a).size()
```

```cpp
#define for0(i, n) for (int i = 0; i < (int)(n); i++)
#define for1(i, n) for (int i = 1; i <= (int)(n); i++)
#define int long long

using namespace std;
typedef long double ld;
typedef pair<ld, ld> pld;

ld eps = 1e-7;

struct line{
    ld A, B, C;
};

ld dist(pld a, pld b){
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
}

line L(pld a1, pld a2){
    ld A = a2.y - a1.y;
    ld B = a1.x - a2.x;
    ld C = a2.x*a1.y - a1.x*a2.y;

    line r = {A, B, C};
    return r;
}

int sgn(ld x){
    if(x < -eps) return -1;
    if(x > eps) return 1;
    return 0;
}

pld tp(pld A, pld B, pld s, pld p){
    line AB = L(A, B);

    ld L = 0, R = 1e6;

    pld v = {p.x - s.x, p.y - s.y};
    pld ab = {B.x - A.x, B.y - A.y};
    ld Fs = AB.A * (s.x + L * ab.x) + AB.B * (s.y + L * ab.y) + AB.C;

    if(abs(v.x*ab.y - v.y*ab.x) < eps) return{1e6, 1e6};

    while(R - L > eps){
        ld M = (L+R)/2.0;

        pld m = {s.x + M*v.x, s.y + M*v.y};

        ld F = AB.A * m.x + AB.B * m.y + AB.C;

        if(sgn(F) == sgn(Fs)) L = M; else R = M;
    }

    return {s.x + L*v.x, s.y + L*v.y};
}

signed main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

```

```
71      pld A, B;
72      cin >> A.x >> A.y >> B.x >> B.y;
73
74      int n, m;
75      cin >> n >> m;
76      vector<pld> s(n), p(m);
77      for0(i, n) cin >> s[i].x >> s[i].y;
78      for0(i, m) cin >> p[i].x >> p[i].y;
79
80      int ans = 0;
81
82      for0(i, n)
83      for0(j, m){
84          line AB = L(A, B);
85          ld tos = AB.A * s[i].x + AB.B * s[i].y + AB.C;
86          ld top = AB.A * p[j].x + AB.B * p[j].y + AB.C;
87
88          if(sgn(tos) == sgn(top)){
89
90              pld w = tp(A, B, s[i], p[j]);
91
92              if(abs(w.x) < 1e6 && abs(w.y) < 1e6){
93
94                  if(abs(dist(A, w) + dist(w, B) - dist(A, B)) < eps) {
95                          ans++;
96                  }
97              }
98
99          }
100     }
101
102     cout << ans;
103 }
```

Задача I.1.2.3. Послание внеземного разума 2 (23 баллов)

Профессор Персиков снова получил послание внеземного разума. Он по-прежнему считает, что доказательством этого является его периодичность. При этом период должен быть равен «константе Персикова» — числу P. К сожалению, послание не совсем периодичное. Если сказать более точно, то оно совсем не периодичное. Однако, это никак не останавливает исследователя космических глубин. Он говорит, что некоторые сигналы были неправильно откалиброваны, отфильтрованы и интерпретированы. По-прежнему мы будем считать, что все сигналы отображаются малыми буквами латинского алфавита, но в данной задаче все они распознаны и знаков вопроса нет. Тем не менее профессор, согласно своей теории, может заявить, что все вхождения такой-то буквы интерпретированы неверно и их **все** следует заменить на вхождения какой-то другой (одной и той же) буквы. Более строго: пусть на позициях $p_{i_1}, p_{i_2}, \ldots p_{i_k}$ и только на них в последовательности находится одна и та же буква. Профессор может выбрать любую другую букву (как встречающуюся в слове, так и не встречающуюся) и поставить ее во всех этих позициях. Например, в слове *qqzbbacabadaba* он может выбрать все вхождения буквы b и заменить их на букву a, получив слово *qqzaaacaaadaaa* (это считается одной заменой, независимо от числа вхождений). Очевидно, что таким образом любое послание можно сделать P-периодическим, но профессор заинтересован сделать как можно меньше таких замен. При этом он хочет получить лексикографически минимальное послание.

Формат входных данных

В первой строке содержится число P — константа Персикова ($1 \leqslant P \leqslant 10^5$). В следующей строке содержится непустая последовательность, состоящая из малых букв латиницы. Длина этой строки не превосходит $2 \cdot 10^2$.

Формат выходных данных

Вывести строку, которая получается из исходной путем минимального числа операций замены вхождений всех букв одного вида на вхождения какой-то (одной и той же) другой буквы. Итоговая строка должна быть P-периодической, то есть любые две ее буквы, расстояние между которыми кратно P должны совпадать. Среди всех таких строк вывести лексикографически минимальную (первую в алфавитном порядке).

Примеры

Пример №1

Стандартный ввод
4 qqzbbacabadaba

Стандартный вывод
aacaaacaaacaaa

Пример программы-решения

Ниже представлено решение на языке C++.

```
1  //граф на буквах, обход в глубину
2  #include <bits/stdc++.h>
3
4  #define pb push_back
5  #define mp make_pair
6  #define x first
7  #define y second
8
9  #define all(x) (x).begin(), (x).end()
10 #define sz(a) (int)(a).size()
11 #define for0(i, n) for (int i = 0; i < (int)(n); i++)
12 #define for1(i, n) for (int i = 1; i <= (int)(n); i++)
13 #define int long long
14
15 using namespace std;
16 typedef long double ld;
17 typedef pair<int, int> pii;
18
19 map<char, set<char> > G;
20 map<char, int> rzb;
21 int num = 0;
22 vector<set<char> > toans;
23 set<char> emp;
24
25 void dfs(char a){
26     toans[rzb[a]].insert(a);
```

```
    for(auto to : G[a]) if(rzb[to] == 0) {
        rzb[to] = rzb[a];
        dfs(to);
    }
}

signed main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    int p;
    cin >> p;
    string s;
    cin >> s;

    vector<set<char> > op(p);

    for0(i, sz(s))  op[i%p].insert(s[i]);

    for0(i, p)
        for(auto q : op[i])
        for(auto w : op[i]) {
            G[q].insert(w);
            G[w].insert(q);
        }

        toans.pb(emp);
        for(auto q : G)
        if(rzb[q.x] == 0){
            num++;
            toans.pb(emp);
            rzb[q.x] = num;
            dfs(q.x);
        }

    for0(i, sz(s)) s[i] = *(toans[rzb[s[i]]].begin());
    cout<<s;
}
```

Задача I.1.2.4. Проект «Ровные дороги» (10 баллов)

При проектировании новой автодороги было принято решение сделать ее не более чем из двух абсолютно горизонтальных участков. Будущую трассу разбили на n равных по длине малых отрезков. Будем считать, что в пределах одного малого отрезка местность имеет одну и ту же высоту h_i. При этом в целях эффективной трансформации местности требуется для выравнивания использовать исключительно грунт с этой же трассы. Это означает, что можно с некоторого малого отрезка высоты h_i взять некоторое количество грунта d так, что высота этого участка станет $h_i - d > 0$. Далее эти d единиц грунта обязательно нужно поместить на другой малый отрезок высоты h_j так, что его высота станет $h_j + d$. Перемещать грунт можно только в пределах одного из двух выбранных участков, то есть отрезки номер i и номер j должны принадлежать одному и тому же горизонтальному после выравнивания участку. Технические требования таковы, что эти два выровненных участка должны иметь целочисленную высоту над уровнем моря. Таким образом, для начала требуется определить сколькими способами можно переместить грунт с возвышенностей в низины так, что образуется не более чем два горизонтальных участка трассы, каждый из которых имеет целочисленную высоту.

Формат входных данных

В первой строке содержится число n — количество малых отрезков, на которые разбили трассу, $2 \leqslant n \leqslant 10^2$. Во второй строке указаны высоты h_i этих отрезков через пробел в порядке слева направо, $1 \leqslant h_i \leqslant 2 \cdot 10^4$.

Формат выходных данных

Вывести количество способов достичь требуемых показателей. Если у двух разбиений трассы в результате выравнивания конечные результаты совпадают, то они считаются одинаковыми.

Пояснения к примеру

Пример №1

Можно указать следующие разбиения не более чем на два участка, после выравнивания в пределах которых образуются целые высоты: [10 4] и [2 7 5 8 6 6 15], результат выравнивания: [7 7] и [7 7 7 7 7 7 7]

[10 4 2 7 5 8] и [6 6 15], результат выравнивания: [6 6 6 6 6 6] и [9 9 9]

[10 4 2 7 5 8 6 6] и [15], результат выравнивания: [6 6 6 6 6 6 6 6] и [15]

[10 4 2 7 5 8 6 6 15], результат выравнивания: [7 7 7 7 7 7 7 7 7].

Результаты выравнивания в первом и последнем способе совпадают, таким образом общий ответ равен 3.

Пример №2

Во втором примере получатся следующие варианты разбиения и выравнивания: [3] [5 2 7 6 4 5 8 1 7], результат выравнивания: [3] и [5 5 5 5 5 5 5 5 5]

[3 5] [2 7 6 4 5 8 1 7], результат выравнивания: [4 4] и [5 5 5 5 5 5 5 5]

[3 5 2 7 6 4 5 8] [1 7], результат выравнивания: [5 5 5 5 5 5 5 5] и [4 4]

Примеры

Пример №1

Стандартный ввод
9 10 4 2 7 5 8 6 6 15
Стандартный вывод
3

Пример №2

Стандартный ввод
10 3 5 2 7 6 4 5 8 1 7
Стандартный вывод
3

Пример программы-решения

Ниже представлено решение на языке C++.

```cpp
#include <bits/stdc++.h>

#define pb push_back
#define mp make_pair
#define x first
#define y second

#define all(x) (x).begin(), (x).end()
#define sz(a) (int)(a).size()
#define for0(i, n) for (int i = 0; i < (int)(n); i++)
#define for1(i, n) for (int i = 1; i <= (int)(n); i++)
#define int long long

using namespace std;
typedef long double ld;
typedef pair<int, int> pii;

signed main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    int n;
    cin >> n;
    vector<int> v(n);
    for0(i, n) cin >> v[i];

    vector<int> suml(n), sumr(n);

    int sum = 0;
    for0(i, n){
        sum += v[i];
        suml[i] = sum;
    }

    sum = 0;
    for(int i = n-1; i >= 0; i--){
        sum += v[i];
        sumr[i] = sum;
    }

     int ans = 0, o1 = 0;
    if(sum % n == 0){
        ans++;
        o1++;
    }

    for(int i = 0; i < n-1; i++){
        if(suml[i] % (i+1) == 0 && sumr[i+1] %(n - i - 1) == 0){
            if(o1 == 1){
                if(suml[i] / (i+1) != sum / n) {
                        ans++;
                }
            }
            else{
                ans++;
            }
        }
```

```
57      }
58      cout << ans;
59  }
```

Задача I.1.2.5. Прогулка по мостам (30 баллов)

Возможно, вы знаете историю про то, как Эйлер гулял по мостам Кенигсберга. Допустим теперь, что Эйлер попал на некий архипелаг, между некоторыми островами которого имеются мосты. Мосты построены таким образом, что между любыми двумя островами архипелага можно построить путь по этим мостам, причем этот путь единственный. Все острова пронумерованы от 1 до n. Эйлер находится на острове номер 1 и желает совершить следующую прогулку: с острова номер 1 он хочет дойти до острова номер 2, далее до острова номер 3 и так далее до острова номер n. С него он хочет вернуться на остров номер 1. Он прекрасно понимает, что при этом ему придется посещать некоторые мосты по многу раз. Осталось выяснить, сколько раз суммарно он пройдет по мостам во время своей прогулки.

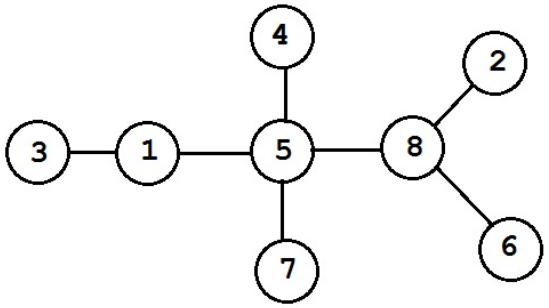
Формат входных данных

В первой строке содержится число n — количество островов в архипелаге, $2 \leqslant n \leqslant 10^5$. В следующих $n-1$ строках содержатся по два числа a_i и b_i через пробел — номера островов, соединенных мостом, $1 \leqslant a_i, b_i \leqslant n, a_i \neq b_i$. Гарантируется, что между любыми двумя островами архипелага можно построить путь по этим мостам, причем этот путь единственный.

Формат выходных данных

Вывести длину описанного пути, то есть суммарное количество пересечений мостов, которые придется совершить.

Пояснения к примеру



На рисунке вы видите расположение мостов в архипелаге из примера. Прогулка Эйлера будет проходить следующим образом:

1 — 5 — 8 — 2 : 3 моста;

2 — 8 — 5 — 1 — 3 : 4 моста;

3 — 1 — 5 — 4 : 3 моста;

4 — 5 : 1 мост;

5 — 8 — 6 : 2 моста;

6 — 8 — 5 — 7 : 3 моста;

7 — 5 — 8 : 2 моста;

8 — 5 — 1 : 2 моста.

Итого он пройдет через $3 + 4 + 3 + 1 + 2 + 3 + 2 + 2 = 20$ мостов.

Примеры

Пример №1

Стандартный ввод
8 5 8 1 3 8 6 7 5 2 8 1 5 4 5
Стандартный вывод
20

Пример программы-решения

Ниже представлено решение на языке C++.

```
//lca двоичным подъемом, обход в глубину
#include <bits/stdc++.h>

#define pb push_back
#define mp make_pair
#define x first
#define y second

#define all(x) (x).begin(), (x).end()
#define sz(a) (int)(a).size()
#define for0(i, n) for (int i = 0; i < (int)(n); i++)
#define for1(i, n) for (int i = 1; i <= (int)(n); i++)
#define int long long

using namespace std;
typedef long double ld;
typedef pair<int, int> pii;

int n, a, b, timer;
vector<vector<int> > G;
const int N = 1e5 + 228;
const int LG = 18;
vector<int> dep, in, out;
```

```
24
25  int up[LG][N];
26
27  void dfs(int a, int p) {
28          in[a] = timer++;
29          up[0][a] = p;
30          for (int i = 1; i < LG; i++) {
31                  up[i][a] = up[i - 1][up[i - 1][a]];
32          }
33          for (int to : G[a]) {
34                  if (to != p) {
35                          dep[to] = dep[a] + 1;
36                          dfs(to, a);
37                  }
38          }
39          out[a] = timer;
40  }
41
42  bool upper(int u, int v) {
43          return in[u] <= in[v] && out[v] <= out[u];
44  }
45
46  int lca(int u, int v) {
47          if (in[u] > in[v]) {
48                  swap(u, v);
49          }
50          if (upper(u, v)) {
51                  return u;
52          }
53          for (int i = LG - 1; i >= 0; i--) {
54                  if (!upper(up[i][u], v)) {
55                          u = up[i][u];
56                  }
57          }
58          return up[0][u];
59  }
60
61  int dist(int u, int v) {
62          int t = lca(u, v);
63          return dep[u] + dep[v] - 2 * dep[t];
64  }
65
66  signed main(){
67      ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);
68
69      cin >> n;
70      G.resize(n);
71      for0(i, n-1){
72          cin >> a >> b;
73          a--;
74          b--;
75          G[a].pb(b);
76          G[b].pb(a);
77      }
78      dep.resize(n, 0);
79          in.resize(n);
80          out.resize(n);
81          dfs(0, 0);
82
83          int ans = 0;
```

```
84          for0(i, n){
85          ans += dist(i, (i+1)%n);
86          }
87
88          cout<<ans;
89  }
```

Пример программы-решения

Ниже представлено решение на языке C++.

```
1   // lca двоичным подъемом, обход в ширину без глубокой рекурсии
2   #include <bits/stdc++.h>
3
4   #define pb push_back
5   #define mp make_pair
6   #define x first
7   #define y second
8
9   #define all(x) (x).begin(), (x).end()
10  #define sz(a) (int)(a).size()
11  #define for0(i, n) for (int i = 0; i < (int)(n); i++)
12  #define for1(i, n) for (int i = 1; i <= (int)(n); i++)
13  #define int long long
14
15  using namespace std;
16  typedef long double ld;
17  typedef pair<int, int> pii;
18
19  int n, a, b, timer;
20  vector<vector<int> > G;
21  const int N = 1e5 + 228;
22  const int LG = 18;
23  vector<int> dep, in, out;
24
25  int up[LG][N];
26
27  bool upper(int u, int v) {
28          return in[u] <= in[v] && out[v] <= out[u];
29  }
30
31  int lca(int u, int v) {
32          if (in[u] > in[v]) {
33                  swap(u, v);
34          }
35          if (upper(u, v)) {
36                  return u;
37          }
38          for (int i = LG - 1; i >= 0; i--) {
39                  if (!upper(up[i][u], v)) {
40                          u = up[i][u];
41                  }
42          }
43
44          return up[0][u];
45  }
46
47  int dist(int u, int v) {
48          int t = lca(u, v);
```

```
49          return dep[u] + dep[v] - 2 * dep[t];
50  }
51
52  signed main(){
53      ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);
54
55      cin >> n;
56      G.resize(n);
57      for0(i, n-1){
58          cin >> a >> b;
59          a--;
60          b--;
61          G[a].pb(b);
62          G[b].pb(a);
63      }
64      dep.resize(n, 0);
65          in.resize(n);
66          out.resize(n, 0);
67
68          deque<int> och;
69          vector<int> par(n), d(n, 1), oo;
70          och.pb(0);
71          oo.pb(0);
72          dep[0] = 0;
73          par[0] = 0;
74          while(sz(och) > 0){
75        int tb = och[0];
76        och.pop_front();
77        for(auto to : G[tb])  if(to != par[tb]){
78          dep[to] = dep[tb] + 1;
79          par[to] = tb;
80
81                  up[0][to] = tb;
82          for (int i = 1; i < LG; i++) {
83                  up[i][to] = up[i - 1][up[i - 1][to]];
84          }
85          och.pb(to);
86          oo.pb(to);
87        }
88
89          }
90          for(int i = sz(oo)-1; i >= 1; i--)
91          d[par[oo[i]]] += d[oo[i]];
92
93      in[0] = 0;
94      for(int i = 1; i < sz(oo); i++){
95          in[oo[i]] = out[par[oo[i]]]+1;
96          out[oo[i]] = in[oo[i]] + 1;
97          out[par[oo[i]]] += 2*d[oo[i]];
98      }
99
100         int ans = 0;
101         for0(i, n){
102         ans += dist(i, (i+1)%n);
103         }
104         cout<<ans;
105 }
```

Третья попытка. Задачи 8–11 класса

Задача I.1.3.1. Послание Аресибо (15 баллов)

Имеется прямоугольное изображение, разбитое на единичные квадратики, размер этого изображения $n \times m$, $(5 \leqslant n, m)$. Каждый его квадратик либо черный либо белый. Известно, что на этом изображении нарисована черным цветом на белом фоне одна четырехсвязная фигура. Фигура называется четырехсвязной, если между любыми двумя ее клетками можно построить путь по клеткам этой фигуры, в котором любые две рядом стоящие клетки являются соседними в изображении либо по горизонтали либо по вертикали. Далее изображение разбили на строки и соединили их в одну большую строку без пробелов и разделителей. Длина этой строки $n \cdot m$. После этого ее отправили в направлении шарового звездного скопления M13, находящегося на расстоянии 25000 световых лет в созвездии Геркулеса. Вы обитатель M13 и перед вами поставили задачу восстановить изображение, исходя из информации о его четырехсвязности. Гарантируется, что решение единственно.

Формат входных данных

На вход подается принятая строка. Она состоит из знаков «.» и «#». Точки соответствуют фону, а решетки — изображенной фигуре. Длина строки не превосходит 5184.

Формат выходных данных

Требуется вывести изображение в исходном виде, то есть разбить его на строки одинаковой длины и расположить их друг под другом.

Примеры

Пример №1

Стандартный ввод
...........###...#..#.#...#..###...#...#....#########..####..#. .####..#..#..#..#.##..##..

Стандартный вывод
....###... #..#.#... #..###... #...#.... ######### ..####..# ..####..# ..#..#..# .##..##..

Пример программы-решения

Ниже представлено решение на языке C++.

```cpp
//dfs, нахождение 4-связной области

#include<bits/stdc++.h>

#define pb push_back
#define mp make_pair
#define x first
#define y second

#define all(x) (x).begin(), (x).end()
#define sz(a) (int)(a).size()
#define for0(i, n) for (int i = 0; i < (int)(n); i++)
#define for1(i, n) for (int i = 1; i <= (int)(n); i++)
#define int long long

using namespace std;
typedef pair<int, int> pii;
typedef long double ld;
typedef vector<string> vs;

int k;
int dx[4] = {-1, 0, 1, 0};
int dy[4] = {0, 1, 0, -1};

bool in_tr(int x, int y, int h, int w){
    return (x >= 0 && y >= 0 && x < h && y < w);
}

void dfs(vector<string> &v, int x, int y){
    v[x][y] = '#';
    k++;
    for0(i, 4){
        int nx = x + dx[i];
        int ny = y + dy[i];

        if(in_tr(nx, ny, sz(v), sz(v[0])) && v[nx][ny] == '$')
            dfs(v, nx, ny);
    }
}

signed main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    string iss;
    cin >> iss;

    int n = sz(iss);
    for0(i, n) if(iss[i] == '#') iss[i] = '$';
    int kk = 0;
    for(auto q : iss)
        kk += (q == '$');

    for(int i = 5; i < n; i++)
    if(n % i == 0 && n/i >= 5){
        string s = iss;
        vector<string> v;
        for0(j, n/i){
            v.pb(s.substr(0, i));
          if(sz(s) > i)  s = s.substr(i);
        }
```

```
61
62          k = 0;
63          int sx, sy;
64          for0(i, sz(v))
65              for0(j, sz(v[i]))
66                  if(v[i][j] == '$'){
67                      sx = i;
68                      sy = j;
69                  }
70
71          dfs(v, sx, sy);
72
73          if(k == kk)
74          for0(i, sz(v)) cout<<v[i]<<endl;
75      }
76  }
```

Задача I.1.3.2. Гравитационный параллакс (10 баллов)

Специалисты фирмы Amazing Star Travel очень ответственно относятся к безопасности своих клиентов. Прежде чем маршрут из точки A в точку B будет одобрен для увлекательных путешествий, он проходит всестороннюю экспертизу. Помимо очевидных опасностей, таких как наличие на маршруте вредных космических излучений, активность пиратов и прогнозирования вспышек сверхновых, в числе прочих анализируется и множество других, более скучных. Современные исследования показали, что на пролетающий на околосветовых скоростях по отрезку AB космический корабль негативное влияние может оказать гравитационное поле звезд и планет. С удалением звезды или планеты от отрезка, это влияние сначала возрастает, а потом скачкообразно падает до нуля. Особенно опасно, если звезда и планета расположены по разные стороны прямой AB. Тогда их воздействие мультиплицируется, то есть перемножается. Особо следует отметить, что эффект мультиплицирования возникает только в паре звезда-планета, а, например, для пар звезда-звезда или планета-планета не обнаружен. Кроме того, этот эффект не наблюдается, если звезда и планета расположены по одну сторону от прямой AB. Осталось сказать, что можно считать эффект гравитационного воздействия равным площади треугольника ABC, где A и B — начало и конец маршрута, а C — место расположения звезды или планеты. Вас просят оценить опасность заданного отрезка AB, то есть найти максимальную величину гравитационного воздействия на отрезок среди всех пар звезда-планета, расположенных по разные стороны прямой AB. Не стоит забывать и о простом воздействии отдельных объектов, которое может оказаться даже больше, чем воздействие пары.

Формат входных данных

В первой строке содержится четыре целых числа через пробел XA, YA, XB, YB — координаты точек A и B. Во второй строке содержатся числа N и M, разделенные пробелом ($0 \leqslant N, M \leqslant 100$) — количество звезд и количество планет соответственно. В каждой из следующих N строк содержатся координаты очередной звезды. Далее в каждой из следующих M строк содержатся координаты очередной планеты. Указанные звезды и планеты, и только они могут оказать ненулевое гравитационное влияние на отрезок AB. Все координаты целые, по модулю не превосходят 1000.

Гарантируется, что никакие три точки из всех вышеперечисленных не находятся на одной прямой.

Формат выходных данных

В ответе нужно выдать ответ на задачу — одно вещественное число округленное ровно до двух знаков после точки. Разделитель между целой и дробной частями ответа — точка.

Пояснения к примеру

Итоговое наибольшее воздействие на отрезок AB оказывают выделенные треугольниками звезда и планета. Воздействие звезды равно площади треугольника ABC, которая равна 11, воздействие планеты равно площади треугольника ABD, которая равна 14. Так как они находятся по разные стороны прямой AB, то их общее воздействие равно произведению $11 \cdot 14 = 154,00$

Примеры

Пример №1

Стандартный ввод
-4 -2 2 0 3 4 -2 4 2 -1 1 -4 -3 3 -1 0 -2 -2 1 -1

Стандартный вывод
154.00

Пример программы-решения

Ниже представлено решение на языке C++.

```
1  //разделение - знаки векторного произведения, площадь - модуль векторного
2  // произведения
3  #include<bits/stdc++.h>
4
5  #define pb push_back
6  #define mp make_pair
7  #define x first
8  #define y second
9
10 #define all(x) (x).begin(), (x).end()
11 #define sz(a) (int)(a).size()
12 #define for0(i, n) for (int i = 0; i < (int)(n); i++)
13 #define for1(i, n) for (int i = 1; i <= (int)(n); i++)
14 #define int long long
15
```

```cpp
using namespace std;
typedef long double ld;
typedef pair<int, int> pii;

int vprod(pii a, pii b){
    return a.x*b.y - b.x*a.y;
}

int sgn_vprod(pii a, pii b){
    int vp = vprod(a, b);
    if(vp < 0) return -1;
    if(vp > 0) return 1;
    return 0;
}

int sqt2(pii a, pii b, pii c){
    pii v1 = {b.x - a.x, b.y - a.y};
    pii v2 = {c.x - a.x, c.y - a.y};

    return abs(vprod(v1, v2));
}

signed main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    pii A, B;
    cin >> A.x >> A.y >> B.x >> B.y;
    pii AB = {B.x - A.x, B.y - A.y};

    int n, m;
    cin >> n >> m;
    vector<pii> s(n), p(m);
    for0(i, n) cin >> s[i].x >> s[i].y;
    for0(i, m) cin >> p[i].x >> p[i].y;

    double ans = 0;

    for0(i, n)
    ans = max(ans, sqt2(A, B, s[i]) / 2.0);

    for0(j, m)
    ans = max(ans, sqt2(A, B, p[j]) / 2.0);

    for0(i, n){
        pii Bs = {s[i].x - B.x, s[i].y - B.y};

        for0(j, m){
            pii Bp = {p[j].x - B.x, p[j].y - B.y};

            if(sgn_vprod(AB, Bs) != sgn_vprod(AB, Bp)){
                ans = max(ans, sqt2(A, B, s[i]) * sqt2(A, B, p[j]) / 4.0);
            }
        }
    }

    cout.precision(2);
    cout<<fixed<<ans;
}
```

Пример программы-решения

Ниже представлено решение на языке C++.

```
//разделение - знаки подстановки в общее уравнение прямой, площадь --- формула
// Герона
#include<bits/stdc++.h>

#define pb push_back
#define mp make_pair
#define x first
#define y second

#define all(x) (x).begin(), (x).end()
#define sz(a) (int)(a).size()
#define for0(i, n) for (int i = 0; i < (int)(n); i++)
#define for1(i, n) for (int i = 1; i <= (int)(n); i++)
#define int long long

using namespace std;
typedef long double ld;
typedef pair<double, double> pii;
pii A, B;

struct line{
    double A, B, C;
};

line L(pii a, pii b){
    line r;
    r.A = b.y - a.y;
    r.B = a.x - b.x;
    r.C = b.x*a.y - a.x*b.y;

    return r;
}

double dist(pii a, pii b){
    return sqrt((a.x - b.x)*(a.x - b.x)+(a.y - b.y)*(a.y - b.y));
}


double sqt2(pii a, pii b, pii c){
    double p = (dist(a, b) + dist(b, c) + dist(a,c)) / 2.0;
    return 2*(sqrt(p * (p-dist(a,b)) * (p-dist(b,c)) * (p-dist(a,c)) ));
}

signed main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    cin >> A.x >> A.y >> B.x >> B.y;

    int n, m;
    cin >> n >> m;
    vector<pii> s(n), p(m);
    for0(i, n) cin >> s[i].x >> s[i].y;
    for0(i, m) cin >> p[i].x >> p[i].y;

    double ans = 0;
    line AB = L(A, B);
```

```
57
58      for0(i, n)
59      ans = max(ans, sqt2(A, B, s[i]) / 2.0);
60
61      for0(j, m)
62      ans = max(ans, sqt2(A, B, p[j]) / 2.0);
63
64      for0(i, n){
65          pii Bs = {s[i].x - B.x, s[i].y - B.y};
66          double stoAB = AB.A*s[i].x + AB.B*s[i].y + AB.C;
67
68          for0(j, m){
69              pii Bp = {p[j].x - B.x, p[j].y - B.y};
70
71              double ptoAB = AB.A*p[j].x + AB.B*p[j].y + AB.C;
72
73
74              if(stoAB < 0 && ptoAB > 0 || stoAB > 0 && ptoAB < 0){
75                  ans = max(ans, sqt2(A, B, s[i]) * sqt2(A, B, p[j]) / 4.0);
76              }
77          }
78      }
79
80      cout.precision(2);
81      cout<<fixed<<ans;
82  }
```

Задача I.1.3.3. Послание внеземного разума 3 (25 баллов)

Профессор Персиков снова на первых полосах новостных агрегаторов! Он сделал очередное гениальное предположение по поводу новой последовательности сигналов из глубин космоса.

- во-первых, говорит профессор, для подтверждения искусственности происхождения сигнала достаточно, чтобы он был периодическим с периодом, **не превосходящим известную** «константу Персикова» P.
- во-вторых, согласно закону диффузного рассеивания информации в некогерентных пространствах при нелинейно возрастающем коэффициенте Заальшютца – Персикова, качество сигнала падает при росте времени его передачи, что означает, что некоторое окончание последовательности можно отбросить как недостоверно опознанное.

Таким образом, все что осталось профессору — отбросить несколько подряд идущих сигналов из конца последовательности так, чтобы она стала периодической с периодом, не превосходящим P. Как обычно, профессор заинтересован удалить как можно меньше сигналов.

Формат входных данных

В первой строке содержится число P — константа Персикова ($1 \leqslant P \leqslant 10^5$). В следующей строке содержится непустая последовательность, состоящая из малых букв латиницы — послание из космоса. Длина этой строки не превосходит $2 \cdot 10^5$.

Формат выходных данных

Вывести одно число — минимальное количество символов, которые нужно удалить из конца последовательности, чтобы она стала периодической с периодом T, не превосходящим P. Последовательность имеет период T, если для любых двух ее символов, расстояние между которыми кратно T верно, что они совпадают.

Пояснения к примеру

Рассмотрим исходную последовательность **abcabcaabcabcabab**. Если из нее ничего не удалять, то ее наименьший период будет равен 15. Если удалить из ее конца одну букву, то период **abcabcaabcabcaba** не изменится, если же удалить две буквы, период **abcabcaabcabcab** станет равен 10. Если удалить три буквы, период abcabcaabcabca равен 7. И этого достаточно для первого теста. Чтобы период не превосходил 3, придется удалить из конца 10 символов и получить **abcabca** с периодом 3. Ну а период 2 можно получить только у начала **ab**, для этого придется удалить 15 символов.

Примеры

Пример №1

Стандартный ввод
8 abcabcaabcabcabab
Стандартный вывод
3

Пример программы-решения

Ниже представлено решение на языке C++.

```
1  //префикс-функция, метод Кнута-Морриса-Пратта
2
3  #include <bits/stdc++.h>
4
5  #define pb push_back
6  #define mp make_pair
7  #define x first
8  #define y second
9
10 #define all(x) (x).begin(), (x).end()
11 #define sz(a) (int)(a).size()
12 #define for0(i, n) for (int i = 0; i < (int)(n); i++)
13 #define for1(i, n) for (int i = 1; i <= (int)(n); i++)
14 #define int long long
15
16 using namespace std;
17 typedef pair<int, int> pii;
18 typedef long double ld;
19 typedef vector<string> vs;
20
21 vector<int> pf (string s) {
22         int n = sz(s);
```

```
        vector<int> pi(n);
        for (int i=1; i<n; ++i) {
                int j = pi[i-1];
                while (j > 0 && s[i] != s[j])
                        j = pi[j-1];
                if (s[i] == s[j])  ++j;
                pi[i] = j;
        }
        return pi;
}

signed main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    int p;
    string s;

    cin >> p >> s;

    vector<int> kmp = pf(s);

    for(int i = sz(kmp)-1; i >= 0; i--)
        if(i - kmp[i] + 1 <= p)
        return cout<<sz(s)-1 - i, 0;
}
```

Задача I.1.3.4. Проект «Ровные дороги» 2 (30 баллов)

При проектировании новой автодороги было принято решение сделать ее не более чем из двух абсолютно горизонтальных участков. Будущую трассу разбили на n равных по длине малых отрезков. Будем считать, что в пределах одного малого отрезка местность имеет одну и ту же высоту h_i. При этом в целях эффективной трансформации местности требуется для выравнивания использовать исключительно грунт с этой же трассы. Это означает, что можно с некоторого малого отрезка высоты h_i взять некоторое количество грунта d так, что высота этого участка станет $h_i - d > 0$. Далее эти d единиц грунта обязательно нужно переместить на другой малый отрезок высоты h_j так, что его высота станет $h_j + d$. Перемещать грунт можно только в пределах одного из двух выбранных участков, то есть отрезки номер i и номер j должны принадлежать одному и тому же горизонтальному после выравнивания участку. В данной версии задачи высоты выравниваемых участков могут быть любыми положительными, в том числе и не целыми числами. Следующим важным вопросом при строительстве являются трудозатраты. По этой причине требуется выбрать такое разбиение трассы ровно на два непустых участка, чтобы суммарный объем перемещенного грунта был минимально возможным.

Формат входных данных

В первой строке содержится число n — количество малых отрезков, на которые разбили трассу, $2 \leqslant n \leqslant 2 \cdot 10^5$. Во второй строке указаны высоты h_i этих отрезков через пробел в порядке слева направо, $1 \leqslant h_i \leqslant 2 \cdot 10^5$.

Формат выходных данных

Вывести два ненулевых числа a и b через пробел. Их сумма должна равняться n. Отрезки с номерами с первого по a-й включительно будут принадлежать первому выровненному участку, отрезки с номерами от $a + 1$ до n будут принадлежать второму выровненному участку. При этом суммарный объем грунта, перемещенного для такого выравнивания, должен быть минимальным среди всех возможных разбиений трассы на два участка. Если минимальных вариантов несколько вывести тот, у которого число a меньше.

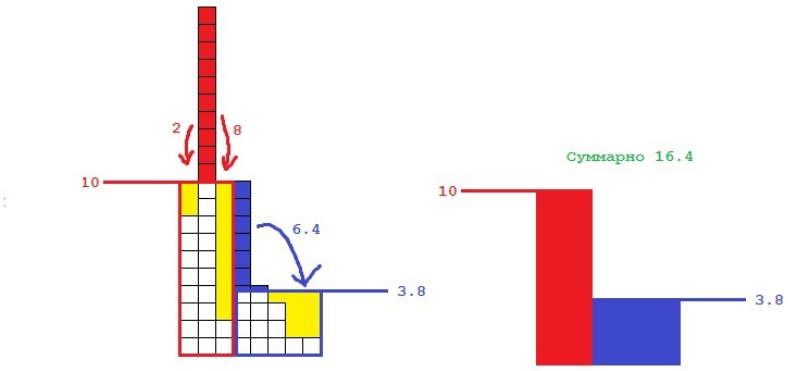
Пояснения к примеру

На следующих рисунках приведены рассуждения для трех вариантов разбиения трассы на два участка,

$a = 3, b = 5$:

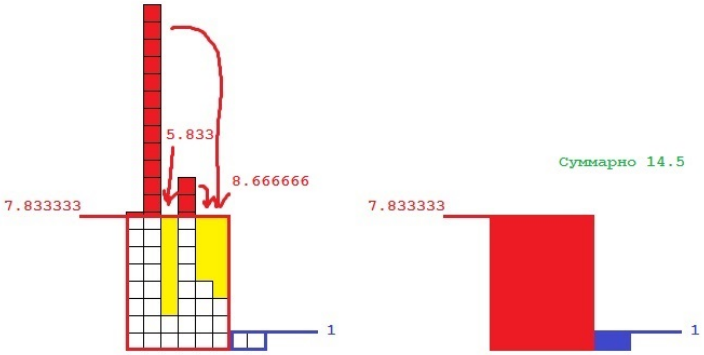
```
a = 3     b = 5
8 20 2 10 4 3 1 1
(8 + 20 + 2) / 3 = 10
на левом  участке будет перемещено 10 единиц

(10 + 4 + 3 + 1 + 1) / 5 = 3.8
на правом участке будет перемещено 6.4 единиц
Всего будет перемещено 10 + 6.4 = 16.4 единиц
```



$a = 6, b = 2$:

```
a = 6       b = 2
8 20 2 10 4 3 1 1
(8 + 20 + 2 + 10 + 4 + 3) / 6 = 7.8333333333
на левом  участке будет перемещено 14.5 единиц
(1 + 1) / 2 = 1
на правом участке будет перемещено 0 единиц
Всего будет перемещено 14.5 + 0 = 14.5 единиц
```



$a = 4, b = 4$:

a = 4 b = 4

8 20 2 10 4 3 1 1

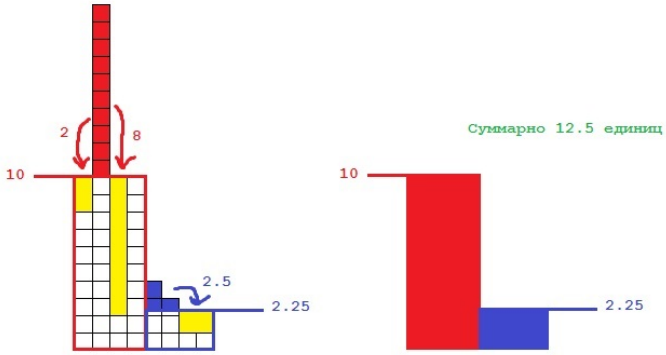
(8 + 20 + 2 + 10) / 4 = 10

на левом участке будет перемещено 10 единиц

(4 + 3 + 1 + 1) / 4 = 2.25

на правом участке будет перемещено 2.5 единиц

Всего будет перемещено 10 + 2.5 = 12.5 единиц



Если перебрать все варианты разбиения трассы на две части, самым оптимальным будет разбиение 4 4 с результатом 12.5.

Примеры

Пример №1

Стандартный ввод
8 8 20 2 10 4 3 1 1
Стандартный вывод
4 4

Пример программы-решения

Ниже представлено решение на языке C++.

```
1  //два дерева отрезков на сумму
2
3  #include<bits/stdc++.h>
4
5  #define pb push_back
6  #define mp make_pair
7  #define x first
8  #define y second
9
10 #define all(x) (x).begin(), (x).end()
```

```
#define sz(a) (int)(a).size()
#define for0(i, n) for (int i = 0; i < (int)(n); i++)
#define for1(i, n) for (int i = 1; i <= (int)(n); i++)
#define int long long

using namespace std;
typedef pair<int, int> pii;
typedef long double ld;
typedef vector<string> vs;

int n, ans = 0, N = 262144;
vector <int> kl, su;
double eps = 1e-7;

int get(int l, int r, vector<int> &t){
    int res=0;
    for(l+=N, r+=N; l<=r; l=(l+1)>>1, r=(r-1)>>1){
        if(l & 1)
            res += t[l];
        if(!(r & 1))
            res += t[r];
    }
    return res;
}
void upd(int pos, int val, vector<int> &t)
{
    pos += N;
    t[pos] += val;
    for(pos >>= 1; pos; pos >>= 1)
    {
        t[pos] = t[pos * 2] + t[pos * 2 + 1];
    }
}

signed main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    cout.precision(3);

    cin >> n;
    vector<int> h(n);
    vector<double> ld(n), rd(n), lsa(n), rsa(n);

    for0(i, n)
        cin >> h[i];

    kl.resize(2*N, 0);
    su.resize(2*N, 0);
    double sum = 0;
    for0(i, n){
        sum += h[i];
        lsa[i] = sum / (i+1);
    }

    for0(i, n){
        upd(h[i], 1, kl);
        upd(h[i], h[i], su);

        int gr = lsa[i] + 1 + eps;
        int k = get(gr, N-1, kl);
```

```
71          double sb = get(gr, N-1, su);
72
73          ld[i] = sb - k * lsa[i];
74      }
75
76      kl.resize(0);
77      su.resize(0);
78      kl.resize(2*N, 0);
79      su.resize(2*N, 0);
80      sum = 0;
81      for(int i = n-1; i >= 0; i--){
82          sum += h[i];
83          rsa[i] = sum / (n - i);
84      }
85
86      for(int i = n-1; i >= 0; i--){
87          upd(h[i], 1, kl);
88          upd(h[i], h[i], su);
89
90          int gr = rsa[i] + 1 + eps;
91          int k = get(gr, N-1, kl);
92          double sb = get(gr, N-1, su);
93
94          rd[i] = sb - k * rsa[i];
95      }
96
97      double mn = 1e18;
98      int a, b;
99      for(int i = 1; i < n; i++)
100         if(ld[i-1] + rd[i] < mn) {
101             mn = ld[i-1] + rd[i];
102             a = i;
103             b = n - i;
104         }
105     cout<<a<<' '<<b;
106 }
```

Задача I.1.3.5. Прогулка по мостам 2 (20 баллов)

Возможно вы знаете историю про то, как Эйлер гулял по мостам Кенигсберга. Допустим теперь, что Эйлер попал на некий архипелаг, между некоторыми островами которого имеются мосты. Мосты построены таким образом, что между любыми двумя островами архипелага можно построить путь по этим мостам, причем этот путь единственный. Все острова пронумерованы от 1 до n. Эйлер находится на острове номер 1 и желает совершить следующую прогулку: с острова номер 1 он хочет дойти до острова номер 2, далее до острова номер 3 и так далее до острова номер n. С него он хочет вернуться на остров номер 1. Он прекрасно понимает, что при этом ему придется посещать некоторые мосты по многу раз. У каждого моста есть смотритель. Одного из них заинтересовало сколько раз по его мосту пройдет Эйлер.

Формат входных данных

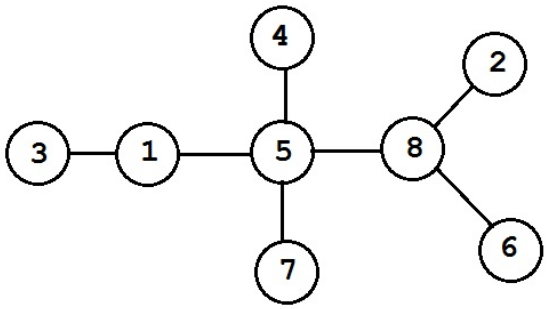
В первой строке содержится два числа через пробел: n — количество островов в архипелаге, $2 \leqslant n \leqslant 10^5$ и b — номер моста, для которого нужно узнать, сколько раз по нему проходит указанный маршрут.В следующих $n - 1$ строках содер-

жатся по два числа a_i и b_i через пробел — номера островов, соединенных мостом, $1 \leqslant a_i, b_i \leqslant n, a_i \neq b_i$. Мосты нумеруются в порядке появления во входных данных, начиная с 1. Гарантируется, что между любыми двумя островами архипелага можно построить путь по этим мостам, причем этот путь единственный.

Формат выходных данных

Вывести одно число — ответ на задачу.

Пояснения к примеру



На рисунке вы видите расположение мостов в архипелаге из примера. Прогулка Эйлера будет проходить следующим образом:

1 — 5 — 8 — **2** — 8 — 5 — 1 — **3** — 1 — 5 — **4** — **5** — 8 — **6** — 8 — 5 — **7** — 5 — **8** — 5 — **1**.

Интересующий нас мост номер 1 соединяет острова 5 и 8. Маршрут проходит по нему 6 раз.

Примеры

Пример №1

Стандартный ввод
8 1 5 8 1 3 8 6 7 5 2 8 1 5 4 5
Стандартный вывод
6

Пример программы-решения

Ниже представлено решение на языке C++.

```cpp
//dfs
#include<bits/stdc++.h>

#define pb push_back
#define mp make_pair
#define x first
#define y second

#define all(x) (x).begin(), (x).end()
#define sz(a) (int)(a).size()
#define for0(i, n) for (int i = 0; i < (int)(n); i++)
#define for1(i, n) for (int i = 1; i <= (int)(n); i++)
#define int long long

using namespace std;
typedef long double ld;
typedef pair<int, int> pii;

int n, z, a, b, q1, q2;
vector<vector<int> > G;
vector<set<int> > rz(2);

void dfs(int a, int p, int dol){
    rz[dol].insert(a);
    for(auto to : G[a]) if(to != p) dfs(to, a, dol);
}

signed main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    cin >> n >> z;
    z--;
    G.resize(n);
    for0(i, n-1){
        cin >> a >> b;
        a--;
        b--;
       if(i != z) {
        G[a].pb(b);
        G[b].pb(a);
       }
       else{
        q1 = a;
        q2 = b;
       }
    }

    dfs(q1, q1, 0);
    dfs(q2, q2, 1);

    vector<int> rzb;
    for(auto q : rz[0]) rzb.pb(q);

    int ans = 0;
    for0(i, sz(rzb)-1){
        if(rzb[i]+1 != rzb[i+1]) ans++;
    }
    if(!(rzb.back() == n-1 && rzb[0] == 0)) ans++;
    cout<<2*ans;
}
```

Пример программы-решения

Ниже представлено решение на языке C++.

```cpp
//обход в ширину, без глубокой рекурсии
#include<bits/stdc++.h>

#define pb push_back
#define mp make_pair
#define x first
#define y second

#define all(x) (x).begin(), (x).end()
#define sz(a) (int)(a).size()
#define for0(i, n) for (int i = 0; i < (int)(n); i++)
#define for1(i, n) for (int i = 1; i <= (int)(n); i++)
#define int long long

using namespace std;
typedef long double ld;
typedef pair<int, int> pii;

int n, z, a, b, q1, q2;
vector<vector<int> > G;
vector<int> rz;
int mn = 1e9, aa, bb;

signed main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    cin >> n >> z;
    z--;

    G.resize(n+1);
    for0(i, n-1){
        cin >> a >> b;

        if(i != z){
        G[a].pb(b);
        G[b].pb(a);
        }
        else{
            q1 = a;
            q2 = b;
        }
    }

    vector<int> och;
    vector<int> mark(n+1, -1);
    och.pb(q1);
    mark[q1] = q1;
    int b = 0;

    while(b < sz(och)){
            int v = och[b];
        for(auto to : G[v]) if(mark[to] == -1){
            och.pb(to);
            mark[to] = v;
        }
        b++;
```

```
    }

    sort(all(och));

       int ans = 0;
    for0(i, sz(och)-1){
        if(och[i]+1 != och[i+1]) ans++;
    }
    if(!(och.back() == n && och[0] == 1)) ans++;

    cout<<2*ans;
}
```

Четвертая попытка. Задачи 8–11 класса

Задача I.1.4.1. Скобки (10 баллов)

Задана строка, в которой могут быть встречены 3 типа скобок: фигурные, квадратные и круглые. Помимо скобок в строке встречаются и другие последовательности символов. Вложенность скобок может быть произвольной. Необходимо проверить корректность скобочной записи: каждой открывающей скобке должна соответствовать следующая за ней закрывающая скобка того же типа на том же уровне вложенности, не должно быть открывающей или закрывающей скобки без пары.

Формат входных данных

Строка, содержащая произвольный набор символов (в т. ч. и без скобок).

Формат выходных данных

- Слово «correct», если запись корректна или не содержит скобок.
- Слово «incorrect», если запись не корректна.

Примеры

Пример №1

Стандартный ввод
(this [is] test)
Стандартный вывод
correct

Пример программы-решения

Ниже представлено решение на языке Python3.

```
def balancedBrackets(expr):
    stack = []
    for char in expr:
        if char in ['(','{','[']:
```

```
            stack.append(char)
        elif char in [')','}', ']']:
            if(len(stack) == 0):
                return False
            top = stack.pop()
            if(top == '(' and char != ')'):
                return ""
            if(top == '{' and char != '}' ):
                return False
            if(top == '[' and char != ']'):
                return False
        else:
            continue
    if(len(stack) > 0):
        return False
    return True

expr = input()
print("correct") if balancedBrackets(expr) else print("incorrect")
```

Задача I.1.4.2. Рисунок (15 баллов)

Задан рисунок, состоящий из пронумерованных точек и линий между ними. Напишите программу, которая скажет, можно ли нарисовать этот рисунок (провести ручку по всем точкам и линиям), не отрывая руки и при этом не проводя одну линию дважды?

Формат входных данных

Первая строка содержит число N — количество точек (число от 0 до 1000 включительно).

Вторая строка содержит число M — количество линий (число от 0 до 1000 включительно).

Далее идет M строк, каждая из которых содержит пары номеров точек, соединенных линиями (нумерация точек с 1, между парой точек может быть проведено несколько линий).

Формат выходных данных

- «Yes», если рисунок можно нарисовать в соответствии с условием.
- «No», если нет.

Примеры

Пример №1

Стандартный ввод
3 2 1 2 2 3

Стандартный вывод
Yes

Пример программы-решения

Ниже представлено решение на языке Python3.

```
def task1():
    vert_num = int(input())
    edge_num = int(input())
    verts = [{"watched": False} for _ in range(vert_num)]
    edges = []
    for _ in range(edge_num):
        edges.append(tuple(int(i)-1 for i in input().split()))

    if vert_num == 0:
        return 'Yes'
    line = [0, 0]
    verts[0]['watched'] = True
    flag = True
    while flag:
        flag = False
        for edge in edges:
            if edge[0] in line or edge[1] in line:
                if edge[0] in line:
                    verts[edge[1]]['watched'] = True
                    if edge[0] == line[0]:
                        line[0] = edge[1]
                    else:
                        line[1] = edge[1]
                else:
                    verts[edge[0]]['watched'] = True
                    if edge[1] == line[0]:
                        line[0] = edge[0]
                    else:
                        line[1] = edge[0]
                edges.remove(edge)
                flag = True
    for v in verts:
        if not v['watched']:
            return 'No'
    if len(edges):
        return 'No'
    return 'Yes'

print(task1())
```

Задача I.1.4.3. Часовой механизм (20 баллов)

К вам обратился владелец часовой мастерской, которая делает часы с прозрачным корпусом. В мастерской есть одна проблема: дизайнеры придумывают то, как будут выглядеть часы, но не задумываются о том, как шестеренки будут крутиться. Поэтому, когда дизайн получают мастера, им приходится проверять работоспособность нарисованного дизайнерами механизма — проверять не заклинит ли механизм от сцепки двух крутящихся в одном направлении шестеренок.

Напишите программу, которая будет делать эту работу за мастеров.

Формат входных данных

Первая строка содержит количество шестеренок N (число от 1 до 1000 включительно).

Вторая строка содержит количество связей между шестеренками M (число от 0 до 1000 включительно), которые сцеплены между собой (если одна из них крутится по часовой стрелке, вторая должна крутиться против часовой и наоборот).

Далее идет M строк с парами чисел, являющихся номерами сцепленных между собой шестеренок (нумерация начинается с 1).

Формат выходных данных

Программа должна вывести одно из 2 слов: «good», если механизм работоспособен или «bad», если механизм заклинит.

Примеры

Пример №1

Стандартный ввод
4 4 1 2 2 3 3 4 4 1

Стандартный вывод
good

Пример программы-решения

Ниже представлено решение на языке Python3.

```
def get_unwatched(verts):
    i = 0
    for vert in verts:
        if not vert['watched']:
            vert['orient'] = 1
            return vert

```

```
def task1():
    vert_num = int(input())
    edges = int(input())
    verts = [{"edges": [], "orient": 0, "watched": False} for _ in range(vert_num)]
    for _ in range(edges):
        a, b = (int(i)-1 for i in input().split())
        verts[a]["edges"].append(b)
        verts[b]["edges"].append(a)

    stack = []
    while vert_num > 0:
        if len(stack) == 0:
            stack.append(get_unwatched(verts))
        vert = stack.pop()
        vert_num -= 1
        vert["watched"] = True
        for next_v in vert['edges']:
            if not verts[next_v]["watched"] and verts[next_v] not in stack:
                stack.append(verts[next_v])
            if verts[next_v]["orient"] != 0 and verts[next_v]["orient"] +
            ↪  vert['orient'] != 3:
                return 'bad'
            elif verts[next_v]["orient"] == 0:
                verts[next_v]["orient"] = 3 - vert['orient']
    return 'good'

print(task1())
```

Задача I.1.4.4. Парковка (20 баллов)

На парковке у бизнес-центра N контрольно-пропускных пунктов(КПП). Известно, что каждый день через каждый день работающие в центре сотрудники въезжают на парковку M раз. Каждый раз сотрудник въезжает в определенное время через один заранее известный КПП и выезжает в определенное время через другой (не обязательно совпадающий с первым). На въезде сотрудник получает пропуск, который он должен отдать на выезде. Один и тот же пропуск может быть использован разными сотрудниками, если они находятся на территории парковки в разное время.

Определите минимальное количество пропусков, которое должно быть суммарно на всех КПП к началу дня, чтобы хватило всем сотрудникам.

Если время въезда и выезда разных сотрудников на одном КПП совпадает, считать, что они могут воспользоваться одним пропуском

Формат входных данных

Первая строка содержит число N (число от 0 до 1000 включительно).

Вторая строка содержит число M (число от 0 до 1000 включительно).

Далее следует M строк, содержащих следующую информацию:

Время въезда (число от 6 до 22 включительно).

Время выезда (число от 7 до 23 включительно).

Номер КПП при въезде (нумерация с 1).

Номер КПП при выезде (нумерация с 1).

Формат выходных данных

Программа должна вывести количество пропусков необходимое суммарно на всех КПП к началу дня.

Примеры

Пример №1

Стандартный ввод
2 2 6 12 1 2 12 22 2 2

Стандартный вывод
1

Пример программы-решения

Ниже представлено решение на языке Python3.

```
def solve():
    required = 0
    kpps_count = int(input())
    kpps = [0] * kpps_count
    n_notes = int(input())
    visitors = []
    for _ in range(n_notes):
        entry_hour, out_hour,in_kpp,out_kpp = [int(readed) for readed in
        ↪  input().split()]
        visitors.append((entry_hour,in_kpp - 1,True))
        visitors.append((out_hour,out_kpp - 1,False))

    visitors.sort(key = lambda visitor: (visitor[0],visitor[2]))

    for visitor in visitors:
        if visitor[2]:
            if not kpps[visitor[1]]:
                required += 1
            else:
                kpps[visitor[1]] -= 1
        else:
            kpps[visitor[1]] += 1
    print(required)

solve()
```

Задача I.1.4.5. Обучаем Терминатора (35 баллов)

Перед отправкой Терминатора T-800 в прошлое для спасения Джона Коннора (события 2 части) обнаружилось, что при анализе текстовых документов, OCR-модуль машины допускает ошибки при чтении символов в записи моделей терми-

наторов. Времени на повторное обучение нейронной сети нет, поэтому было принято решение написать hot-fix на символы «T», «0», «1», «8» и «-». При чтении Терминатор каждый символ переводит в матрицу 10 на 10 точек, где 1 означает наличие заполнения, а 0 — отсутствие.

Символы распознаются следующим образом:

- «T» — два прямоугольника лежащих друг на друге, левая граница верхнего прямоугольника левее нижнего, правая граница верхнего прямоугольника правее нижнего.
- «0» — заполненный прямоугольник с прямоугольным вырезом внутри, границы выреза не лежат на сторонах внешнего прямоугольника.
- «8» — заполненный прямоугольник с двумя прямоугольными вырезами внутри, границы вырезов не лежат на сторонах внешнего прямоугольника, границы вырезов не пересекаются, нижняя граница одного выреза выше другого.
- «1» — заполненный прямоугольник, ширина прямоугольника строго меньше его длины.
- «-» — заполненный прямоугольник, ширина прямоугольника строго больше его длины.

Необходимо, чтоб остальные комбинации интерпретировались символом «X».

Напишите программу для решения поставленной задачи.

Формат входных данных

10 строк состоящих из 10 символов «0» или «1».

Формат выходных данных

Один из символов «T», «0», «1», «8», «-» или «X».

Примеры

Пример №1

Стандартный ввод
0000000000 0001110000 0001010000 0001010000 0001110000 0001110000 0001010000 0001010000 0001110000 0000000000

Стандартный вывод
8

Пример программы-решения

Ниже представлено решение на языке Python3.

```python
def get_borders(l, sym=1):
    a, b = None, None
    for i in range(len(l)):
        if sym in l[i]:
            a = i
            break
    for i in range(len(l)-1, -1, -1):
        if sym in l[i]:
            b = i
            break
    return a, b

def get_lr(l, sym=1):
    left, right = None, None
    for i in range(len(l)):
        if l[i] == sym:
            left = i
            break
    for i in range(len(l)-1, -1, -1):
        if l[i] == sym:
            right = i
            break
    return left, right

def get_rectangle(l, b):
    result = []
    for i in range(b[0], b[2]+1):
        tmp = []
        for j in range(b[1], b[3]+1):
            tmp.append(l[i][j])
        result.append(tmp)
    return result

def is_rectangle(l, sym=1):
    up, bottom = get_borders(l, sym)
    if up is None:
        return None
    left, right = get_lr(l[up], sym)
    for i in range(up, bottom+1):
        if (left, right) != get_lr(l[i], sym):
            return None
    return (up, left, bottom, right)

def task5():
    rows = []
    for _ in range(10):
        rows.append([int(i) for i in input()])
    a = is_rectangle(rows)
    if a:
        tmp = get_rectangle(rows, a)
        if is_rectangle(tmp, 0):
            return '0'
        for i in range(len(tmp)):
            if is_rectangle(tmp[:i], 0) and is_rectangle(tmp[i:], 0):
                return '8'
        if a[2]-a[0] > a[3]-a[1]:
```

```
            return '1'
        if a[2]-a[0] < a[3]-a[1]:
            return '-'
    else:
        up, bottom = get_borders(rows)
        for i in range(up, bottom+1):
            a = is_rectangle(rows[:i])
            b = is_rectangle(rows[i:])
            if a and b and a[1] < b[1] and a[3] > b[3]:
                return 'T'
    return 'X'

print(task5())
```

Задачи первого этапа. Физика

Первая попытка. Задачи 8–9 класса

Задача I.2.1.1. (20 баллов)

Серфинг тренажер в аквапарке представляет собой наклонную плоскость, вверх вдоль которой насосы прокачивают поток воды. Найти подъемную силу, которая действует поперечно со стороны воды на доску с человеком, если он неподвижен относительно плоскости. Угол наклона плоскости $\alpha = 30^\circ$, масса человека $m = 100$ кг, ускорение свободного падения $g = 10$ м/с^2. Ответ дать в ньютонах с точностью до целых.

Решение

Равенство проекций сил вдоль нормали к наклонной плоскости:

$$N = mgCos\alpha = 866\ H$$

Ответ: 866 ± 1.

Задача I.2.1.2. (20 баллов)

Серфинг тренажер в аквапарке представляет собой наклонную плоскость, вверх вдоль которой насосы прокачивают поток воды. Найти скорость течения воды V, если человек на доске неподвижен относительно плоскости. Считать, что на доску со стороны воды действует сила лобового сопротивления $F = C\frac{\rho V^2}{2}S$, где C — коэффициент, ρ — плотность воды, S — площадь поперечного сечения доски. Угол наклона плоскости $\alpha = 30^\circ$, масса человека $m = 100$ кг, ускорение свободного падения $g = 10$ м/с^2, $C = 0,5$, $\rho = 10^3$ кг/м^3, $S = 200$ см^2. Ответ дать в метрах в секунду с точностью до целых.

Решение

Равенство проекций сил вдоль наклонной плоскости:

$$mgSin\alpha = F = C\frac{\rho V^2}{2}S$$

Отсюда находим скорость течения воды:

$$V = \sqrt{\frac{2mgSin\alpha}{\rho CS}} = 10 \text{ м/с}$$

Ответ: 10.

Задача I.2.1.3. (20 баллов)

Серфинг тренажер в аквапарке представляет собой наклонную плоскость, вверх вдоль которой насосы прокачивают поток воды. Полный расход воды равен $Q = 1000$ литров в секунду. На высоте $h = 3$ м скорость течения воды $V = 10$ м/с. $\rho = 10^3$ кг/м^3 — плотность воды, ускорение свободного падения $g = 10$ м/с^2. Мощность одного насоса $P_1 = 10$ кВт, а его КПД $\nu = 0.8$. Сколько нужно поставить насосов у основания наклонной плоскости? Ответ дать в штуках с точностью до целого.

Решение

Закон сохранения энергии:

$$N\eta P_1 = \rho Q(\frac{V^2}{2} + gh)$$

где N — число насосов.

$$N_1 = \frac{\eta Q(V^2 2 + gh)}{\eta P_1} = 10 \text{ штук}$$

Ответ: 10.

Задача I.2.1.4. (20 баллов)

С помощью электродвигателя в колодце на ворот наматывается трос с ведром воды. Длина троса $l = 10$ м и его масса $m = 5,0$ кг, а масса ведра с водой $M = 12$ кг. Ведро поднимается со скоростью $V = 0,5$ м/с. Постоянное напряжение на двигателе $U = 200$ В. Какова максимальная сила тока I_1, протекающего через обмотку двигателя с нулевым сопротивлением? Радиусом ворота пренебречь. Ускорение свободного падения $g = 10$ м/с^2. Ответ дать в амперах с точностью до сотых долей.

Решение

Максимальная мощность двигателя равна максимальной механической мощности:

$$UI_1 = gV(m + M)$$

Отсюда максимальный ток равен:

$$I_1 = \frac{gV(m + M)}{U} = 0,43 \text{ А}$$

Ответ: 0.43 ± 0.01.

Задача I.2.1.5. (20 баллов)

С помощью электродвигателя в колодце на ворот наматывается трос с ведром воды. Длина троса $l = 10$ м и его масса $m = 5,0$ кг, а масса ведра с водой $M = 12$ кг. Ведро поднимается со скоростью $V = 0,5$ м/с. Постоянное напряжение на двигателе $U = 200$ В. Какова максимальная сила тока I_2, протекающего через обмотку двигателя с сопротивлением $R = 20$ Ом? Радиусом ворота пренебречь. Ускорение свободного падения $g = 10$ м/с^2. Ответ дать в амперах с точностью до сотых долей.

Решение

Если учесть сопротивление обмотки, то максимальная мощность двигателя равна максимальной механической мощности плюс небольшая мощность тепловых потерь в обмотке:

$$UI_2 = gV(m + M) + I_2^2 R$$

Отсюда максимальный ток примерно равен:

$$I_2 \approx I_1 + \frac{I_1^2 R}{U} = 0,45 \text{ А}$$

Ответ: 0,45 ± 0,01.

Первая попытка. Задачи 10–11 класса

Задача I.2.2.1. (20 баллов)

Однородный стержень массой $m = 2,0$ кг и длиной $l = 2,0$ м покоится на горизонтальной ледяной поверхности. В центр стержня попадает шайба и передает ему импульс $\Delta p = 0,3$ кг·м/с за время $\Delta t = 40$ мс. Найти ускорение центра стержня после удара. Ответ дать в метрах на секунду в квадрате с точностью до десятых долей.

Решение

Ускорение центра стержня после удара: $a_{\text{ц}} = \frac{\Delta p}{m\Delta t} = 3.8$ м/с^2

Ответ: 3,8 ± 0,1.

Задача I.2.2.2. (20 баллов)

Однородный стержень массой $m = 2,0$ кг и длиной $l = 2,0$ м покоится на горизонтальной ледяной поверхности. В центр всего стержня и в центр половины стержня с противоположных сторон одновременно попадают две шайбы, причем каждая шайба передает ему импульс $\Delta p = 0,3$ кг·м/с за время $\Delta t = 40$ мс. Найти ускорение конца стержня после удара, если стержень начинает двигаться, вращаясь с угловой скоростью $\omega = 3,0$ с$^{-1}$. Ответ дать в метрах на секунду в квадрате с точностью до десятых долей.

Решение

Центр стержня после удара неподвижен. Движение сводится к вращению вокруг центра стержня.

Ускорение конца стержня после удара $a_K = \omega^2 \frac{l}{2} = 9,0$ м/с^2.

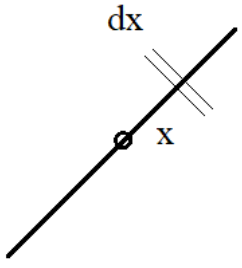
Ответ: 9,0.

Задача I.2.2.3. (20 баллов)

Однородный стержень массой $m = 2,0$ кг и длиной $l = 2,0$ м покоится на горизонтальной ледяной поверхности. В центр всего стержня и в центр половины стержня с противоположных сторон одновременно попадают две шайбы, причем каждая шайба передает ему импульс $\Delta p = 0,3$ кг·м/с за время $\Delta t = 40$ мс. До ударов шайбы двигались перпендикулярно стержню. Найти силу натяжения в центре половины стержня после удара, если стержень начинает двигаться, вращаясь с угловой скоростью $\omega = 3,0$ с$^{-1}$. Ответ дать в ньютонах с точностью до десятых долей.

Решение

Центр стержня после удара неподвижен. Движение сводится к вращению вокруг центра стержня.



Пусть x — координата интересующей нас точки, отсчитываемая от центра стержня. Разность сил натяжения стержня на маленьком отрезке длиной Δx и массой Δm уравновешена центробежной силой, т. е.:

$$\Delta T = \omega^2 x \Delta m = \omega^2 x \frac{m}{l} \Delta x$$

Тогда силу натяжения в точке X можно найти суммированием сил ΔT на участке стержня от X до $\frac{l}{2}$:

$$T(x) = \sum \omega^2 \frac{m}{l} x \Delta x = \omega^2 \frac{m}{l} \sum x \Delta x$$

Нас интересует сила натяжения в точке $x = \frac{l}{4}$:

$$T\frac{l}{4} = \omega^2 \frac{m}{l} \sum_{\frac{l}{4}}^{\frac{l}{2}} x \Delta x$$

Сумма $\sum_{\frac{l}{4}}^{\frac{l}{2}} = \frac{3}{32} l^2$ равна площади трапеции с основаниями $\frac{l}{2}$ и $\frac{l}{4}$ и высотой $\frac{l}{4}$. Тогда сила натяжения в центре половины стержня после удара:

$$T\left(\frac{l}{4}\right) = \frac{3}{32} m\omega^2 l = 3,4 \text{ Н}$$

Ответ: 3,4 ± 0,1.

Задача I.2.2.4. (20 баллов)

Незнайка на ракете НИП-2 прилетел на небольшую планету радиуса $R = 5,0$ км. Привязав к нити длиной $l = 5,0$ см маленький камень массой $m = 0,1$ кг, Незнайка соорудил маятник и измерил период его малых колебаний в разных точках поверхности планеты. Во всех точках период получился одинаковым $T = 20$ с. Чему равна средняя плотность планеты? Гравитационная постоянная равна $G = 6,7 \cdot 10^{-11}$ Н·м^2·кг$^{-2}$. Ответ дать в тоннах на кубический метр с точностью до десятых долей.

Решение

Период колебаний математического маятника:

$$T = 2\pi\sqrt{\frac{l}{g}}$$

Ускорение свободного падения на поверхности планеты:

$$g = \frac{GM}{R^2}$$

где M — масса планеты.

Средняя плотность планеты равна:

$$\rho = \frac{M}{\frac{4}{3}\pi R^3} = \frac{3\pi l}{GRT^2} = 3,5 \cdot 10^3 \text{ кг/м}^3$$

Ответ: 3,5 ± 0,1.

Задача I.2.2.5. (20 баллов)

Затем Незнайка нашел шахту, ведущую к центру планеты. Он измерил период малых колебаний маятника на различных глубинах в шахте и получил одинаковый результат $T = 20$ с. Найти плотность планеты $\rho(R/2)$ на расстоянии $(R/2)$ до ее центра. Гравитационная постоянная равна $G = 6,7 \cdot 10^{-11}$ Н·м^2·кг$^{-2}$. Ответ дать в тоннах на кубический метр с точностью до десятых долей.

Решение

Период колебаний математического маятника:

$$T = 2\pi\sqrt{\frac{l}{g}}$$

Ускорение свободного падения на расстоянии r от центра планеты:

$$g = \frac{GM}{R^2}$$

где m — масса шара радиуса r. Так как период одинаковый, то ускорение свободного падения от радиуса r не зависит и масса m равна:

$$m(r) = \frac{g}{G}r^2$$

Масса шарового слоя радиуса r и малой толщиной Δr равна:

$$\Delta m = m(r + \Delta r) - m(r) \approx \frac{g}{G}2r\Delta r,$$

а его объем равен $\Delta V = 4\pi r^2 \Delta r$.

Тогда плотность планеты на расстоянии r:

$$\rho(r) = \frac{\Delta m}{\Delta V} = \frac{2\pi l}{GT^2 r}$$

Если $r = \frac{R}{2}$, то:

$$\rho(\frac{R}{2}) = \frac{4\pi l}{GT^2 R} = 4.7 \cdot 10^3 \text{ кг/м}^3$$

Ответ: 4,7 ± 0,1.

Вторая попытка. Задачи 8–9 класса

Задача I.2.3.1. Северный ветер 1 (20 баллов)

Существует проект создания ветроэлектростанции мощностью $P = 10$ МВт (мегаватт) за полярным кругом на побережье Северного Ледовитого океана для перекачки электроэнергии в Китай на расстояние $l = 5000$ км. Рассмотрим работу отдельного обычного ветрогенератора с тремя крыльями, закрепленными на тонкой горизонтальной оси. Он преобразует энергию ветра в электрическую энергию, накапливаемую в аккумуляторе. Найти мощность (кинетическая энергия в единицу времени) воздушного потока, проходящего через вращающиеся крылья, если длина крыла $r = 6,0$ м, плотность воздуха $\rho = 1,2$ кг/м^3, средняя скорость ветра $V = 5,0$ м/с. Ответ дать в кВт и округлить до десятых долей.

Решение

Кинетическая энергия в единицу времени воздушного потока, проходящего через вращающиеся крылья равна:

$$P_1 = \frac{\Delta m V^2}{2\Delta t}$$

где $\Delta m = \rho S \nu \Delta t$ — масса воздуха проходящего через площадь $S = \pi r^2$ за время Δt. Отсюда мощность воздушного потока равна:

$$P_1 = \frac{\rho \pi r^2 V^3}{2} = 8,5 \text{ кВт}$$

Точность 0,1 кВт.

Ответ: 8,5 ± 0,1.

Задача I.2.3.2. Северный ветер 2 (20 баллов)

Продолжение задачи «Северный ветер 1».

КПД η отдельного ветрогенератора это отношение его электрической мощности к мощности воздушного потока. Обычно $\eta = 0,35$. Найти постоянный ток зарядки I аккумулятора ветрогенератора с напряжением $U = 24$ В. Ответ дать в А с точностью до целых.

Решение

Из определения КПД и электрической мощности находим

$$I = \frac{\eta P_1}{U} = 124 \text{ А}$$

Точность 5 А.

Ответ: 124 ± 5.

Задача I.2.3.3. Северный ветер 3 (20 баллов)

Продолжение задач «Северный ветер 1–2».

КПД η отдельного ветрогенератора это отношение его электрической мощности к мощности воздушного потока. Обычно $\eta = 0,35$. Сколько нужно поставить отдельных последовательно соединенных ветрогенераторов на электростанции мощностью $P = 10$ МВт? Ответ в штуках с точностью до сотен.

Решение

Мощности складываются. Поэтому:

$$N = \frac{P}{\eta P_1} = 3,4 \text{ тыс. шт.}$$

Точность 0,1 тыс. шт.

Ответ: 3400 ± 100.

Задача I.2.3.4. Северный ветер 4 (20 баллов)

Продолжение задач «Северный ветер 1–3».

Кабель для перекачки электроэнергии в Китай имеет длину $l = 5000$ км и сопротивление $R = 100$ Ом. Найти массу этого кабеля, если его изготовить из алюминия с удельным сопротивлением $\rho = 0,028$ Ом·мм^2/м при температуре $t = 20$°C. Плотность алюминия $\rho_{\text{пл}} = 2700$ кг/м^3. Ответ дать в тыс. тонн и округлить до целых.

Решение

Сопротивление кабеля:

$$R = \frac{\rho l}{S},$$

где S — площадь сечения кабеля.

Масса кабеля равна::

$$m = \rho_{\text{пл}} l S = \frac{\rho_{\text{пл}} \rho l^2}{R} = 19 \text{ тыс. тонн}$$

Точность 1 тыс. тонн.

Ответ: 19 ± 1.

Задача I.2.3.5. Северный ветер 5 (20 баллов)

Продолжение задач «Северный ветер 1–4».

Кабель для перекачки электроэнергии в Китай имеет длину $l = 5000$ км и радиус поперечного сечения $r = 21$ мм. Найти сопротивление R этого кабеля, если его

изготовить из алюминия. Удельное сопротивление алюминия увеличивается по линейному закону вдоль кабеля от значения $\rho_1 = 0,020$ Ом $\cdot$ мм^2/м при температуре $t_1 = -40°C$ на побережье Северного Ледовитого океана до значения $\rho_2 = 0,025$ Ом$\cdot$ $\cdot$мм^2/м при температуре $t_2 = 0°$C в Китае. Ответ дать в Ом с точностью до целых.

Решение

Так как удельное сопротивление алюминия увеличивается по линейному закону, то:

$$R = \frac{\rho_{\text{среднее}} l}{S} = \frac{(\rho_1 + \rho_2) l}{2\pi r^2} = 81 \text{ Ом}$$

Точность 1 Ом.

Ответ: 81 $\pm$ 1.

Вторая попытка. Задачи 10–11 класса

Задача I.2.4.1. Знайка на воздушном шаре (20 баллов)

Знайка наполнил с помощью насоса через трубку пустой воздушный шар горячим воздухом с температурой $T = 400$ К и завязал трубку, чтобы воздух не выходил из шара. Конечный объем шара $V = 2,0$ м^3, давление воздуха внутри шара равно атмосферному давлению $p = 1,0 \cdot 10^5$ Па, молярная масса воздуха составляет $M = 29$ г/моль. Универсальная газовая постоянная $R = 8,31$ Дж/(моль $\cdot$ К). Какая масса горячего воздуха прошла через насос? Ответ округлить до десятых долей килограмма.

Решение

Из уравнения Менделеева — Клапейрона получаем массу горячего воздуха:

$$m_B = \frac{pVM}{RT} = 1,7 \text{ кг}$$

Точность 0,1 кг.

Ответ: 1,7 $\pm$ 0,1.

Задача I.2.4.2. Знайка на воздушном шаре. Продолжение 1 (20 баллов)

В условиях предыдущей задачи найти ускорение, с которым начнет подниматься воздушный шар после наполнения горячим воздухом. Температура окружающего воздуха $T_0 = 300$ К, масса оболочки шара и корзины с коротышками равна $m = 0,4$ кг, ускорение свободного падения $g = 10$ м/с^2. Ответ дать в м/с^{2}и округлить до десятых долей.

Решение

Из 2 закона Ньютона:

$$(m_B + m)a = F_A - (m_B + m)g$$

Где сила Архимеда равна:

$$F_A = \frac{pMgV}{RT_0}$$

Тогда:

$$a = g(\frac{m_B T}{(m_B + m)T_0} - 1) = 0,8 \text{ м/с}^2$$

Точность 0,1 м/с^2.

Ответ: 0,8 ± 0,1.

Задача I.2.4.3. Знайка на воздушном шаре. Продолжение 2 (20 баллов)

Используем условия задач 1 и 2. После того как воздух в воздушном шаре остыл до температуры T_1 шар стал опускать вниз с постоянной скоростью $v = 1,0$ м/с. Найти температуру T_1, если коэффициент сопротивления воздуха равен $k = 0,40$ Н·с^2/м^2. Ответ дать в К и округлить до целых.

Решение

Из второго закона Ньютона:

$$0 = F_{Al} + F_{con} - (m_B + m)g,$$

где сила сопротивления воздуха $F_{con} = k\nu^2$, а сила Архимеда равна $F_{Al} = \frac{m_B g T1}{T_0}$. Отсюда:

$$T_1 = T_0 \left(\frac{m_B + m}{m_B} - \frac{k\nu^2}{m_B g} \right) = 362K$$

Точность 5 К.

Ответ: 362 ± 3.

Задача I.2.4.4. Незнайка на вращающейся планете (20 баллов)

Незнайка на ракете НИП-2 прилетел на небольшую планету радиуса $R = 5,0$ км. Привязав к нити длиной $l = 5,0$ см маленький камень массой $m = 0,1$ кг, Незнайка соорудил маятник и измерил период его малых колебаний в разных точках поверхности планеты. Во всех точках период получился одинаковым $T = 20$ с. Затем Незнайка увеличил точность измерений периода колебаний маятника и обнаружил, что на экваторе период на $\Delta T = 0,20$ с больше, чем на полюсе. Определить период обращения планеты вокруг своей оси. Ответ дать в часах с точностью до целого.

Решение

Период колебаний математического маятника на полюсах:

$$T = 2\pi\sqrt{\frac{l}{g}},$$

где g — ускорение свободного падения на поверхности планеты. Ускорение свободного падения на поверхности планеты:

$$g = \frac{4\pi^2 l}{T^2}.$$

Период колебаний математического маятника на экваторе увеличивается под действием центробежной силы:

$$T + \Delta T = 2\pi\sqrt{\frac{l}{g - \omega^2 R}},$$

где ω — угловая скорость вращения планеты.

Отсюда период обращения планеты вокруг своей оси приблизительно равен:

$$T_{\text{об}} = \frac{2\pi}{\omega} = \sqrt{\frac{RT^3}{2l\Delta T}} = 4,47 \cdot 10^4 \text{ с} = 12 \text{ ч}.$$

Точность 1 ч.

Ответ: 12 ± 1.

Задача I.2.4.5. Незнайка на вращающейся планете. Продолжение задачи (20 баллов)

В условиях предыдущей задачи найти превышение веса на нити маленького камня массой $m = 0,1$ кг на полюсе над весом камня на экваторе планеты. Ответ дать в микроньютонах с точностью до целых.

Решение

Период колебаний математического маятника на полюсах:

$$T = 2\pi\sqrt{\frac{l}{g}},$$

где g — ускорение свободного падения на поверхности планеты.

Ускорение свободного падения на поверхности планеты:

$$g = \frac{4pi^2 l}{T^2}$$

Период колебаний математического маятника на экваторе увеличивается под действием центробежной силы:

$$T + \Delta T = 2\pi\sqrt{\frac{l}{g - \omega^2 R}},$$

где ω — угловая скорость вращения планеты.

Отсюда:

$$\omega^2 R = 2g\frac{\Delta T}{T}$$

Веса камня на полюсе и на экваторе отличаются на значение центробежной силы, действующей на камень на экваторе:

$$\Delta P = m\omega^2 R = 2m\frac{4\pi^2 l}{T^2}\frac{\Delta T}{T} = 10\,\text{мкН}.$$

Точность 1 мкН.

Ответ: 10 ± 1.

Третья попытка. Задачи 8–9 класса

Задача I.2.5.1. Незнайка на ракете (20 баллов)

Незнайка летит на ракете НИП-2 массой $m = 10$ кг в космосе со скоростью $V_1 = 5,0$ м/с. Затем он для увеличения скорости включает электрореактивный двигатель с силой тяги $F = 2 \cdot 10^{-5}$ Н на время $t = 275$ ч. Электрореактивный двигатель выбрасывает назад струю ионов ксенона. Найти конечную скорость ракеты V_2, если за время работы двигателя израсходовано $m_1 = 2,0$ кг ксенона. Другие силы на ракету не действуют. Ответ дать в м/с с точностью до десятых долей.

Решение

Из 2 закона Ньютона получаем:

$$(m - m_1)V_2 - mV_1 = Ft$$

Отсюда:

$$V_2 = \frac{Ft + mV_1}{m - m_1} = 8,7\,\text{м/с}$$

Точность 0,1 м/с.

Ответ: 8,7 ± 0,1.

Задача I.2.5.2. Незнайка на ракете (Продолжение задачи 1) (20 баллов)

Незнайка летит на ракете НИП-2 массой $m = 10$ кг в космосе со скоростью $V_1 = 5,0$ м/с. Затем он для увеличения скорости включает электрореактивный двигатель с силой тяги $F = 2 \cdot 10^{-5}$ Н на время $t = 275$ ч. Электрореактивный двигатель выбрасывает назад струю ионов ксенона. Найти среднее ускорение ракеты a, если за время работы двигателя израсходовано $m_1 = 2,0$ ксенона. Другие силы на ракету не действуют. Ответ дать в мкм/с^2 с точностью до десятых долей.

Решение

Среднее ускорение равно:

$$a = \frac{V_2 - V_1}{t}$$

где конечная скорость равна:

$$V_2 = \frac{Ft + mV_1}{m - m_1}$$

Тогда:

$$a = \frac{F + m_1 V_1 t}{m - m_1} = 3,8 \text{ мкм/с}^2$$

Точность 0,1 мкм/с^2.

Ответ: 3,8 ± 0,1.

Задача I.2.5.3. Незнайка и самовар (20 баллов)

Незнайка первый раз нагревает воду в электросамоваре, подключенном к источнику постоянного напряжения $U = 12$ В. Масса воды равна $m = 20$ г, а ее удельная теплоемкость $c = 4200$ Дж/(кг · С). Начальная температура воды $T_0 = 20$°С. Через какое время t_1 вода закипит? Потерями тепла пренебречь. Незнайка считает, что электрическое сопротивление нагревательного элемента самовара в процессе нагревания не изменяется и равно $R = 10,8$ Ом. Ответ дать в минутах с точностью до десятой доли минуты.

Решение

Закон сохранения энергии:

$$cm(T_K - T_0) = \frac{U^2}{R} t_1$$

где $T_K = 100°C$ — температура кипения воды.

Отсюда

$$t_1 = \frac{R}{U^2} cm(T_K - T_0) = 8,4 \text{ мин}$$

Точность 0,1 мин.

Ответ: 8,4 ± 0,1.

Задача I.2.5.4. Незнайка и самовар (Продолжение задачи 3) (20 баллов)

Незнайка второй раз нагревает воду в электросамоваре, подключенном к источнику постоянного напряжения $U = 12$ В. Масса воды равна $m = 20$ г, а ее удельная теплоемкость $c = 4200$ Дж/(кг $\cdot$ С). Начальная температура воды $T_0 = 20$°C. Через какое время t_2 вода закипит? Потерями тепла пренебречь. Теперь Незнайка учитывает, что электрическое сопротивление нагревательного элемента самовара зависит от температуры T линейно: $R = R_0(1 + \alpha T)$, где $R_0 = 10$ Ом, температурный коэффициент сопротивления $\alpha = 0,004C^{-1}$. Ответ дать в минутах с точностью до десятой доли минуты.

Решение

Закон сохранения энергии за малое время Δt, когда температура немного изменяется от T до $T + \Delta T$:

$$cm\Delta T = \frac{U^2}{R_0(1 + \alpha T)}\Delta t$$

Отсюда:

$$\Delta t = \frac{cmR_0}{U^2}(1 + \alpha T)\Delta T$$

В этом уравнении температура T принимает любые значения в диапазоне от $T_0 = 20^\circ$ до температуры кипения воды $T_K = 100^\circ$.

Складываем все малые времена Δt и получаем в левой части уравнения искомое время t_2, а в правой части возникает сумма площадей узких прямоугольников с переменной высотой $(1 + \alpha T)$ и шириной ΔT:

$$t_2 = \frac{cmR_0}{U^2}\sum_{T_0}^{T_K}(1 + \alpha T)\Delta T$$

Сумма площадей прямоугольников:

$$\sum_{T_0}^{T_K}(1 + \alpha T)\Delta T = \frac{1 + \alpha T_K + 1 + \alpha T_0}{2}(T_K - T_0)$$

равна площади трапеции с основаниями $1 + \alpha T$ и $1 + \alpha T_K$ и высотой $T_K - T_0$. Тогда:

$$t_2 = \frac{R_0}{U^2}cm(1 + \frac{\alpha}{2}(T_K + T_0))(T_K - T_0) = 9,6 \text{ мин}$$

Точность 0,1 мин.

Ответ: 9,6 $\pm$ 0,1.

Задача I.2.5.5. Незнайка и самовар (Продолжение задач 3 и 4) (20 баллов)

Незнайка забыл выключить закипевший электросамовар. Через какое время t_3 после начала кипения вода в нем полностью испарится? Самовар все это время подключен к источнику постоянного напряжения $U = 12$ В. Масса воды равна $m = 20$ г,

а ее удельная теплота парообразования $L = 2,3 \cdot 10^6$ Дж/кг. Потерями тепла пренебречь. Учесть, что электрическое сопротивление нагревательного элемента самовара зависит от температуры T линейно: $R = R_0(1+\alpha T)$, где $R_0 = 10$ Ом, температурный коэффициент сопротивления $\alpha = 0,004\, C^{-1}$. Ответ дать в минутах с точностью до минуты.

Решение

Закон сохранения энергии:

$$mL = \frac{U^2}{R_0(1+\alpha T_K)} t_3$$

где $T_K = 100°C$ — температура кипения воды.

Отсюда:

$$t_3 = \frac{mL}{U^2} R_0(1+\alpha T_K) = 75 \text{ мин}$$

Точность 1 мин.

Ответ: 75 ± 1.

Третья попытка. Задачи 10–11 класса

Задача I.2.6.1. Незнайка улетает (20 баллов)

Незнайка на ракете НИП-2 прилетел на небольшую планету радиуса $R = 5,0$ км. Привязав к нити длиной $l = 5,0$ см маленький камень, Незнайка соорудил маятник и измерил период его малых колебаний в разных точках поверхности планеты. Во всех точках период получился одинаковым $T = 20$ с. Завершив исследования, Незнайка улетел на ракете на круговую орбиту, находящуюся в космосе на небольшой высоте над поверхностью планеты. Определить скорость ракеты V_1 на орбите. Ответ дать в м/с с точностью до десятых долей.

Решение

Период колебаний математического маятника на поверхности:

$$T = 2\pi\sqrt{\frac{l}{g}}$$

где g — ускорение свободного падения на поверхности планеты.

Ускорение свободного падения на поверхности планеты:

$$g = \frac{4\pi^2 l}{T^2}$$

Первая космическая скорость на орбите равна:

$$V_1 = \sqrt{gR} = \frac{2\pi\sqrt{lR}}{T} = 5,0 \text{ м/с}$$

Точность 0,1 м/с.

Ответ: 5,0 ± 0,1.

Задача I.2.6.2. Незнайка улетает (Продолжение задачи 1) (20 баллов)

Используем условия задач 1. Незнайка летает на ракете на круговой орбите, находящейся в космосе на небольшой высоте над поверхностью планеты. Затем он, включив на короткое время орбитальный двигатель, увеличивает скорость ракеты до значения V_2 и улетает на очень большое расстояние от планеты. Там ракета имеет скорость $V_3 = 10$ м/с. Найти скорость V_2. Ответ дать в м/с с точностью до десятых долей.

Решение

Период колебаний математического маятника на поверхности:

$$T = 2\pi\sqrt{\frac{l}{g}}$$

где g — ускорение свободного падения на поверхности планеты.

Ускорение свободного падения на поверхности планеты:

$$g = \frac{4\pi^2 l}{T^2}$$

Закон сохранения энергии для ракеты массой m:

$$\frac{mV_2^2}{2} - mgR = \frac{mV_3^2}{2}$$

Отсюда:

$$V_2 = \sqrt{V_3^2 + \frac{8\pi^2 lR}{T^2}} = 12,2\,\text{м/с}$$

Точность 0,1 м/с.

Ответ: 12,2 ± 0,1.

Задача I.2.6.3. Бамперные машинки (20 баллов)

Незнайка и Пончик катаются на аттракционе «Бамперные машинки». Каждая машинка имеет форму твердого диска радиуса $a = 6,0$ см, окруженного упругим резиновым кольцом (бампером) шириной $b = 1,0$ см. Масса машины с Незнайкой $m_1 = 450$ г, а масса машины с Пончиком $m_2 = 475$ г. Найти расстояние от центра машинки с Незнайкой до общего центра масс двух машинок в момент времени, когда машинки касаются друг друга бамперами. Считать, что центр масс каждой машинки с водителем совпадает с центром диска. Ответ дать в см и округлить до десятых долей.

Решение

Центр масс двух машинок находится на расстоянии:

$$x = \frac{2m_2(a+b)}{m_1+m_2} = 7,2 \text{ см от центра машинки с Незнайкой.}$$

Точность 0,1 см.

Ответ: 7,2 ± 0,1.

Задача I.2.6.4. Бамперные машинки (Продолжение задачи 3) (20 баллов)

Используем условия задачи 3. Незнайка и Пончик катаются на аттракционе «Бамперные машинки». Каждая машинка имеет форму твердого диска радиуса $a = 6,0$ см, окруженного упругим резиновым кольцом шириной $b = 1,0$ см. Масса машины с Незнайкой $m_1 = 450$ г, а масса машины с Пончиком $m_2 = 475$ г. Происходит столкновение машинок. Найти модули скоростей машинок V_1 и V_2 сразу после центрального абсолютно упругого соударения, если их скорости перед ударом равны $V_{01} = 20$ см/с и $V_{02} = 25$ см/с соответственно. Силами трения пренебречь. Ответ дать в см/с и округлить до целых.

Решение

После центрального абсолютно упругого соударения машинки меняют направления своих скоростей.

Проектируем закон сохранения импульса на направление скорости $\overrightarrow{V_{01}}$:

$$m_1V_{01} - m_2V_{02} = -m_1V_1 + m_2V_2$$

Закон сохранения энергии:

$$\frac{m_1V_{01}^2}{2} + \frac{m_2V_{02}^2}{2} = \frac{m_1V_1^2}{2} + \frac{m_2V_2^2}{2}$$

Выносим массы и делим второе уравнение на первое. Выражаем:

$$V_2 = V_{01} + V_{02} - V_1$$

Тогда:

$$V_1 = \frac{2m_2V_{02} + (m_2 - m_1)V_{01}}{m_1+m_2} = 26 \text{ см/с}$$

$$V_2 = \frac{2m_1V_{01} + (m_2 - m_1)V_{02}}{m_1+m_2} = 19 \text{ см/с}$$

Точность 1 см/с.

Ответ: $V_1 = 26 \pm 1$; $V_2 = 19 \pm 1$.

Задача I.2.6.5. Бамперные машинки (Продолжение задачи 3 и 4) (20 баллов)

Используем условия задач 3 и 4. Незнайка и Пончик катаются на аттракционе «Бамперные машинки». Каждая машинка имеет форму твердого диска радиуса $a = 6,0$ см, окруженного упругим резиновым кольцом шириной $b = 1,0$ см. Масса машины с Незнайкой $m_1 = 450$ г, а масса машины с Пончиком $m_2 = 475$ г. Происходит столкновение машинок. Найти модуль скорости центра масс машинок $V_{\text{ц}}$ сразу после центрального абсолютно упругого соударения, если скорости машинок перед ударом равны $V_{01} = 20$ см/с и $V_{02} = 25$ см/с соответственно. Силами трения пренебречь. Ответ дать в см/с и округлить до целых.

Решение

В отсутствии внешних горизонтальных сил скорость центра масс остается постоянной:

$$V_{\text{ц}} = \frac{m_2 V_{02} - m_1 V_{01}}{m_2 + m_1} = 3 \text{ см/с}$$

Точность 1 см/с.

Ответ: 3 ± 1.

Четвертая попытка. Задачи 8–9 класса

Задача I.2.7.1. Ветроэлетростанция 1 (20 баллов)

Существует проект создания ветроэлектростанции мощностью $P = 1000$ МВт (мегаватт) в России. Рассмотрим работу отдельного обычного ветрогенератора с тремя крыльями, закрепленными на тонкой горизонтальной оси. Он преобразует энергию ветра в электрическую энергию, накапливаемую в аккумуляторах. Глава «Роснано» Анатолий Чубайс недавно сообщил, что в Ульяновске производятся крылья из углепластика длиной $r = 65$ м. Найти мощность (кинетическая энергия в единицу времени) воздушного потока, проходящего через вращающиеся крылья, если плотность воздуха $\rho = 1,2$ кг/м^3, средняя скорость ветра $V = 10,0$ м/с. Ответ дать в МВт и округлить до десятых долей.

Решение

Кинетическая энергия в единицу времени воздушного потока, проходящего через вращающиеся крылья равна:

$$P_1 = \frac{\Delta m V^2}{2\Delta t}$$

где $\Delta m = \rho S \nu \Delta t$ — масса воздуха проходящего через площадь $S = \pi r^2$ за время Δt. Отсюда мощность воздушного потока равна

$$P_1 = \frac{\rho \pi r^2 V^3}{2} = 8,0 \text{ МВт}$$

Точность 0,1 МВт.

Ответ: 8,0 ± 0,1.

Задача I.2.7.2. Ветроэлетростанция 2 (20 баллов)

Продолжение задачи 1. КПД η отдельного ветрогенератора это отношение его электрической мощности к мощности воздушного потока. Пусть $\eta = 0,33$. Найти силу тока зарядки 1000 последовательно соединенных аккумуляторов. Напряжение каждого аккумулятора $U = 48$ В. Ответ дать в амперах и округлить до целого.

Решение

Из определения КПД и электрической мощности находим:

$$I = \frac{\eta P_1}{1000U} = 55\ A$$

Точность 1 А.

Ответ: 55 ± 1.

Задача I.2.7.3. Ветроэлетростанция 3 (20 баллов)

Продолжение задач 1 и 2. КПД η отдельного ветрогенератора это отношение его электрической мощности к мощности воздушного потока. Пусть $\eta = 0,33$. Сколько нужно поставить отдельных ветрогенераторов, чтобы мощность всей ветроэлектростанции составляла $P = 1000$ МВт? Ответ дать в штуках.

Решение

Мощности складываются. Поэтому:

$$N = \frac{P}{\eta P_1} = 379$$

Точность 5 шт.

Ответ: 379 ± 5.

Задача I.2.7.4. Электросамовар 1 (20 баллов)

В кафе «Самоваръ» нагревают воду в электросамоваре объемом $V = 25$ л, подключенном к источнику постоянного напряжения $U = 300$ В. Плотность воды равна $\rho = 1000$ кг/м^3, а ее удельная теплоемкость $c = 4200$ Дж/(кг $\cdot$ С). Начальная температура воды $T_0 = 18C$. Через какое время t_1 вода закипит? Потери тепла составляют 30%. Среднее электрическое сопротивление нагревательного элемента самовара $R = 12,4$ Ом. Ответ дать в минутах с точностью до целой.

Решение

Закон сохранения энергии:

$$cm(T_K - T_0) = 0,7\frac{U^2}{R}t_1$$

где $T_K = 100°C$ — температура кипения воды.

Масса воды $m = \rho V$. Отсюда:

$$t_1 = \frac{R}{0.7U^2}c\rho V(T_K - T_0) = 28 \text{ мин}$$

Точность 1 мин.

Ответ: 28 ± 1.

Задача I.2.7.5. Электросамовар 2 (20 баллов)

В кафе «Самоваръ» нагревают воду в электросамоваре объемом $V = 25$ л, подключенном к источнику постоянного напряжения $U = 300$ В. По окончании работы забыли выключить закипевший полный электросамовар. Через какое время t_3 после начала кипения вода в нем полностью испарится? Плотность воды равна $\rho = 1000$ кг/м^3, а ее удельная теплота парообразования $L = 2,3 \cdot 10^6$ Дж/кг. Потери тепла составляют 30%. Учесть, что электрическое сопротивление нагревательного элемента самовара зависит от температуры T линейно: $R = R_0(1 + \alpha T)$, где $R_0 = 10$ Ом, температурный коэффициент сопротивления $\alpha = 0,004\,C^{-1}$. Ответ дать в минутах с точностью до целой.

Решение

Закон сохранения энергии:

$$mL = \frac{0.7U^2}{R_0(1 + \alpha T_K)}$$

где $T_K = 100°C$ — температура кипения воды. Отсюда:

$$t_3 = \frac{\rho V L}{0.7U^2}R_0(1 + \alpha T_K) = 213 \text{ мин}$$

Точность 5 мин.

Ответ: 213 ± 5.

Четвертая попытка. Задачи 10–11 класса

Задача I.2.8.1. Воздушный шар с горячим воздухом (20 баллов)

Коротышки решили путешествовать на воздушном шаре. Для этого они наполнили с помощью насоса через трубку пустой воздушный шар горячим воздухом с

температурой $T = 420$ К и закрыли трубку, чтобы воздух не выходил из шара. Конечный объем шара $V = 2,5$ м^3, давление воздуха внутри шара равно атмосферному давлению $p = 1,0 \cdot 10^5$ Па, молярная масса воздуха составляет $M = 29$ г/моль. Универсальная газовая постоянная $R = 8,31$ Дж/(моль $\cdot$ К). Какая масса горячего воздуха прошла через насос? Ответ округлить до десятых долей килограмма.

Решение

Из уравнения Менделеева – Клапейрона получаем массу горячего воздуха.

$$m_B = \frac{pVM}{RT} = 2,1 \text{ кг}$$

Точность 0,1 кг.

Ответ: 2,1 $\pm$ 0,1.

Задача I.2.8.2. Воздушный шар с горячим воздухом. Продолжение 1 (30 баллов)

В условиях задачи 1 найти ускорение, с которым начнет подниматься воздушный шар после отцепления от куста. Температура окружающего воздуха $T_0 = 290$ К, масса оболочки шара и корзины с коротышками равна $m = 0,6$ кг, ускорение свободного падения $g = 10$ м/с^2. Ответ дать в м/с^2 и округлить до десятых долей.

Решение

Из второго закона Ньютона:

$$(m_B + m)a = F_A - (m_B + m)g$$

где сила Архимеда равна:

$$F_A = \frac{pMgV}{RT_0}$$

Тогда:

$$a = g(\frac{m_B T}{(m_B + m)T_0} - 1) = 1,2 \text{ м/с}^2$$

Точность 0,2 м/с^2.

Ответ: 1,2 $\pm$ 0,2.

Задача I.2.8.3. Воздушный шар с горячим воздухом. Продолжение 2 (20 баллов)

Используем условия задач 1 и 2. После того как воздух в воздушном шаре остыл до температуры T_1 шар с коротышками стал опускать вниз с постоянной скоростью $v = 1,5$ м/с. Найти температуру T_1, если коэффициент сопротивления воздуха равен $k = 0,35$ Н$\cdot$с^2/м^2. Ответ дать в кельвинах и округлить до целых.

Решение

Из второго закона Ньютона:

$$0 = F_{Al} + F_{com} - (m_B + m)g$$

где сила сопротивления воздуха $F_{con} = k\nu^2$, а сила Архимеда равна:

$$F_{Al} = \frac{m_B g T_1}{T_0}$$

Отсюда:

$$T_1 = T_0 \left(\frac{m_B + m}{m_B} - \frac{k\nu^2}{m_B g} \right) = 362\ K$$

Точность 3 К.

Ответ: 362 ± 3.

Задача I.2.8.4. Путешественник и экзопланета (20 баллов)

Космический путешественник прилетел на звездолете на экзопланету радиуса $R = 5500$ км. Привязав к нити длиной $l = 50$ см маленький камень массой $m = 1,1$ кг, путешественник получил маятник. Он измерил время 10 полных колебаний маятника в разных точках поверхности экзопланеты. Во всех точках время получилось одинаковым $t = 20$ с. Чему равна средняя плотность планеты? Гравитационная постоянная равна $G = 6,7 \cdot 10^{-11}$ Н·м^2·кг$^{-2}$. Ответ дать в тоннах/м3 с точностью до десятых.

Решение

Период колебаний математического маятника:

$$T = 2\pi\sqrt{\frac{l}{g}} = \frac{t}{10} = 2c$$

Ускорение свободного падения на поверхности планеты:

$$g = \frac{GM}{R^2}$$

где M — масса планеты.

Средняя плотность планеты равна:

$$\rho = \frac{M}{\frac{4}{3}\pi R^3} = \frac{3\pi l}{GRT^2} = 3,2\ \text{т/м}^3$$

Точность 0,1 т/м^3.

Ответ: 3,2 ± 0,1.

Задача I.2.8.5. Путешественник и экзопланета. Продолжение задачи 4 (20 баллов)

Используем условия задачи 4. Завершив измерения, путешественник улетел на звездолете на круговую орбиту, находящуюся в космосе на небольшой высоте над поверхностью экзопланеты. Определить скорость звездолета V_1 на орбите. Ответ дать в км/с с точностью до десятых долей.

Решение

Период колебаний математического маятника на поверхности:

$$T = 2\pi\sqrt{\frac{l}{g}} = \frac{t}{10} = 2\ c$$

где g — ускорение свободного падения на поверхности планеты.

Ускорение свободного падения на поверхности планеты:

$$g = \frac{4\pi^2 l}{T^2}$$

Первая космическая скорость на орбите равна:

$$V_1 = \sqrt{gR} = \frac{2\pi\sqrt{lR}}{T} = 5,2\text{км/с}$$

Точность 0,1 км/с.

Ответ: 5,2 ± 0,1.