ПРОФИЛЬ
«АВТОНОМНЫЕ ТРАНСПОРТНЫЕ СИСТЕМЫ»

Профиль «Автономные транспортные системы» посвящен решению реальной задачи получения, идентификации и доставки грузов в пункт назначения разными типами транспорта с учетом их взаимодействия с объектами инфраструктуры. Участникам данного профиля предстоит решать инженерные задачи по созданию и управлению беспилотным транспортом на суше, море и в воздухе, а также обеспечивать взаимодействие со спутниками связи.

Профиль включает в себя задачи по двум школьным предметам: **физика** и **информатика.**

§1 Первый отборочный этап

Первый отборочный тур проводится индивидуально в сети интернет, работы оцениваются автоматически средствами системы онлайн-тестирования. Для каждой из параллелей (9 класс или 10-11 класс) предлагается свой набор задач по физике, информатика общая для всех классов. На решение задач учащимся отводилось 2 суток. Первый этап состоял из двух независимых попыток с разными задачами, участник мог решать только одну попытку (в случае, если он решал обе, в зачет шла лучшая). Решение каждой задачи дает определенное количество баллов. Баллы зачисляются в полном объеме за правильное решение задачи. Участники получают оценку за решение задач в совокупности по всем предметам данного профиля (математика и физика) - суммарно от 0 до 25 баллов.

1.1 Первая попытка
Задачи по физике (9 класс)

Задача 1.1.1 (2 балла)

Условие:

Квадрокоптер, движущийся параллельно поверхности Земли со скоростью V = 3м/с, обнаружил впереди себя птицу, летящую точно навстречу, с такой же по модулю скоростью. Программой квадрокоптера предусмотрен маневр уклонения, при котором коптер сразу после обнаружения препятствия совершает разворот в плоскости параллельной поверхности Земли, сохраняя при этом постоянную по модулю скорость и постоянное ускорение а = 2,1м/с^2. После завершения разворота птица оказывается неподвижной относительно квадрокоптера и находится при этом от него на том же расстоянии, на котором была, когда коптер ее обнаружил. Найдите это расстояние в метрах с точностью до десятых.

Решение:

Поскольку квадрокоптер оказался неподвижен относительно птицы после разворота, это значит, что он совершил разворот на 180°. Радиус разворота можно найти, зная центростремительное ускорение и скорость разворота., следовательно, разворот совершен за время, а птица за это время прошла расстояние. Тогда положение объектов после разворота можно изобразить на рисунке, где x – искомое расстояние.

Используя теорему Пифагора для треугольника на рисунке получим:

$$x^2 = (x - \pi R)^2 + 4R^2, \text{откуда } x = \frac{(4+\pi^2)}{2\pi} R = \frac{(4+\pi^2)}{2\pi} \frac{V^2}{a} = 9.5\text{м}$$

Ответ:
L = 9.46м

Задача 1.1.2 (3 балла)

Условие:

На некоторой планете может быть реализован следующий эксперимент. При плоских колебаниях математического маятника длиной L = 3 м максимальная сила натяжения нити отличается от минимальной в k = 4 раза, если максимальный угол отклонения равен некоторому значению α. Такой же угол α с вертикалью образует нить маятника, если она вращается с периодом T = 4,0 с вокруг вертикальной оси, проходящей через точку подвеса. Определите ускорение свободного падения на данной планете. (Ответ округлите до десятых).

Решение:

На рисунке 1 показаны два крайних положения маятника, совершающего плоские колебания. Применим второй закон Ньютона для произвольного угла отклонения от вертикали ϕ в проекциях на направление нити:

$$\frac{mv^2(\varphi)}{L} = T(\varphi) - mg\cos\varphi, \quad \text{или} \quad T(\varphi) = \frac{mv^2(\varphi)}{L} + mg\cos\varphi,$$

где v(ϕ) — скорость маятника, T(ϕ) — сила натяжения нити при произвольном угле отклонения ϕ. Поскольку скорость v и cosϕ возрастают при переходе от максимального отклонения ϕ = α к положению равновесия ϕ = 0, то

(1) $$T_{\max} = mg + \frac{mv^2}{L},$$

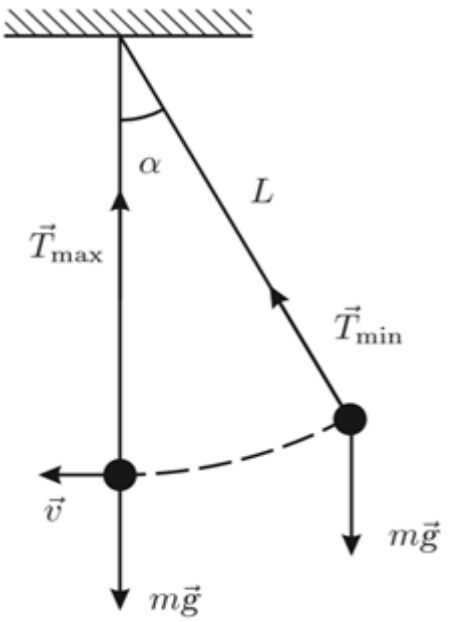


Рис 1

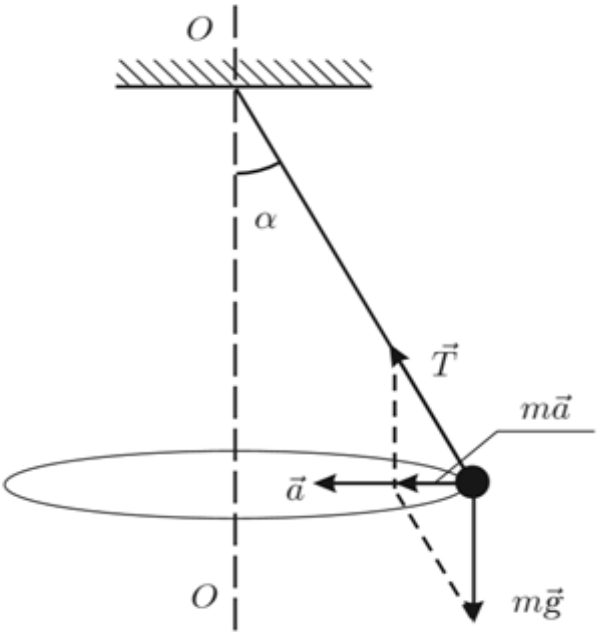


Рис 2

$$T_{\min} = mg\cos\alpha. \quad (2)$$

Скорость v в положении равновесия найдём, воспользовавшись законом сохранения энергии:

$$\frac{mv^2}{2} = mgL(1-\cos\alpha). \quad (3)$$

Из (1), (2) и (3) получим

$$\cos\alpha = \frac{3}{k+2} = \frac{1}{2}. \quad (4)$$

Рассмотрим теперь вращательное движения маятника (конический маятник), изображённого на рисунке 2. Ускорение груза ~a направлено к оси вращения OO. ~ T — сила натяжения нити. По второму закону Ньютона

$$m\vec{a} = m\vec{g} + \vec{T}.$$

Из рисунка 2 видно, что

$$ma = mg\,\mathrm{tg}\,\alpha. \quad (5)$$

Центростремительное ускорение

$$a = \omega^2 L\sin\alpha. \quad (6)$$

Из (5), (6) и (4) получим:

$$g = \omega^2 L\cos\alpha = \frac{3}{k+2}\left(\frac{2\pi}{T}\right)^2 L = 3{,}7\ \text{м/с}^2.$$

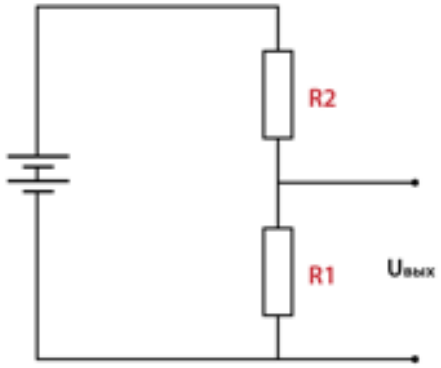
Ответ:
3,7 м/с2

Задача 1.1.3 (3 балла)

Условие:

Вам дано два LiPo аккумулятора номинальным напряжением 3,7В. Вам необходимо контролировать напряжение на аккумуляторе чтобы не допускать его разрядку ниже 3В. На полностью заряженном аккумуляторе напряжение может доходить до 4.2В. АЦП контроллера (вольтметр) может измерить напряжение до 5В.
У вас есть любые резисторы мощностью 0.05Вт. Необходимо рассчитать схему делителя дающую запас по входу АЦП 10%, и обеспечивающую запас мощности резисторов 80%.

Рассчитайте R1 и R2 удовлетворяющие заданным параметрам?



Решение:

Максимальная мощность на резисторе 0.2*0.05 = 0.01
Максимальное напряжение после деления на АЦП 0.9*5 = 4.5В
(8.4) ^2/(R1+R2) =0.01
R1+R2 = 7056 - минимальные сопротивления

Uвых/U = R1/(R1+R2)
4.5/ 8.4 = R1/(7056)
R1=3780
R2=3276

Ответ:

R1=3780
R2=3276

Задача 1.1.4 (2 балла)

Условие:

Металлический шарик массой 100 грамм сбрасывают, падает на пружину с жесткостью 750 Н/м. После падения шарик отскакивает от пружины на высоту 10 см, падает назад, отскакивает на высоту 1,5 см, после чего система успокаивается (шарик остается на пружине).

На сколько мм сожмется пружина? Ответ округлите до десятой.

Решение:

mg = kdx
*dx=0.1*9.8/750 = 0.001306 м*
1м = 1000 мм
1.3мм

Ответ:

1.3

1.2 Первая попытка
Задачи по физике (10-11 класс)

Задача 1.2.1 (2 балла)

Условие:

Квадрокоптер массой 500г поддерживается в воздухе 4 воздушными винтами с размахом лопастей 10 см каждый. Какова должна быть минимальная полезная мощность каждого из двигателей, вращающих винты этого квадрокоптера, если каждый двигатель вращает один воздушный винт? Ответ выразите в ваттах с точностью до целых. Квадрокоптер предполагается использовать при температуре 27˚С и давлении 760мм рт.ст.

Решение:

*N = ((mg)^3/(16*pi*ro*d^2))^1/2 =14Вт , где m - масса аппарата, d- диаметр лопастей, ro = R*mu/RT = 1.178 кг/м^3 - плотность воздуха.*

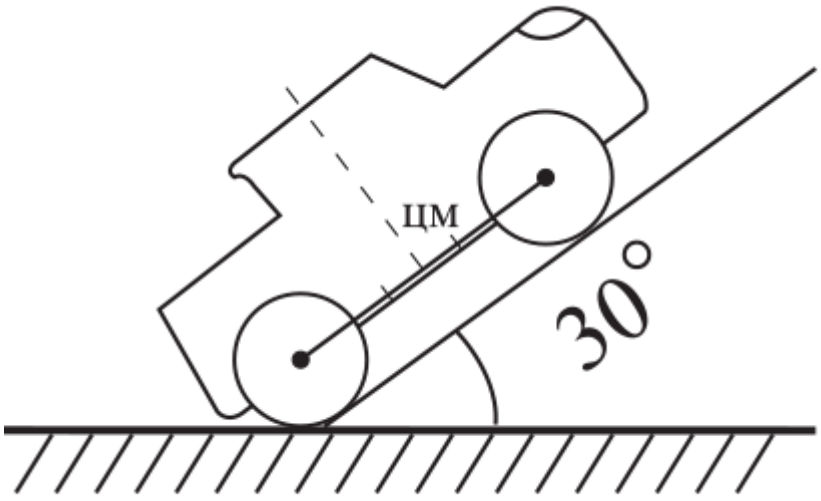
Ответ:

14Вт

Задача 1.2.2 (3 балла)

Условие:

Переднеприводный автомобиль съехал с обледеневшей горки на лед озера и остановился так, как показано на рисунке. Передние колеса находятся на горке, а задние на идеально гладком льду и горки не касаются. Каким должен быть минимальный коэффициент трения между передними колесами и поверхностью горки, чтобы автомобиль мог заехать на нее обратно. Ответ округлите до сотых. Проекция центра тяжести автомобиля находится ровно посередине прямой соединяющей центры колес.



Решение:

Минимальный коэффициент трения между передними колесами и поверхностью горки, чтобы автомобиль мог заехать на нее, равен (mg сократились) M*тр*=tga(1+cos^2(a))=1.01

Ответ:
1.01

Задача 1.2.3 (2 балла)
Условие:
Искусственный спутник Земли движется по круговой орбите на высоте h=476 км. Чему равна скорость его движения?

Решение:
Центростремительная сила, удерживающая спутник на круговой орбите, равна гравитационной силе притяжения:
*m*V^2/r = G*m*M/r^2*
r=Rземли+h
*V=Sqrt((G*M)/(Rземли + h))*
*G = 6.67408*10^-11*
*M= 5.9742*10^24*
Rземли = 6371 км
*V= Sqrt((6.67408*10^-11*5.9742*10^24)/(6371000+476000))*

Ответ:
V=7,62км/с.

Задача 1.2.4 (3 балла)
Условие: У вас есть конденсатор, который вы разряжаете через некоторую нагрузку. Каждую секунду заряд конденсатора уменьшается на 0.1%. Через сколько секунд заряд на конденсаторе уменьшится в два раза?
Ответ округлите до целых.

Решение:
I=U/R
U=q/c
I=dq/dt
dq/dt = q/rc
dq/q = dt/rc
Для определения заряда в любое время:
Интеграл (от Q до q) dq/q = Интеграл (от 0 до t) dt/rc
-ln(Q/q)=t/RC
*q=Q*e^(-t/RC)*
--
q/Q = 0.5
*-ln(0.5)*R*C=t*
--
q/Q = 0.999 t=1c
RC = 1/ -ln(0.999) =999.499

*t = -999.499*ln(0.5)= 692.79*

Ответ:
693

1.3 Первая попытка
Информатика (9-11 класс)

Задача 1.3.1 (1 балл)

Условие:

Пусть задано число N (не более 10 000). Необходимо вывести количество натуральных чисел, не превосходящих N и при этом нацело делящихся на свою сумму цифр.

```
Sample Input:
  2401
Sample Output:
  485
```

Решение:

Реализовано на языке C:

```
#include <iostream>
using namespace std;

int fsum(int n){
      int sum=0;
      while(n>9){
            sum+=n%10;
            n/=10;
      }
      sum+=n;
      return sum;
}

int main()
{
      int in, out=0, sum=0;
      cin>>in;
      if(in<=10){
            cout<<in;
            return 0;
      }
      else{
            out+=10;
            for(int i=11; i<=in; i++){
                  sum=fsum(i);
                  if(i%sum==0)
                        out++;
            }
            cout<<out;
      }
      return 0;
}
```

Критерии оценки:

Для генерации тестов и проверки результата используется следующий код на языке Python.

```
#Python3
import random
import math
def generate():
   num_tests = 20
   tests = []
   for test in range(num_tests):
       a = random.randrange(9999)
       a=a+1
       test_case = "{} ".format(a)
       tests.append(test_case)
   return tests

def solve(dataset):
   a = dataset

   #print("start "+str(a))
   count = 0
   for  num in range(1,int(a)+1):
       mod = 0
       sumMod = 0
       ind = 0
       #print("num"+str(num))
       while (10**ind <= num):
           #print  ("Ind"+str(ind))
           dig1 = math.floor(num/(10**ind))%10
           ind = ind+1
           #print  ("indI"+str(ind))

           #print(str(dig1))
           sumMod =sumMod+dig1
       #print (str(sumMod))
       if (num % sumMod == 0):
           count=count+1
           #print ("ok!")

   #print (str(count))
   return str(count)

def check(reply, clue):
   return int(reply) == int(clue)
```

Задача 1.3.2 (3 балла)

Условие:

В компьютерной игре есть N уровней (500>N>=1) . По завершению каждого уровня полагается бонус в некотором количестве золотых. Игрок может или взять бонус, или отказаться от бонуса, но тогда бонус следующего уровня удвоится (это работает только один раз, т.е. на следующем уровне он обязан взять бонус, но через уровень может опять отказаться ради удвоения).

На вход подается последовательность натуральных чисел, которые означают бонусы уровней от 1го до Nго, числа от 100 до 500.

В качестве ответа необходимо выдать число равное максимальной сумме золотых, которую пользователь может заработать оптимально, забирая бонусы и отказываясь от них ради удвоения. Верным считается ответ равный или менее чем на 10% отличающийся от теоретически возможной максимальной суммы. Обращаем внимание на ограничение по времени работы программы!

```
Sample Input:
  129 436 447 237 187 305 495 241 193 166 183 427 492 162 352 108 306
  402 497 486 380 238 343 157 255 129 441 260 399 197 318 218 487 416
  449 343 167 421 199 195 245 412 378 428 183 218 315 462 460 387 474
  193 131 374 498 307 154 378 366 242 364 270 329 203 198 440 473 499
  443 443 497 362 265 164 284 282 189 193 489 415 309 460 334 434 448
  491 324 203 320 329 308 182 438 430 140 223 369 172 303 218 443 272
  364 297 221 473 270 138 233 440 400 481 394 329 190 415 320 289 359
  232 457 120 242 248 100 186 300 449 420 464 196 424 361 453 402 274
  155 392 339 375 159 162 111 338 224 103 142 163 491 463 375 232 431
  355 493 408 107 267 184 496 493 110 194 449 390 164 150 489 204 296
  429 442 260 356 461 406 204 488 311 309 361 318 247 282
Sample Output:
  68890
```

Решение:

```
//C
#include <stdio.h>
#include <string.h>
#include <iostream>
using namespace std;
int main()
{
   string str;
   int num1=0;
   int num2=0;
   int all=0;
   int pos=0;
   getline(cin,str);
   //cout << str;
   int len=0;
   for(int i=0;str[i];i++,len++);
   for(int i=0;i<=len-1;i++)
   {
           num1=0;num2=0;
      for(;;i++)
      {
        if(str[i]<='9' and str[i]>='0')break;
      }
      while(str[i]<='9' and str[i]>='0' and i<=len-1)
      {
       num1=num1*10+((int)str[i]-48);
       i++;
      }
      if(i>=len-1)
      {
          all+=num1;
          break;
      }
      else
```

```
       {
       i++;
       while(str[i]<='9' and str[i]>='0' and i<=len-1)
       {
        num2=num2*10+((int)str[i]-48);
        i++;
       }

      // printf("\n%d\n%d\n",num1,num2);
       if(num2>num1){all+=(num2*2);}
       else {all+=(num1+num2);}

       if(i>=len-1)break;
       }




   }



   printf("%d",all);
   getchar();getchar();
   return 0;
}
```

Критерии оценки:

```
#Python3
#This is sample code challenge
import random
import math

def getMax(lvls, pos):
   if pos == 0:
       return (lvls[0],lvls[0])
   thisPrice = 0
   if pos < len(lvls):
       thisPrice = lvls[pos]

   (maxPrevAny, maxPrevAcc) = getMax(lvls,pos-1)
   #maxPrevAcc = getMax(lvls,pos-1,True)

   noAccept = max(int(maxPrevAny),int(maxPrevAcc))+int(thisPrice)
   accept = int(maxPrevAcc) + 2*int(thisPrice)-int(lvls[pos-1])
   maxResAny = max(accept,noAccept)
   return (int(maxResAny),int(noAccept))

def generate():
   num_tests = 10
   tests = []
   for test in range(num_tests):
       a = random.randrange(500)
       a = a+1
       res = ""
```

```
        for lvl in range(0,a):
            lvl = random.randrange(400)
            lvl = lvl+100
            res = res + " "+ str(lvl)
        test = res;
        tests.append(test)
       # print (str(res))
    return tests

def solve(dataset):
    a = dataset.split()
    #print("input"+ str(a))
    if len(a)==0:
        return 0
    res = max(getMax(a,len(a)))
    #print("res "+ str(res))

    return str(res)

def check(reply, clue):
    if int(reply) == int(clue):
        return int(reply) == int(clue)
    return max(int(reply),int(clue))/min(int(reply),int(clue))<1.11
```

Задача 1.3.3 (1 балл)

Условие:

Беспилотный летательный аппарат оснащен направленным вниз ультразвуковым сонаром для измерения высоты полёта. Сонар испускает ультразвуковой импульс, после чего принимает отражённый от земли сигнал. Рассчитав задержку между испусканием импульса и приемом отраженного сигнала и зная скорость звука в среде, можно определить высоту полёта. Однако, отраженный сигнал не всегда фиксируется точно, поэтому часто приходится производить несколько измерений и усреднять их. Для усреднения результатов было решено использовать их медиану.

Медиана — такое число выборки, что половина из элементов выборки больше него, а другая половина меньше него. Например, если получены значения { 8, 1, 1, 3, 9 }, медианой является число 3. Если в выборке чётное число значений, медианой считать среднее арифметическое двух средних значений: например, для выборки { 4, 1, 2, 3 } медианой следует считать 2,5 (среднее арифметическое 2 и 3).

Получив серию измерений с сонара, рассчитайте расстояние до земли. Рассчитайте и выведите медиану полученных значений. Скорость звука принять равной 340 м/с.

Формат входных данных:

Первая строка содержит число измерений n, последующие n строк — данные с сонара: отметка времени испускания сигнала и отметка времени приема отраженного сигнала, разделенные пробелом. Отметка времени — число миллисекунд, прошедшее с момента включения летательного аппарата до наступления события. Представляет собой целое неотрицательное число, не превышающее $4 * 10^9$.

Формат выходных данных:

Высота полёта в метрах (точность не менее 0,01 м), дробная часть числа отделяется точкой.

```
Sample Input 1:
  47
  462373 462418
  462483 462529
```

```
  462648 462651
  462798 462831
  463022 463025
  463119 463156
  463226 463254
  463298 463324
  463497 463552
  463625 463654
  463758 463773
  463773 463821
  464005 464059
  464240 464250
  464391 464402
  464524 464580
  464627 464669
  464800 464811
  464816 464844
  464976 464998
  465017 465027
  465041 465085
  465176 465176
  465362 465369
  465522 465554
  465663 465697
  465852 465910
  465922 465969
  466056 466057
  466095 466146
  466327 466342
  466476 466497
  466561 466572
  466688 466730
  466843 466887
  467045 467086
  467280 467314
  467476 467520
  467552 467570
  467684 467729
  467807 467808
  467955 467995
  468194 468208
  468350 468398
  468567 468567
  468692 468728
  468891 468943

Sample Output 1:
  5.6100
Sample Input 2:
  12
  814020 814055
  814118 814129
  814165 814206
  814341 814351
  814442 814447
  814610 814656
  814679 814683
  814772 814784
```

```
  814835 814889
  814949 814954
  814983 815023
  815184 815239

Sample Output 2:
  3.9950
```

Решение:

```
#Python3
import sys

SOUND_SPEED = 340.0

string_in = ' '.join([value.strip() for value in
sys.stdin.readlines()])
arr = [int(value) for value in string_in.split(' ')]

arr_delta = []
for i in range(1, len(arr), 2):
   arr_delta.append(arr[i+1]-arr[i])

arr_delta.sort()
len_div = len(arr_delta)//2

if len(arr_delta) % 2 == 0:
   M = (arr_delta[len_div]+arr_delta[len_div-1])/2
else:
   M = arr_delta[len_div]

print('{:.4f}'.format(SOUND_SPEED*(M/2)/1000))
```

Критерии оценки:

```
# Python 3

import random
from statistics import median

def generate():
   num_tests = 50
   tests = []

   for test in range(num_tests):
       num_measurements = random.randint(0, 9) * (10 **
random.randrange(3)) + random.randint(1, 9)
       test_case = [ str(num_measurements) ]
       last_time_of_detection = random.randrange(1000000)
       for _ in range(num_measurements):
           time_of_emission = last_time_of_detection +
random.randrange(200)
           time_of_detection = time_of_emission + random.randrange(60)
           last_time_of_detection = time_of_detection
```

```
            measurement = "{} {}".format(time_of_emission,
time_of_detection)
            test_case.append(measurement)
        test_case_str = "\n".join(test_case) + "\n"
        tests.append(test_case_str)

    return tests

def solve(dataset):
    parsed_dataset = dataset.splitlines()
    n = int(parsed_dataset[0])
    time_of_flight = []
    for x in range(n):
        time_of_emission, time_of_detection = (int(i) for i in
parsed_dataset[x+1].split())
        time_of_flight.append(time_of_detection - time_of_emission)

    distances = [tof / 2 * 0.340 for tof in time_of_flight]
    return "{:.4f}".format(median(distances))

def check(reply, clue):
      answer_abs_error = abs(float(clue) - float(reply))
      return answer_abs_error < 0.01
```

Задача 1.3.4 (2 балла)

Условие:

Точка посадки беспилотного летательного аппарата отмечена с помощью маркера квадратной формы определенного размера, который распознается бортовым компьютером с камерой, направленной вертикально вниз. Программа, осуществляющая распознавание маркера, выводит его смещение относительно центра кадра и площадь (в пикселях). Наблюдаемая площадь маркера зависит только от высоты полета, а смещение относительно центра кадра — только от смещения коптера в горизонтальной плоскости. Экспериментально установлено, что на высоте 1 м над маркером его площадь составляет 250000 пикселей, а смещению маркера на 1 м в горизонтальной плоскости на той же высоте соответствует смещение его изображения на 1000 пикселей. Камеру считать идеальной дырочной (оптика не вносит искажений).

Требуется на основании данных с бортового компьютера определить расстояние от находящегося в воздухе коптера до точки посадки.

Формат входных данных:

Первая строка содержит координаты распознанного маркера (в пикселях, начало координат в центре кадра), вторая — его площадь (в пикселях). Числа являются целыми. Координаты по модулю не превышают 2000 пикселей, площадь — не более 16 000 000 пикселей.

Формат выходных данных:

Расстояние до точки посадки в метрах (точность не менее 0,01 м), дробная часть числа отделяется точкой.

Иллюстрации:

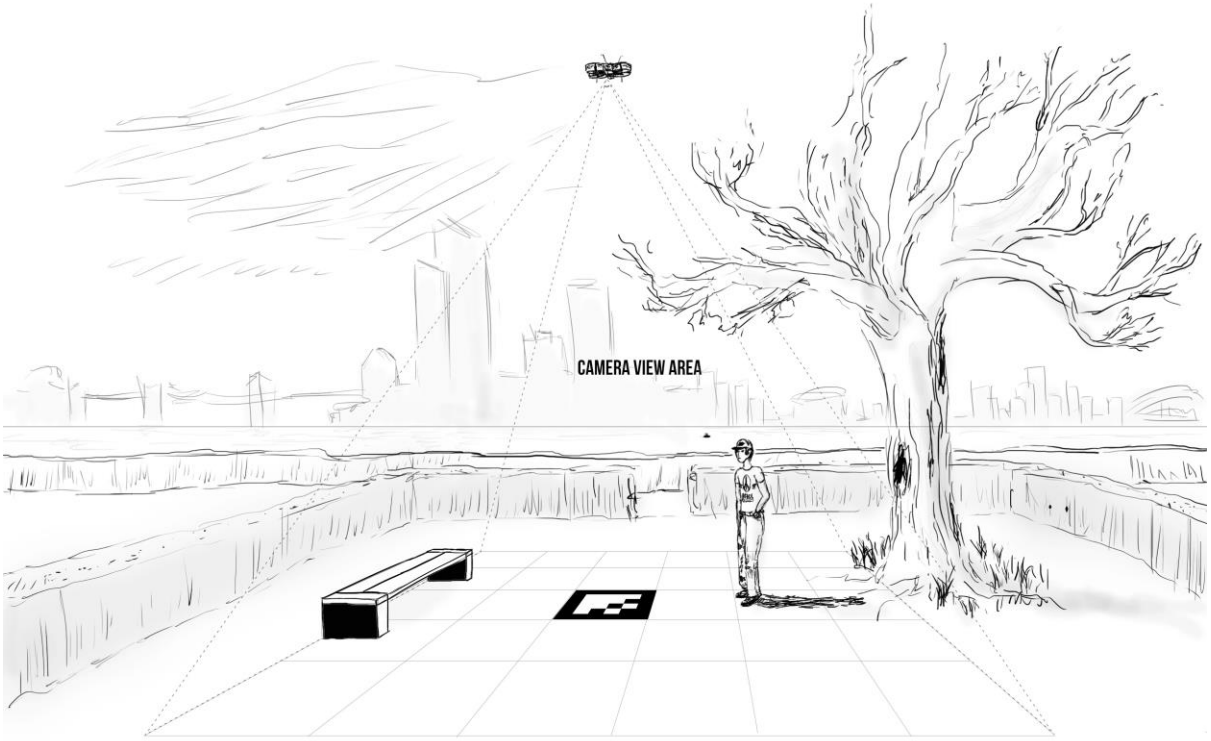


Рис. 1. Коптер над ArUco-маркером

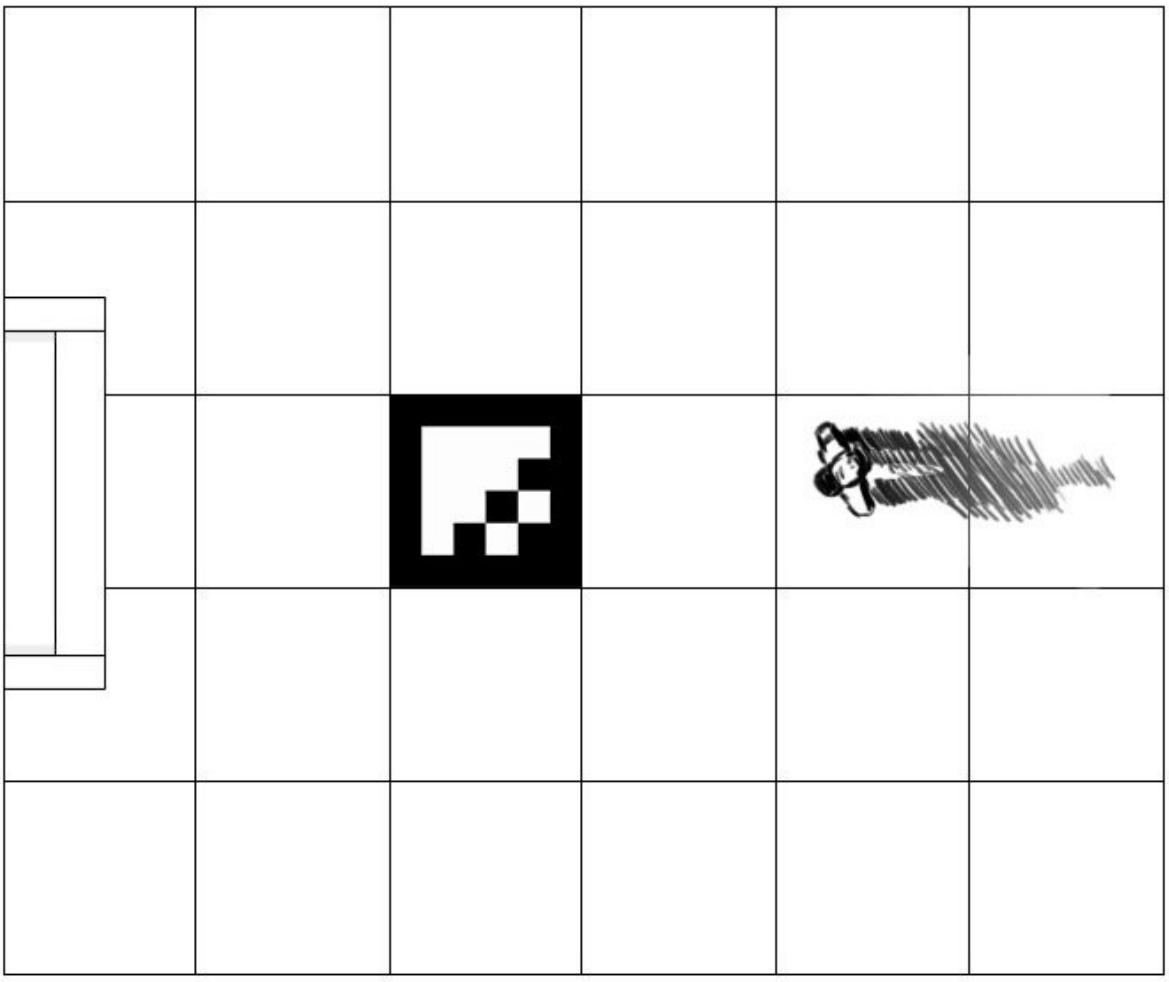


Рис. 2. Изображение с бортовой камеры

```
Sample Input 1:
  -97 -796
  11961

Sample Output 1:
  5.860144769058316
Sample Input 2:
  1537 1617
  1638673

Sample Output 2:
  0.954921321547401
Sample Input 3:
  -1586 -28
  84327

Sample Output 3:
  3.228659372895824
```

Решение:

```
// C

#include<stdio.h>
#include<math.h>

double sqrt(double x);
int main()
{
double x, y, s, h, l, q;
scanf("%lf %lf %lf", &x, &y, &s);
h=1000*500/sqrt(s);
l=500*x/sqrt(s);
q=500*y/sqrt(s);
printf("%lf", sqrt(h*h + l*l + q*q)/1000);
return 0;

}
```

Критерии оценки:

```
import random
from math import sqrt

def generate():
   num_tests = 100
   tests = []
   for test in range(num_tests):
       x = random.randint(-2000, 2000)
       y = random.randint(-2000, 2000)
       area = (random.randint(1, 20) ** 4) * (random.randint(1, 9)
** 2) + random.randint(0, 1000)
       test_case = "{} {}\n{}\n".format(x, y, area)
       tests.append(test_case)
   return tests
```

```
def solve(dataset):
   parsed_dataset = dataset.splitlines()
   marker_position = tuple(int(i) for i in
parsed_dataset[0].split())
   marker_area = int(parsed_dataset[1])

   height = 1 / (sqrt(marker_area) / 500)
   distance_XY = tuple(i / 1000 * height for i in marker_position)
   distance = sqrt(sum(i ** 2 for i in distance_XY) + height ** 2)

   return str(distance)

def check(reply, clue):
   answer_abs_error = abs(float(clue) - float(reply))
   return answer_abs_error <= 0.01
```

Задача 1.3.5 (3 балла)

Условие:

Два квадрокоптера летают на одной высоте. Чтобы избежать столкновений, перед вылетом аппараты отправляют свои полетные программы на центральный сервер, который должен убедиться в том, что катастрофы в воздухе не произойдёт. Формат полётной программы следующий: в первой строке записано число инструкций n, в следующих n строках содержатся выполняемые последовательно инструкции, состоящие из буквы, определяющей направление полета, и числа секунд, которое аппарат будет лететь в данном направлении. Направления кодируются следующим образом: N — север, E — восток, S — юг, W — запад, H — аппарат зависает без движения, промежуточные направления отсутствуют. Скорость полёта коптеров одинакова и составляет 10 м/с. Число секунд в инструкции всегда целое. После выполнения последней инструкции аппарат совершает посадку и уже не может ни с чем столкнуться. Точка взлёта второго коптера расположена на 100 м южнее и на 350 м западнее точки взлёта первого коптера, взлёт осуществляется одновременно.

Требуется установить, произойдёт ли столкновение и, если да, то спустя какое время с момента начала полёта.

Формат входных данных:

Полетная программа первого коптера, затем — полётная программа второго коптера.

Таким образом, в первой строке содержится число полетных инструкций n первого коптера, в следующих n строках — полётные инструкции первого коптера, в (n+2)-ой строке — число полетных инструкций m второго коптера, в следующих за ней m строках — полетные инструкции второго коптера. Формат полетной инструкции приведен выше, буква от числа отделяется пробелом.

Формат выходных данных:

Целое число: нуль, если столкновение не произойдёт, или число секунд с момента начала полёта до столкновения.

Решение:

```
// C

#include <iostream>
#include <vector>
#include <cstdlib>
```

```
using namespace std;

int main() {
   int n, dx, dy, t, c;
   vector < vector <int> > x(2), y(2);
   x[0].push_back(35);
   y[0].push_back(10);
   x[1].push_back(0);
   y[1].push_back(0);
   for (int k = 0; k < 2; k++) {
       cin >> n;
       dx = 0;
       dy = 0;
       for (int i = 0; i < n; i++) {
           c = ' ';
           while (c == 32 || c == 10) {
               c = cin.get();
           }
           if (c == 'N') {
               dx = 0;
               dy = 1;
           } else if (c == 'S') {
               dx = 0;
               dy = -1;
           } else if (c == 'E') {
               dx = 1;
               dy = 0;
           } else if (c == 'W') {
               dx = -1;
               dy = 0;
           } else if (c == 'H') {
               dx = 0;
               dy = 0;
           } else {
               cout << "ERROR";
               return 0;
           }
           cin >> t;
           for (int j = 0; j < t; j++) {
               x[k].push_back(x[k][x[k].size() - 1] + dx);
               y[k].push_back(y[k][y[k].size() - 1] + dy);
           }
       }
   }
   for (int i = 0; i < min(x[0].size(), x[1].size()) + 1; i++) {
       if (x[0][i] == x[1][i] && y[0][i] == y[1][i]) {
           cout << i;
           return 0;
       }
   }
   cout << 0;
   return 0;
}
```

Критерии оценки:

```
#Python3
import random

def generate():

   def generate_program():
       length = random.randint(30, 300)
       program = '{}\n'.format(length)
       for m in range(length):
           instruction = random.choice(['N', 'E', 'S', 'W', 'H'])
           repeats = random.randint(1, 50)
           program += "{} {}\n".format(instruction, repeats)
       return program

   result = []
   check_zero = True
   while len(result) < 100:
       programs = generate_program() + generate_program()
       collision = solve(programs)
       if collision != "0" or not check_zero:
           result.append(programs)
           check_zero = random.randint(0, 10) < 10

   return result

def simulate(start_position, instruction_dict):
   # NED coordinates in meters
   positions = [start_position]
   for instruction in instruction_dict:
       for second in range(instruction['time']):
           if instruction['direction'] == 'N':
               positions.append((positions[-1][0] + 10, positions[-
1][1]))
           elif instruction['direction'] == 'E':
               positions.append((positions[-1][0], positions[-1][1]
+ 10))
           elif instruction['direction'] == 'S':
               positions.append((positions[-1][0] - 10, positions[-
1][1]))
           elif instruction['direction'] == 'W':
               positions.append((positions[-1][0], positions[-1][1]
- 10))
           elif instruction['direction'] == 'H':
               positions.append(positions[-1])
   return positions

def solve(dataset):
   lines = dataset.splitlines()

   num_first_instructions = int(lines.pop(0))
   first_instructions = []
   for string_num in range(num_first_instructions):
       raw_instruction = lines.pop(0).split()
       instruction = {
           'direction': raw_instruction[0],
           'time': int(raw_instruction[1])
```

```
        }
        first_instructions.append(instruction)

    num_second_instructions = int(lines.pop(0))
    second_instructions = []
    for string_num in range(num_second_instructions):
        raw_instruction = lines.pop(0).split()
        instruction = {
            'direction': raw_instruction[0],
            'time': int(raw_instruction[1])
        }
        second_instructions.append(instruction)

    first_positions = simulate((0, 0), first_instructions)
    second_positions = simulate((-100, -350), second_instructions)

    time_to_collision = 0

    for index, (first_pos, second_pos) in
enumerate(zip(first_positions, second_positions)):
        if first_pos == second_pos:
            time_to_collision = index
            break

    return str(time_to_collision)

def check(reply, clue):
    return int(reply) == int(clue)
```

1.4 Вторая попытка
Задачи по физике (9 класс)

Задача 1.4.1 (2 балла)

Условие:

Каков предельный угол наклона дороги, на котором можно остановить автомобиль, если известно, что для того, чтобы пройти на ровном участке дороги поворот радиусом R = 25м, не вылетев с трассы, ему пришлось снизить скорость до 40км/ч. Ответ дайте в градусах округлив до целых.

Решение:

Запишем 2 закон Ньютона, для автомобиля, проходящего поворот, учтя, что движется он с центростремительным ускорением V^2/R = mu*mg, где mu - коэффициент трения между колесами и поверхностью. В свою очередь известно, что максимальный угол наклона поверхности, на которой тело покоится может быть найден как alpha = arctg(mu)

Ответ:

26+-1

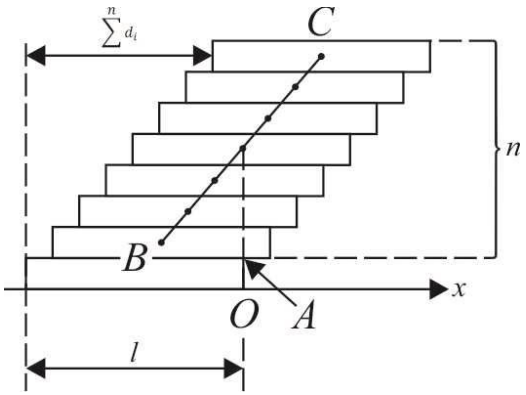
Задача 1.4.2 (3 балла)

Условие:

При разгрузке корабля в порту, коробки с грузом укладывают на горизонтальную поверхность друг на друга, смещая каждую последующую коробку относительно предыдущей вдоль длинной стороны на расстояние dn=1+5(n-1) см, где n - номер укладываемой коробки (нижнюю коробку считаем “нулевой”). Какое максимальное количество коробок N можно уложить друг на друга прежде чем коробки начнут падать? Все коробки имеют одинаковые линейные размеры (a=0,4 м, b=1 м, c=2,5 м) и массу m=25 кг, масса равномерно распределена по всему объему коробки.

Решение:

Коробки начнут падать, когда часть стопки, состоящая из n коробок, начнет опрокидываться, вращаясь вокруг оси, проходящей через точку A (см. рисунок).



Центр тяжести системы из n коробок, лежащих на нижней (n+1)й коробке, располагается на середине прямой BC, соединяющей их центры тяжести. Выбрав начало координат в точке O, находим горизонтальные координаты точек B и C: $x_B = d_1 - l/2$, $x_C = d_n - l/2$. *Где l- длина стороны максимального размера, вдоль которой происходит смещение (в данном случае стороны c и l=250 см).* Следовательно, горизонтальная координата центра тяжести этой системы

$$x_O = \frac{x_B + x_C}{2} = \frac{d_1 - l/2 + d_1 + 5(n-1) - l/2}{2} = \frac{2d_1 - l + 5(n-1)}{2}$$

Условие равновесия системы из n коробок имеет вид: $x_O \leq 0$ и, следовательно, $n \leq 50{,}6$, так как N=n+1=51

Ответ:
51

Задача 1.4.3. (2 балла)

Условие:

К телу массой 100 грамм находящемуся в состоянии покоя приложена сила F = 50Н направленная вертикально вверх. Сила действует на тело в течении 2-х секунд и пропадает.

Найти время, через которое тело достигает максимальной высоты относительно начального положения. Ответ округлить до ближайшего целого числа.

Решение:

F=m*a Так же на тело действует сила тяжести mg
F-m*g=m*a
a=F-m*g/m
t1=2c
V =(F-m*g)*t1/m - приобретенная скорость
Максимальный подъем V -> 0, действует только сила тяжести

V=m*g*t2/m
t2=V/g = (F-m*g)*t1/m*g = F*t1/m*g-t1

Время подъема относительно начальной точки T=t1+t2 = F*t1/m*g = 50*2/0.1*9.81 = 101.93

Ответ:
102

Задача 1.4.4 (3 балла)

Условие:

Электрическую лампу включили в цепь с регулируемым напряжением. Для накала нити в лампе требуется напряжение. В начальный момент напряжение на лампе было равно 3В. Вследствие испарения материала нити за каждый час ее диаметр уменьшается на 20% от имеющегося (при заданном напряжении). Какое напряжение необходимо подать через два часа работы лампы, чтобы температура накала стала такой же, как и в начальный момент?

Решение:
Рпад=Ризл
Рпад=U^2/R
*Ризл=k*f(T)*S*
*U^2/R=k*f(T)*S*
*U=k*f(T)*П*d*l*p*l/S*
*U=k*f(T)*П*d*l*p*4*l/П*d*
*U^2=k*f(T)*p*4*l^2/d*
Через два часа работы ее диаметр будет:
*d2=0.8^2*d1*
*U1^2=k*f(T)*p*4*l^2/d1*
*U2^2=k*f(T)*p*4*l^2/d2*

U1^2/U2^2=d2/d1
*U2=U1*sqrt(d1/d2)=3*sqrt(1/0.8^2)=3.75*

Ответ:
3,75 В

1.5 Вторая попытка
Задачи по физике (10-11 класс)

Задача 1.5.1 (2 балла)
Условие:

Квадрокоптер, летит со скоростью V = 10м/с на высоте H = 30м по прямой проходящей над оператором. Для управления квадрокоптером оператор использует ик-пульт. В момент, когда расстояние между оператором и квадрокоптером по прямой составляет d = 100м, оператор нажимает кнопку на пульте. После нажатия кнопки коптер с некоторой задержкой сбрасывает груз, причем оператор ловит этот груз не сходя со своего места. Пренебрегая сопротивлением воздуха при падении груза найдите, время задержки. Ответ дайте в секундах с точностью до десятых.

Решение:

Сигнал ИК-пульта распространяется со скоростью света, т.е. временем, за которое сигнал преодолел расстояние d можно пренебречь. Зная d и h можно найти расстояние вдоль поверхности между оператором L, тогда полное время между нажатием на кнопку и поимкой груза может быть найдено, как tau0 = L/V, т.е. оторвавшись от коптера груз будет иметь туже скорость вдоль оси параллельной Земле. Тогда остается вычесть из этого времени время на падение груза tau1 = (2h/g)^½. Искомое время tau = tau0-tau1

Ответ:
7.09+-0.03с

Задача 1.5.2 (1 балл)
Условие:

Плавучая станция мониторинга водоемов работает в море. Плотность морской воды составляет: $\rho = 1{,}1 \cdot 10^3$ кг/м^3 . Поддерживающее основание станции имеет объем: $V_1 = 21{,}2 \cdot 10^{-3}$ м^3 и полностью погружено в воду. Камера с полезной нагрузкой весом $P_2 = 712\ H$ находится над поверхностью воды на 0,1 ее объема. Средняя плотность камеры с полезной нагрузкой составляет: $\rho_2 = 1{,}2 \cdot 10^3$ кг/м^3. Определить среднюю плотность материала, из которого сделано поддерживающее основание станции.

Решение:

Камера с полезной нагрузкой и поддерживающее основание находится в воде в равновесии, а значит сумма всех сил, действующих на систему равна 0. Следовательно, $P_1 + P_2 - F_1 - F_2 = 0$, где P_1- вес поддерживающего основания, P_2- вес камеры с полезной нагрузкой, F_1 - сила, выталкивающая поддерживающее основание, F_2- сила, выталкивающая камеру с полезной нагрузкой.
Заменим эти силы соответствующими выражениями
$P_1 = \rho_1 \cdot g \cdot V_1$,$F_1 = \rho \cdot g \cdot V_1$, $F_2 = \rho \cdot g \cdot V_2 = \rho \cdot g \cdot 0{,}9 \cdot P_2/(\rho_2 \cdot g) = 0{,}9 \cdot P_2 \cdot \rho/\rho_2$так как $V_2 = 0{,}9 \cdot P_2/(\rho_2 \cdot g)$
Получим:

$\rho_1 \cdot g \cdot V_1 + P_2 - \rho \cdot g \cdot V_1 - 0{,}9 \cdot P_2 \cdot \rho/\rho_2 = 0$, откуда

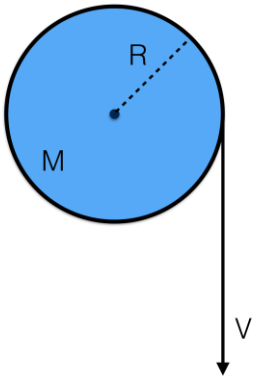
$$\rho_1 = (\rho \cdot g \cdot V_1 + 0{,}9 \cdot P_2 \cdot \rho/\rho_2 - P_2)/(g \cdot V_1) \approx 0{,}5 \cdot 10^3 \text{кг/м}^3$$

Ответ:
$\approx 0{,}5 \cdot 10^3 \text{кг/м}^3$(+- 10 кг/м3)

Задача 1.5.3 (1 балл)

Условие:

К диску массой 1.2 кг и радиусом 10 см привязана тонкая нить. Диск закреплен на идеальной оси (без трения) и может вращаться вокруг нее. Нить выдерживает натяжение 2 Ньютона - после этого рвется. С каким максимальным ускорением можно потянуть за нить начиная с момента покоя системы, чтобы она не порвалась.



Решение:

I=MR^2/2
*M1= T*R - момент силы натяжения*
*M2= I*ε*
M1=M2
T*R=I*ε
T*R=m*R^2*ε/2
T= m*R*ε/2=m*R*a/(2*R)=m*a/2
a=2*T/m

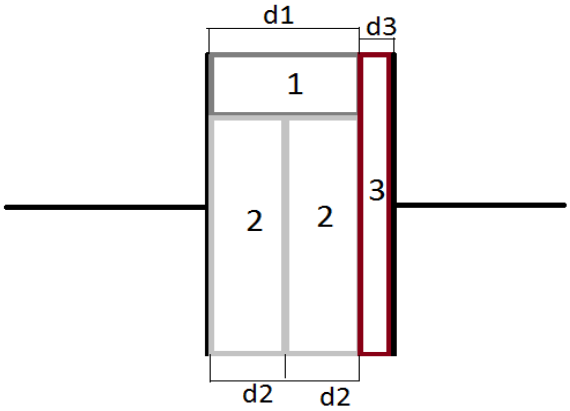
Ответ:

3.(3) Н

Задача 1.5.4 (1 балл)

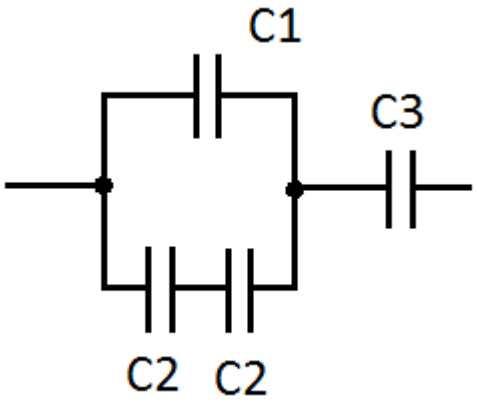
Условие:

Определите электроемкость плоского конденсатора, где в качестве диэлектрика взяли кварц (ε1 =4), стекло (ε2 = 7) и парафин (ε3 = 2) и расположили, как показано на рисунке. Причем, d1=2мм, d2=1мм, d3=0,2мм. Площадь пластин конденсатора S=200см^2. Площадь соприкосновения стекла с пластинами 150см^2, а у кварца 50см^2



Решение:

Данный конденсатор можно представить в следующем виде:



Для последовательного соединения конденсаторов общая емкость вычисляется:
1/Cnm=1/Cn+1/Cm
А для параллельного соединения: Cnm=Cn+Cm
*Емкость плоского конденсатора вычисляется по формуле: C=ε0*ε*S/d*
Для этого случая получается:
1/Собщ= 1/(C1+C2/2)+1/C3
*C20=C1*C2/2*C2=C2/2*
C12=C1+C2/2
1/Собщ=1/(C1+C2/2)+1/C3
Собщ=(C1*C3+C3*C2/2)/(C1+C2/2+C3)
Собщ=(e0*e1*e3*s1*s3/(d1*d3)+e0*e3*e2*s3*s2/(d2*d3*2))/(e1*s1/d1+0.5*e2*s2*d2+e3*s3/d3)
e0=8.85*10^(-12) Ф/м
e1=4 Ф/м, d1= 2 мм, s1=50 см^2
e2=7 Ф/м, d2=1 мм, s2=150 см^2
e3=2, d3=0.2 мм, s3=200 см^2
C1=e0*e1*s1/d1=88.5 ПФ
C2=0.9 нФ
C3=1.77 нФ
C общ=1.05 ПФ

Ответ:
1,05 пФ

1.6 Вторая попытка.
Задачи по информатике (9-11 класс)

Задача 1.6.1 (1 балл)
Условие:
Для составления цепочек длины k разрешается использовать буквы А и Б, причём одна из букв (А или Б) должна стоять в цепочке три или более раз.
Сколько всего существует таких цепочек длины k=4? (Перечислять все такие цепочки не надо - только определить их количество).

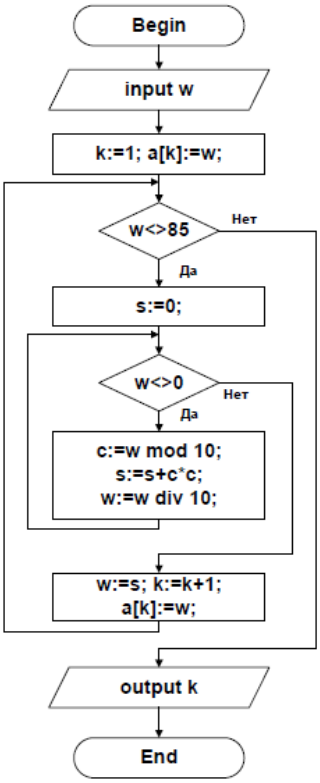
Решение:
аббб, бббa, баaa, аааб,абаа, бабб, ббаб, ааба, бббб, аааа.

Ответ:
10

Задача 1.6.2 (1 балл)

Условие:

Дана блок-схема алгоритма.



Найдите все значения переменной ***w***, которые можно подать на вход алгоритма для того, чтобы выполнились следующие условия:

1. На выходе алгоритма получилось значение переменной ***k***, равное 5.

2. Все элементы массива ***a***, значения которых получались во время выполнения алгоритма, оказались двузначными натуральными числами.

В ответе укажите одно натуральное число – **сумму** найденных значений ***w***.

Решение:

Проанализировав алгоритм, можно увидеть, что он заполняет элементы массива ***a*** числами таким образом, что каждый следующий элемент массива является суммой квадратов цифр предыдущего элемента массива. При этом первый элемент задается с помощью переменной ***w*** на входе алгоритма, и возможные значения этой переменной на входе алгоритма, приводящие к выполнению указанных условий, нам необходимо определить.

Поскольку алгоритм завершился со значением ***k***=5, а условием завершения внешнего цикла является получение значения ***w***=85, пятый элемент массива ***a[5]***=85. Число 85 может быть получено как сумма квадратов цифр двузначных чисел 29, 92, 67 или 76. При этом числа 67, 76 и 92 не могут быть представлены как сумма квадратов двух натуральных чисел, следовательно, ***a[4]***=29.

Число 29 может быть представлено как сумма квадратов цифр 5 и 2, что дает нам двух претендентов на значение ***a[3]***. Рассмотрим оба варианта:

Пусть ***a[3]***=52. 52 может быть представлено суммой квадратов натуральных чисел только как 36+16 = 62 + 42. При этом 46 нельзя представить суммой квадратов двух натуральных чисел, значит ***a[2]***, если следовать этой ветви может быть равно только 64 . Тогда при условии, что все элементы массива должны быть двузначными числами, ***a[1]***, а, следовательно, и ***w***=80. Мы нашли одно возможное значение w на входе алгоритма.

Рассмотрим другой вариант. Пусть ***a[3]***=25. Заметим, что оно может быть получено как сумма квадратов цифр трех вариантов двузначных чисел: 34, 43 и 50. Число 43 можно сразу отбросить, поскольку оно не может быть представлено как сумма квадратов двух натуральных чисел. Рассмотрим два варианта значения ***a[2]*** в этом случае: 34 и 50.

Если ***a[2]***=34, то ***a[1]*** может быть либо 35, либо 53. Мы нашли еще два возможных значения ***w*** на входе алгоритма.

Если ***a[2]***=50, то ***a[1]*** может принять значение или 55, 71 или 17. Следовательно, мы нашли еще три возможных значения ***w*** на входе алгоритма. Обратим внимание, что мы рассмотрели все возможные ветви и, соответственно определили все возможные значения ***w*** на входе алгоритма: 80, 35, 53, 55, 71 и 17. Их сумма равняется 311, что и является ответом на задание.

Ответ:
311

Задача 1.6.3 (1 балл)

Условие:

*Д*ля настройки радиомодуля ему необходимо передать строку из 12 байт в шестнадцатеричном коде согласно следующему описанию:

Байт	**Название**	**Описание**	**По умолчанию (hex)**
0	Адрес назначения	MSB	7E
1			7E
2			7E
3		LSB	7E
4	Адрес устройства	MSB	7E
5			7E
6			7E
7		LSB	7E
8	Канал передачи (RF Channel)	100 kHz шаг	6B
9	Мощность сигнала (Tx Power)	00 = -10 dBm, 01 = -2 dBm, 02 = +6 dBm, 03 = +10 dBm	00
10	Tx - размер пакета данных		1E
11	Rx - размер пакета данных		1E

Значение байта частоты = (Необходимая частота – 422.4MHz) * 10

Какую строку надо отправить в передатчик чтобы сконфигурировать его на работу на частоте 434.2 Мгц и на мощности +6dBm? Остальные настройки должны остаться по умолчанию.

Решение:
(434.2-422.4)*10=118
118 - > hex - > 76

+6dBm ->02

Ответ:
7E7E7E7E7E7E7E7E76021E1E

Задача 1.6.4 (2 балла)

Условие:

Алгоритм, написанный С. Келвин, является считалочкой для роботов. В одну считалочку входит K слов. N роботов расставлены по кругу и пронумерованы от одного до N по часовой стрелке. Счет ведется по часовой стрелке. После K счетов робот, на которого указывает K-тый счет, выбывает, и считалочка начинается снова, пока не останется один робот. Помогите С. Келвин определить число счетов K для заданного количества роботов N, удовлетворяющее следующим условиям:

- робот №1 должен остаться последним
- счет на каждом кругу начинается с робота №1
- выбывающий K-тый робот - всегда робот с наибольшим номером из оставшихся

На стандартный ввод вам подается целое число N от 2 до 20 - количество роботов в кругу.

На стандартный вывод нужно передать целое число K (2<k<2^64), удовлетворяющее перечисленным условиям.

Обратите внимание на то, что в задаче присутсвует ограничение на лимит времени в 1 секунду, умноженный на коэффициент от одного (для языков C, C++) до трех (python3, javascript, ruby etc.)

```
Sample Input 1:
2

Sample Output 1:
2
Sample Input 2:
6

Sample Output 2:
60
```

Решение:

```
N = int(input())
K = 0
if N == 20:
   K = 232792560
else:
   if N == 19:
       K = 232792560
   else:
```

```
        if N == 18:
            K = 12252240
        else:
            if N == 17:
                K = 12252240
            else:
                if N == 16:
                    K = 720720
                else:
                    if N == 15:
                        K = 360360
                    else:
                        if N == 14:
                            K = 360360
                        else:
                            if N == 13:
                                K = 360360
                            else:
                                if N == 12:
                                    K = 27720
                                else:
                                    if N == 11:
                                        K = 27720
                                    else:
                                        if N == 10:
                                            K = 2520
                                        else:
                                            if N == 9:
                                                K = 2520
                                            else:
                                                if N == 8:
                                                    K = 840
                                                else:
                                                    if N == 7:
                                                        K=420
                                                    else:
                                                        if N==6:
                                                            K=60
                                                        else:
                                                            if N==5:
                                                                K=60
                                                            else:
                                                                if
N==4:
                                                                    K=1
2
                                                                else:
                                                                    if
N==3:

 K=6
                                                                    els
e:

 if N==2:

     K=2
print(K)
```

Критерии оценки:

```
#Python3
import random as rd

def generate():
   tests = ['2\n', '6\n', '4\n', '3\n', '5\n', '7\n', '8\n', '9\n',
'10\n', '12\n', str(rd.choice(range(13,16)))+'\n',
str(rd.choice(range(16,21)))+'\n']
   return tests

def gcd(a, b):
   while b != 0:
       t = b
       b = a % b
       a = t
   return a

def produce(n):
   if n == 2:
       return 2
   res = produce(n-1)
   if res % n != 0: return int(res*n/gcd(max(res, n), min(res,n)))
   else: return res

def solve(dataset):
   n = int(dataset.strip())
   return str(produce(n))

def check(reply, clue):
   return int(reply.strip()) == int(clue.strip())
```

Задача 1.6.5 (3 балла)

Условие:

Робот находится в помещении размером 6x6 клеток. Помещение окружено стенами, так же между отдельными клетками могут быть установлены перегородки. Робот движется между отдельными клетками, перемещаясь по одной клетке за один раз. Стены не разрушаются при столкновении, а перегородки - разрушаются. Если робот сталкивается со стеной или с неповрежденной перегородкой, то он остается на той же клетке и разворачивается против часовой стрелки на 90 градусов (если представить поле как правостороннюю систему координат), после чего начинает движение в новом направлении. Перегородка от столкновения повреждается, стена - нет. При столкновении с поврежденной перегородкой робот окончательно разрушает ее, затем продолжает движение в том же направлении, в котором двигался до столкновения.

Каждое движение между клетками или неудачная попытка движения с столкновением отнимает у робота 1 единицу энергии. Робот обладает запасом в 300 единиц энергии, когда энергия заканчивается, он останавливается. Ваша задача - определить, сколько перегородок (целых и поврежденных, но не разрушенных!) останется на поле, когда робот остановится.

Поле задается набором координат клеток по OX в интервале [1 - 6] и по OY в интервале [1 - 6]. Изначально робот находится в клетке (1, 1) и движется в направлении уменьшения координат по OX.

На стандартный ввод вам подается следующий набор значений: в первой строке два целых числа, разделенных пробелом - начальное количество перегородок на поле (не считая стен, ограничивающих поле!) N и размер стороны поля X, в следующих N строках

- набор координат перегородок, по одной в строку. Координаты одной стенки состоят из четырех целых чисел, разделенных пробелами, которые задают координаты двух клеток, которые разделенных данной стенкой. Первые два числа - координаты первой клетки x1, y1, затем идут координаты второй клетки x2, y2.

На стандартный вывод вы должны передать количество стенок, которое останется на поле, когда робот остановится.

```
Sample Input:
  34 6
  4 1 5 1
  5 5 5 4
  4 2 4 3
  5 6 5 5
  2 3 1 3
  5 4 5 3
  2 3 3 3
  4 5 4 4
  5 2 5 3
  2 3 2 4
  4 3 4 4
  3 1 4 1
  6 3 5 3
  4 2 3 2
  5 4 6 4
  4 2 4 1
  3 4 3 3
  1 2 2 2
  2 2 2 3
  5 1 6 1
  2 5 3 5
  4 2 5 2
  3 2 2 2
  1 1 1 2
  3 4 4 4
  3 4 2 4
  3 5 3 4
  5 5 6 5
  2 1 1 1
  2 5 2 4
  3 1 3 2
  1 4 2 4
  4 3 5 3
  1 5 2 5

Sample Output:
  29
```

Решение:

```
  #include <iostream>

  using namespace std;
  int n, xmax, d=0, e=300, x=1, y=1, a, xnew=1, ynew=1;
  bool bah=false;

  int main()
  {
```

```
    cin >> n >> xmax;
    int w[5][n];
    a=n;
    for (int i=0; i<n; i++){
        cin >> w[0][i] >> w[1][i] >> w[2][i] >> w[3][i];
        w[4][i]=2;
    }
    while (e>0){
        xnew=x; ynew=y;
        bah=false;
        switch (d) {
            case 0: ++xnew; break;
            case 1: ++ynew; break;
            case 2: --xnew; break;
            case 3: --ynew; break;
        }
        if (xnew>xmax || ynew>xmax || xnew<1 || ynew<1) {d++;
bah=true;}
        else for (int i=0; i<n; i++) if ((x==w[0][i] && y==w[1][i] &&
xnew==w[2][i] && ynew==w[3][i]) || (xnew==w[0][i] && ynew==w[1][i]
&& x==w[2][i] && y==w[3][i])){
                w[4][i]--;
                bah=true;
                if (w[4][i]==0) {w[0][i]=w[1][i]=w[2][i]=w[3][i]=0;
a--; bah=false; d--;}
                d++;
            }
        if (d==4) d=0;
        e--;
        if (!bah) {x=xnew; y=ynew;}
    }
    cout << a;
    return 0;
}
```

Критерии оценки:

```
import random
import sys

def generate():
    numTests = 20
    numWalls = 50
    tests = []
    fieldSize = 6

    for test in range(numTests):
        wallList = ""
        wallSet = set()
        for _ in range(numWalls):
            x = random.randrange(1, fieldSize)
            y = random.randrange(1, fieldSize)
            z = [0]*4
            z[random.randrange(4)] = random.choice([-1,1])
            wallCoords = (x+z[0], y+z[1], x+z[2], y+z[3])
            if 0 in wallCoords or fieldSize+1 in wallCoords:
```

```
                continue

            wo = Wall(*wallCoords)
            if wo.wallHash() not in wallSet:
                wallSet.add(wo.wallHash())
                wallList += "{} {} {} {}\n".format(wallCoords[0],
wallCoords[1], wallCoords[2], wallCoords[3])
        wallList = str(len(wallSet)) + " " + str(fieldSize) + '\n' +
wallList
        tests.append(wallList)

    return tests

def solve(dataset):
    data = [i for i in dataset.strip().split('\n')]
    wallList = []
    dummy, size = [int(i) for i in data[0].strip().split()]

    for d in data[1:]:
        wallList.append([int(i) for i in d.strip().split()])

    field = Field([Wall(*wall) for wall in wallList], size+1)

    drone = Drone(field)

    return str(drone.move())


def check(reply, clue):
    return int(reply.strip()) == int(clue.strip())


class Field:
    def __init__(self, wallList, border):
        self.nWalls = len(set(wallList))
        # border is the size of box + 1
        # add Walls to the field
        [wallList.append(Wall(0, i, 1, i)) for i in range(1, border)]
        [wallList.append(Wall(border-1, i, border, i)) for i in
range(1, border)]
        [wallList.append(Wall(i, 0, i, 1)) for i in range(1, border)]
        [wallList.append(Wall(i, border-1, i, border)) for i in
range(1, border)]
        self.nWalls = len(set(wallList)) - self.nWalls
        # Walls were added, now form dict
        self.wallDict = {wall.wallHash(): wall for wall in wallList}
        self.border = border


    def hit(self, Wall):
        result = -1
        if Wall.wallHash() in self.wallDict.keys():
            result = self.wallDict[Wall.wallHash()].hit(self.border)
            if result == 0: self.wallDict.pop(Wall.wallHash())

        return result

    def __str__(self):
```

```
        return sum()


class Wall:
    def __init__(self, x1, y1, x2, y2):
        assert (abs(x1-x2) != 1 and abs(y1-y2) != 0) or (abs(x1-x2)
!= 0 and abs(y1-y2) != 1)
        self.tileA = (x1,y1)
        self.tileB = (x2,y2)
        self.state = 2


    def wallHash(self):
        ta = min(self.tileA, self.tileB)
        tb = max(self.tileA, self.tileB)
        return "{}{}{}{}".format(ta[0], ta[1], tb[0], tb[1])


    def touch(self):
        return self.state


    def hit(self, border):
        if '0' in self.wallHash() or str(border) in self.wallHash():

            return self.state
        else:
            self.state -= 1
            return self.state


    def __str__(self):
        return "{} {} {} {}".format(self.tileA[0], self.tileA[1],
self.tileB[0], self.tileB[1])

class Drone:
    def __init__(self, field, stepCounter = 300):
        self.field = field
        self.position = Wall(1,1,0,1)
        self.hit = self.field.hit
        self.stepCounter = stepCounter


    def move(self):
        self.stepCounter -= 1
        result = self.hit(self.position)
        if self.stepCounter == 0: return len(self.field.wallDict) -
self.field.nWalls
        if result == 2 or result == 1: return self.rotate()
        elif result == 0 or result == -1:
            tileA = self.position.tileA
            tileB = self.position.tileB
            dx = tileB[0] - tileA[0]
            dy = tileB[1] - tileA[1]
            self.position = Wall(tileA[0]+dx, tileA[1]+dy, \
                                 tileB[0]+dx, tileB[1]+dy)
            return self.move()
```

```
    def rotate(self):
        tileA = self.position.tileA
        tileB = self.position.tileB
        # vector rotation otherclocwise
        rx = -(tileB[1] - tileA[1]) # new OX rot
        ry = -(tileA[0] - tileB[0]) # new OY rot
        self.position = Wall(tileA[0], tileA[1], tileA[0]+rx,
tileA[1]+ry)
        return self.move()
```

Задача 1.6.6 (2 балла)

Условие:

Управление беспилотным летательным аппаратом осуществляется с помощью команд, содержащих необходимую скорость по трем осям координат. Значение по каждой из осей задано в м/с и задано целым числом -10 до 10.

При этом, существует несколько источников команд: автопилот аппарата, наземная управляющая станция, а также расположенные на различных участках пилоты. Каждый из источников управления имеет приоритет, выраженный целым числом, причем чем число выше, тем приоритетнее источник. Значение приоритета источника не может меняться. У двух различных источников управления значения приоритета не могут быть одинаковыми.

Команда, содержащая нулевые значения по всем трем осям интерпретируется как отсутствие вмешательства источника управления и снимает все данные им команды (текущей устанавливается последняя наиболее приоритетная неснятая команда). При отсутствии управляющих значений от всех источников аппарат должен зависнуть в воздухе (все скорости устанавливаются в ноль).

Требуется на основе списка последовательно полученных от источников управления команд вычислить итоговую скорость летательного аппарата по трем осям.

Формат входных данных

Первая строка содержит количество последовательно принятых команд от источников управления. Последующие строки содержат управляющие команды. Каждая команда содержит в себе приоритет источника управления (целое число от 0 до 10) и три компоненты скорости. Значения в управляющей команде разделены пробелами.

Формат выходных данных

Три компоненты скорости, которую должен набрать летательный аппарат, разделенные пробелам.

Решение:

```
#include <iostream>

#include <stdio.h>
using namespace std;
int main() {
    int n,p,s;
    s=0;
    cin>>n;
    int A[n][4];
    for (int i=0; i<n; i++)
        {
            cin>>A[i][1]>>A[i][2]>>A[i][3]>>A[i][4];
         }
```

```
          for (int p=10; p>=0; p--)
         {
             for (int i=n; i>=0; i--)
             {
                if (A[i][1] == p)
                {
                     if ((A[i][2] == 0) and (A[i][3] == 0) and
(A[i][4] == 0)) {
                          goto RE1;
                     }
                     else{cout<<A[i][2]<<' '<<A[i][3]<<' '<<A[i][4];
                     s=1;
                     goto RE;}

                     }
          }
          RE1:
          ;
          }
RE:
;
if (s==0) {cout<<0<<' '<<0<<' '<<0;}
return 0;
}
```

Критерии оценки:

Ниже приведен код для автоматической проверки задачи в системе Stepik.

```
#Python3
import random

def solve(dataset):
    command_per_source = {}
    lines = dataset.splitlines()
    lines_count = int(lines.pop(0))
    for line_index in range(lines_count):
    line = lines.pop(0)
    values = line.split()
    priority = int(values[0])
    if values[1:] == ['0', '0', '0']:
          # clear command
          del command_per_source[priority]
    else:
          # set command
          command_per_source[priority] = values[1:]
    if len(command_per_source.keys()) == 0:
    velocity = ['0', '0', '0']
    else:
    velocity = command_per_source[max(command_per_source.keys())]
    return ' '.join(velocity)

def generate_one():
    count = random.randint(10, 30)
    commands = ''
    command_per_source = {}
    for n in range(count):
```

```
    if len(command_per_source.keys()) > 0 and random.randint(0, 10)
> 5:
          priority = random.choice(list(command_per_source.keys()))
          velocity = 0, 0, 0
          del command_per_source[priority]
    else:
          priority = random.randint(0, 10)
          velocity = 0, 0, 0
          while velocity == (0, 0, 0):
                velocity = random.randint(-10, 10), random.randint(-
10, 10), random.randint(-10, 10)
          command_per_source[priority] = velocity
    commands += '%s %s %s %s\n' % (priority, velocity[0],
velocity[1], velocity[2])
    if random.randint(0, 10) > 6:
    # make an intentional 0, 0, 0 answer
    for priority in command_per_source.keys():
          commands += '%s 0 0 0\n' % priority
          count += 1
    commands = '%s\n%s' % (count, commands)
    return commands

def generate():
    result = []
    for n in range(100):
    result.append(generate_one())
    return result

def check(reply, clue):
    return reply == clue
```