§3 Заключительный этап: индивидуальная часть

В финале профиля «Разработка приложений виртуальной и дополненной реальности» участники выполняли задания по математике и информатике, а после – создавали виртуальный урок по физике, где должны были визуализировать, как положительные и отрицательные заряды притягиваются или отталкиваются по закону Кулона.

3.1 Задачи по математике (9 класс)

Задача 3.1.1 (10 баллов)

Условие:

В ряд выписаны целые числа от 1 до 2018. Можно ли между ними расставить знаки «+» и «-» так, чтобы сумма равнялась нулю? Ответ поясните.

Решение:

Нет, среди чисел от 1 до 2018 есть 1009 нечетных, сумма (разность) нечетного числа нечетных чисел – нечетное число. Поэтому нельзя расставить знаки так, чтобы получить четное число 0.

Ответ: Нельзя

Задача 3.1.2 (10 баллов)

Условие:

На кольцевой дороге через равные промежутки расположены 25 постов, на каждом стоит полицейский. Полицейские пронумерованы в каком-то порядке числами от 1 до 25. Требуется, чтобы они перешли по дороге так, чтобы снова на каждом посту был полицейский, но по часовой стрелке за номером 1 стоял номер 2, за номером 2 стоял номер 3, ..., за номером 25 стоял номер 1. Докажите, что если организовать переход так, чтобы суммарное пройденное расстояние было наименьшим, то кто-то из полицейских останется на своём посту.

Решение:

Будем считать, что длина дороги равна 25. Пусть при переходе от исходной расстановки A в некоторую "упорядоченную" расстановку B каждый из полицейских переместился (разумеется, каждый из них двигался по меньшей дуге, соединяющей его исходное положение с новым). Докажем, что суммарное пройденное расстояние можно уменьшить. Не менее 13 полицейских шли на новое место в одном направлении (пусть по часовой стрелке). Рассмотрим расстановку C, получающуюся из B сдвигом на одно место против часовой стрелки. Теперь при переходе из A в C как минимум у 13 полицейских пройденные расстояния уменьшились на 1, а у остальных, если и увеличились, то не больше чем на 1. В результате суммарное расстояние уменьшилось как минимум на 1.

Задача 3.1.3 (10 баллов)

Условие:

Два человека начали одновременно спускаться по движущемуся вниз эскалатору. Первый идет вдвое быстрее, чем второй. Сколько ступенек на эскалаторе, если к концу спуска первый прошел 60 ступенек, а второй – 40?

Решение:

Пусть на эскалаторе x ступенек, скорость первого (относительно неподвижного эскалатора)– a ст./мин., скорость второго - $2a$ ст./мин., скорость эскалатора – b ст./мин. Тогда

$$\begin{cases} \dfrac{ax}{a+b} = 40, \\ \dfrac{2ax}{2a+b} = 60. \end{cases}$$

Разделим первое уравнение на второе, получим $b = 2a$. Подставим в первое уравнение системы $\frac{ax}{a+2a} = 40$, откуда $x = 120$.

Ответ: 120

Задача 3.1.4 (10 баллов)

Условие:

Известно, что $a + b + c = 7, \frac{1}{a+b} + \frac{1}{b+c} + \frac{1}{c+a} = \frac{7}{10}$. *Найдите* $\frac{a}{b+c} + \frac{b}{c+a} + \frac{c}{a+b}$.

Решение:

Рассмотрим выражение

$$\left(1 + \frac{a}{b+c}\right) + \left(1 + \frac{b}{a+c}\right) + \left(1 + \frac{c}{b+a}\right) =$$

$$= \frac{a+b+c}{b+c} + \frac{a+b+c}{a+c} + \frac{a+b+c}{b+a} = (a+b+c)\left(\frac{1}{b+c} + \frac{1}{a+c} + \frac{1}{b+a}\right) = 7 \cdot \frac{7}{10} = 4{,}9.$$

Тогда

$$\frac{a}{b+c} + \frac{b}{a+c} + \frac{c}{b+a} = 4{,}9 - 3 = 1{,}9.$$

Ответ: 1,9

3.2 Задачи по математике (10-11 класс)

Задача 3.2.1 (10 баллов)

Условие:

В ряд выписаны целые числа от 1 до 2018. Можно ли между ними расставить знаки «+» и «-» так, чтобы сумма равнялась нулю? Ответ поясните.

Решение:

Нет, среди чисел от 1 до 2018 есть 1009 нечетных, сумма (разность) нечетного числа нечетных чисел – нечетное число. Поэтому нельзя расставить знаки так, чтобы получить четное число 0.

Ответ: Нельзя

Задача 3.2.2 (10 баллов)

Условие:

Решите уравнение $\sqrt[3]{2-x} = 1 - \sqrt{x-1}$.

Решение:

ОДЗ $= [1; +\infty)$. Возведем обе части в куб, тогда

$$2 - x = 1 - 3\sqrt{x-1} + 3(x-1) - (x-1)\sqrt{x-1}.$$

Сделаем замену переменной $t = \sqrt{x-1}$ и получим уравнение

$$2 - t^2 - 1 = 1 - 3t + 3t^2 - t^3.$$

Далее $t^3 - 4t^2 + 3t = 0, t(t^2 - 4t + 3) = 0$. Корнями последнего уравнения будут 0, 1, 3.

Возвращаясь обратно к переменной x, получаем следующие решения исходного уравнения: 1, 2, 10.

Ответ: 1, 2, 10.

Задача 3.2.3 (10 баллов)

Условие:

Дана возрастающая геометрическая прогрессия b_n. Известно, что $b_4 + b_3 - b_2 - b_1 = 5$. Докажите, что $b_6 + b_5 \geq 20$.

Решение:

Обозначим знаменатель геометрической прогрессии через q. *Тогда* $b_4 + b_3 - b_2 - b_1 = b_1(q^3 + q^2 - q - 1) = b_1(q+1)(q^2-1) = 5$. Откуда $b_1(q+1) = \frac{5}{q^2-1}$ или $b_6 + b_5 = b_1 q^4(q+1) = \frac{5q^4}{q^2-1}$. Заметим, что $(q^2-1)^2 \geq 0$ или $q^4 \geq 4q^2 - 4$, $\frac{q^4}{q^2-1} \geq 4$. Тогда легко получить, что $b_6 + b_5 = 5 \cdot \frac{q^4}{q^2-1} \geq 5 \cdot 4$.

Задача 3.2.4 (10 баллов)

Условие:

Решите в целых числах уравнение $6xy - 4x + 9y - 366 = 0$.

Решение:

$$6xy - 4x + 9y - 366 = 0,$$
$$2x(2 - 3y) = 9y - 366,$$
$$2x = \frac{9y - 366}{2 - 3y} = -3 - \frac{360}{2 - 3y}.$$

Заметим, что число $\frac{360}{2-3y}$должно быть целым. Кроме того, так как левая часть четная, то $\frac{360}{2-3y}$ – нечетное число. В делителях числа $2 - 3y$ должны быть все двойки, делители числа 360, при этом число $2 - 3y$ не кратно 3. Учитывая, что $360 = 2^3 \cdot 3^2 \cdot 5$, получаем, что

$2 - 3y = \pm 2^3$ или $2 - 3y = \pm 2^3 \cdot 5$.

Рассмотрев четыре случая, получаем ответ: $(3; 14), (-24; -2)$.

Ответ: $(3; 14), (-24; -2)$

Критерии оценивания

Баллы	Правильность (ошибочность) решения
10	Полное верное решение.
8-9	Верное решение. Имеются небольшие недочеты, в целом не влияющие на решение.
6-7	Решение в целом верное. Однако оно содержит ряд ошибок, либо не рассмотрение отдельных случаев, но может стать правильным после небольших исправлений или дополнений.
5-4	Верно рассмотрен один из двух (более сложный) существенных случаев.
2-3	Доказаны вспомогательные утверждения, помогающие в решении задачи.
1	Рассмотрены отдельные важные случаи при отсутствии решения (или при ошибочном решении).
0	Решение неверное, продвижения отсутствуют.
0	Решение отсутствует.

3.3 Задачи по информатике

Задача 3.3.1 (100 баллов)

Входной файл:	input.txt	Ограничение времени:	1 сек
Выходной файл:	output.txt	Ограничение памяти:	256 Мб
Максимальный балл:	100		

Условие

Объект в компьютерной игре должен переместиться по горизонтали из точки с x-координатой AA в точку с x-координатой B (B>A)B (B>A) за время, в точности равное TT кадрам анимации.

Пусть за кадр с номером ii объект перемещается вправо на целое число пикселей didi (таким образом, d1+d2+⋯+dT=B−Ad1+d2+⋯+dT=B−A). Для обеспечения плавности анимации ускорение объекта не должно превосходить одного пикселя на кадр за кадр. То есть для любого i>0i>0 должно выполняться |di−di−1|≤1|di−di−1|≤1. Будем считать, что d0=0d0=0.

Напишите программу, которая по данным AA, BB и TT находит подходящий набор didi или определяет, что это сделать невозможно.

Формат входного файла

Входной файл содержит целые числа A B TA B T.

Формат выходного файла

Выходной файл должен содержать TT целых чисел didi. Если существует несколько решений, выведите любое из них. Если решения не существует, выведите единственное число −1−1.

Ограничения

0≤A<B≤1090≤A<B≤109, 1≤T≤100001≤T≤10000

Описание подзадач и системы оценивания

Баллы за первую подзадачу начисляются только в случае, если все тесты этой подзадачи успешно пройдены. Баллы за вторую подзадачу начисляются за каждый тест в отдельности, но только в случае прохождения всех тестов первой подзадачи.

Подзадача	Баллы	Дополнительные ограничения TT	Необходимые подзадачи
1	25	T≤10T≤10	
2	75	T≤10000T≤10000	1

Примеры тестов

№	Входной файл (input.txt)	Выходной файл (output.txt)
1	1 5 3	1 2 1
2	3 10 2	-1

Решение:

```
#include <algorithm>
#include <cmath>
#include <cstdio>
```

```
#include <cstdlib>
#include <functional>
#include <iostream>
#include <map>
#include <queue>
#include <set>
#include <sstream>
#include <stack>
#include <string>
#include <vector>


intmain() {
   std::ios_base::sync_with_stdio(false);
   std::cin.tie(NULL);

    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);

   longlonga, b, t;
   std::cin >> a >> b >> t;

   longlongde = b - a;
   if(t * (t + 1) / 2 < de) {
      std::cout << -1;
      return0;
   }
   longlongam = 1;
   longlongsum = 0;
   std::vector<longlong> ans;
   while(sum + am <= de) {
      sum += am++;
   }
   --am;

   for(inti = 0; i < t - am - (de - sum != 0); ++i) {
      std::cout << 0 << " ";
   }

   for(inti = 0; i < am; ++i) {
      std::cout << i + 1 << " ";
      if(de - sum == i + 1) {
         std::cout << i + 1 << " ";
      }
   }
}
```

Задача 3.3.2 (100 баллов)

Входной файл:	input.txt	Ограничение времени:	1 сек
Выходной файл:	output.txt	Ограничение памяти:	256 Мб
Максимальный балл:	100		

Условие

Вы — начинающий разработчик игрового движка. Не успели освоиться, как заказчик потребовал поддержку VR, к тому же попросил продемонстрировать его работу уже завтра!

Коллективом проектных менеджеров было решено разработать следующую демонстрацию. На игровом поле будут заданы положения NN зарядов, также будут даны значения зарядов в кулонах. Задание заключается в том, чтобы промоделировать перемещение зарядов под действием сил электростатического взаимодействия.

Решено сначала написать двумерный прототип, т.е. каждый заряд имеет две координаты в метрах. Также решено промоделировать заданное число MMшагов, длительностью TT секунд каждый. Чтобы упростить моделирование, предполагается, что в течение каждого шага, действующие на заряды силы не изменяются. Перед каждым шагом необходимо пересчитать действующие на каждый заряд силы согласно закону Кулона. Согласно этому закону сила взаимодействия каждой пары зарядов по модулю будет равна F1,2=k|q1|·|q2|r21,2F1,2=k|q1|·|q2|r1,22. Сила будет направлена в сторону взаимодействующего заряда, в случае, если его значение противоположно по знаку, и в обратную сторону — иначе. Для простоты коэффициент kk необходимо взять численно равным 11. Массу всех зарядов считать одинаковой, равной 11 кг.

Предлагается также использовать уравнение движения x=x0+vx0·T+ax·T22x=x0+vx0·T+ax·T22, где x0,vx0x0,vx0 — положение и скорость в момент времени перед шагом (аналогичные параметры для координаты yy), axax — компонента xx ускорения, которую можно вычислить из второго закона Ньютона m·ax=Fxm·ax=Fx, где FxFx — компонента xx силы кулоновского взаимодействия (аналогично с компонентой yy). Время TT — это длительность шага.

Все заряды могут проходить сквозь друг в друга. Чтобы избежать деления на ноль при вычислении закона Кулона, в знаменателе при вычислении решено брать максимум из 0.10.1 и r2r2.

Осталось дело за малым — написать код!

Отправка решения и тестирование

Все тесты к данной задаче доступны в одном файле. Этот файл можно скачать ЗДЕСЬ.

Формат входного файла

Первая строка входного файла содержит 2 целых числа NN, MM и 1 вещественное число TT — количество зарядов, количество шагов, и длительность каждого шага моделирования соответственно.

В каждой из NN последующих строк заданы 3 вещественных числа xi,yi,qixi,yi,qi, разделенных пробелом — координаты и значение ii-го заряда соответственно.

Формат выходного файла

В выходной файл выведите MM раз по NN строк — результаты моделирования после каждого шага. В каждой строке по две координаты xi,jxi,j и yi,jyi,j jj-го заряда в момент времени после ii-го шага. Координаты будут считаться верными, если либо абсолютная, либо относительная погрешность не превосходит 10−210−2 (второй знак после запятой).

Примеры тестов

№	Входной файл (input.txt)	Выходной файл (output.txt)
1	2 10 0.1 0 0 1 1 1 -1	0.00176776 0.00176776 0.99823223 0.99823223 0.00708363 0.00708363 0.99291636 0.99291636 0.01599877 0.01599877 0.98400122 0.98400122 0.02861942 0.02861942 0.97138057 0.97138057 0.04511558 0.04511558 0.95488441 0.95488441 0.06573648 0.06573648 0.93426351 0.93426351 0.09083666 0.09083666 0.90916333 0.90916333 0.12092011 0.12092011 0.87907988 0.87907988 0.15671878 0.15671878 0.84328121 0.84328121 0.19934315 0.19934315 0.80065684 0.80065684
2	4 10 0.01 0 0 1 1 1 1 0 1 1 1 0 -1	0.00003232 -0.00006768 1.00006768 0.99996768 -0.00003232 1.00003232 0.99993232 0.00006768 0.00012930 -0.00027069 1.00027069 0.99987070 -0.00012928 1.00012928 0.99972928 0.00027072 0.00029100 -0.00060897 1.00060897 0.99970900 -0.00029084 1.00029084 0.99939087 0.00060913 0.00051753 -0.00108237 1.00108237 0.99948247 -0.00051695 1.00051695 0.99891704 0.00108296 0.00080907 -0.00169068 1.00169068 0.99919093 -0.00080750 1.00080750 0.99830775 0.00169225 0.00116585 -0.00243360 1.00243360 0.99883415 -0.00116236 1.00116236 0.99756291 0.00243709 0.00158815 -0.00331078 1.00331078 0.99841185 -0.00158138 1.00158138

		0.99668245 0.00331755 0.00207632 -0.00432178 1.00432178 0.99792368 -0.00206437 1.00206437 0.99566626 0.00433374 0.00263078 -0.00546610 1.00546610 0.99736922 -0.00261110 1.00261110 0.99451422 0.00548578 0.00325196 -0.00674316 1.00674316 0.99674804 -0.00322131 1.00322131 0.99322619 0.00677381

Решение:

```
#include <iostream>
#include <iomanip>
#include <vector>
#include <cmath>

using namespace std;

#define sqr(x) ((x)*(x))

struct vec {
   long double x;
   long double y;
};

vec operator+ (const vec& a, const vec& b) {
   return {a.x + b.x, a.y + b.y};
}

vec operator* (const vec& a, long double k) {
   return {a.x * k, a.y * k};
}

vec operator* (long double k, const vec& a) {
   return a * k;
}

struct object {
   long double q;
   vec pos = {0, 0};
   vec v = {0, 0};
   vec force = {0, 0};
};
```

```
long double sdist(const vec& a, const vec& b) {
   return sqr(a.x - b.x) + sqr(a.y - b.y);
}

void norm(vec& a) {
   long double l = sqrt(sqr(a.x) + sqr(a.y));
   a.x /= l;
   a.y /= l;
}

long double dist(const vec& a, const vec& b) {
   return sqrt(sqr(a.x - b.x) + sqr(a.y - b.y));
}

// const long double k = 8.9875517873681764e9l;
const long double k = 1;

vec calc_force(const object& a, const object& b) {
   vec res;
   res.x = a.pos.x - b.pos.x;
   res.y = a.pos.y - b.pos.y;

   norm(res);

   return res * (k * a.q * b.q / max(0.1l, sdist(a.pos, b.pos)));
}

void update_object(object& a, long double dt) {
   a.pos = a.pos + a.v * dt + a.force * sqr(dt) * 0.5;
   a.v = a.v + a.force * dt;
   a.force = {0, 0};
}

int main() {
   freopen("input.txt", "r", stdin);
   freopen("output.txt", "w", stdout);

   int n;
   cin >> n;
   int steps;
   cin >> steps;
   long double dt;
   cin >> dt;

   vector<object> objects(n);

   for (auto& obj : objects) {
      cin >> obj.pos.x >> obj.pos.y >> obj.q;
   }

   cout << fixed << setprecision(22);
```

```
    for (int s = 0; s < steps; s++) {
        for (int i = 0; i < objects.size(); i++) {
            for (int j = 0; j < objects.size(); j++) {
                if (i != j) {
                    objects[i].force = objects[i].force + calc_force(objects[i], objects[j]);
}
                }
            }

            for (auto& obj : objects) {
                update_object(obj, dt);
            }

            for (auto& obj : objects) {
                cout << obj.pos.x << " " << obj.pos.y << endl;
}
    }
}
```

Задача 3.3.3 (100 баллов)

Входной файл:	input.txt	Ограничение времени:	1 сек
Выходной файл:	output.txt	Ограничение памяти:	256 Мб
Максимальный балл:	100		

Условие

Лабиринт в компьютерной игре представлен полем из HH строк по WW символов каждая, где каждый символ равен либо '.' (ASCII 46), что означает свободную клетку, либо '#' (ASCII 35), что означает стену.

Игроки появляются в случайной свободной клетке лабиринта и могут за один ход перемещаться на любую из соседних по горизонтали или вертикали свободных клеток. Обозначим перемещения игрока буквами N, S, W, E для направлений на север, юг, запад и восток соответственно. Направление взгляда игрока совпадает с направлением последнего сделанного им хода. Перед первым ходом игрок смотрит на север.

Игра построена на основе трёхмерного графического движка с видом от первого лица. Поэтому в каждый момент времени игрок видит только содержимое клетки непосредственно перед ним в направлении его взгляда, а также двух клеток непосредственно слева и справа от него. Один из игроков опубликовал видео своих перемещений по игровому миру. В результате анализа видео было получено количество шагов NN и символы LiLi FiFi RiRiдля для каждого хода, обозначающие содержимое клеток слева, впереди и справа перед ii-м ходом игрока.

Вокруг лабиринта находится неограниченное поле из пустых клеток, но известно, что игрок никогда не покидал пределы лабиринта.

Напишите программу, которая определит начальную позицию и последовательность ходов игрока или выяснит, что это сделать невозможно.

Формат входного файла

Первая строка входного файла содержит целые числа HWHW. Следующие HH строк содержат по WW символов . и # каждая — описание лабиринта. Строка номер H+2H+2 содержит целое число NN. Следующие NN строк состоят из трёх символов LiLi FiFi RiRi каждая.

Формат выходного файла

Первая строка выходного файла должна содержать два целых числа xx yy — столбец и строку начальной позиции игрока (нумерация начинается с 1). Вторая строка должна состоять из NN символов N, S, W, E — последовательность ходов игрока. Если существует несколько решений, выведите любое из них. Если решения не существует, выведите единственное число −1−1.

Ограничения

1≤W,H≤1001≤W,H≤100, 1≤N≤10001≤N≤1000, 1≤x≤W1≤x≤W, 1≤y≤H1≤y≤H

Примеры тестов

№	Входной файл (input.txt)	Выходной файл (output.txt)
1	4 5 ..... .#.#. .#.#. .###. 6 #.. ... #.. ... #.# ###	5 2 N W W S S
2	1 1 # 1 ##.	-1

Решение:

```
#define _CRT_SECURE_NO_WARNINGS

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <vector>
#include <algorithm>
#include <cmath>
#include <stack>
#include <functional>
#include <set>
#include <queue>
#include <string>
#include <map>
#include <sstream>
#include <cctype>
#include <assert.h>
#define sqr(x) ((x)*(x))
using namespace std;

char mp[105][105];
```

```
string step[1005];
bool dp[1005][105][105];
int W, H;

struct from
{
   int x, y;
   char dir;
};

from arr[1005][105][105];

string look(int x, int y, char dir)
{
   string res = "";
   if (dir == 'N')
   {
                  res += mp[x][y - 1];
                  res += mp[x - 1][y];
                  res += mp[x][y + 1];
   }
   if (dir == 'W')
   {
                  res += mp[x + 1][y];
                  res += mp[x][y - 1];
                  res += mp[x - 1][y];
   }
   if (dir == 'S')
   {
                  res += mp[x][y + 1];
                  res += mp[x + 1][y];
                  res += mp[x][y - 1];
   }
   if (dir == 'E')
   {
                  res += mp[x - 1][y];
                  res += mp[x][y + 1];
                  res += mp[x + 1][y];
   }
   return res;
}

bool check(int x, int y)
{
   return x >= 1 && y >= 1 && x <= H && y <= W && mp[x][y] != '#';
}

int main()
{
   freopen("input.txt", "r", stdin);
   freopen("output.txt", "w", stdout);

   cin >> H >> W;
   for (int i = 0; i <= H + 1; i++)
                  for (int j = 0; j <= W + 1; j++)
                                 mp[i][j] = '.';

   for (int i = 1; i <= H; i++)
                  for (int j = 1; j <= W; j++)
```

```
                cin >> mp[i][j];
        int N;
        cin >> N;
        for (int i = 0; i < N; i++)
                cin >> step[i];

        for (int i = 1; i <= H; i++)
                for (int j = 1; j <= W; j++)
                        if (mp[i][j] == '.' && look(i, j, 'N') == step[0])
                                dp[0][i][j] = 1;

        int dx[4] = { 0, 0, -1, 1 };
        int dy[4] = { 1, -1, 0, 0 };
        char dir[4] = { 'E', 'W', 'N', 'S'};
        for (int q = 0; q < N - 1; q++)
                for (int i = 1; i <= H; i++)
                        for (int j = 1; j <= W; j++)
                                for (int z = 0; z < 4; z++)
                                        if (dp[q][i][j] && check(i + dx[z], j + dy[z]) && look(i + dx[z], j +
dy[z], dir[z]) == step[q + 1])
                                        {
                                                dp[q + 1][i + dx[z]][j + dy[z]] = 1;
                                                arr[q + 1][i + dx[z]][j + dy[z]].dir = dir[z];
                                                arr[q + 1][i + dx[z]][j + dy[z]].x = i;
                                                arr[q + 1][i + dx[z]][j + dy[z]].y = j;
                                        }
        vector <char> way;
        N--;
        for (int i = 1; i <= H; i++)
                for (int j = 1; j <= W; j++)
                {
                        if (dp[N][i][j])
                        {
                                int x = i;
                                int y = j;
                                while (N > 0)
                                {
                                        way.push_back(arr[N][x][y].dir);
                                        int xx = x;
                                        x = arr[N][xx][y].x;
                                        y = arr[N][xx][y].y;
                                        N--;
                                }
                                cout << y << " " << x << endl;
                                reverse(way.begin(), way.end());
                                for (int i = 0; i < way.size(); i++)
                                        cout << way[i] << endl;
                                return 0;
                        }
                }
        cout << -1;

}
```