**Муниципальный этап всероссийской олимпиады школьников по информатике 2014-2015 уч.год**

**9-11 классы**

**Задача 1. Многоугольник (100 баллов)**

Выпуклый многоугольник задан последовательностью координат своих вершин в порядке обхода: (x1, y1), (x2, y2), (x3, y3),  . . .  , (xn, yn). Вычислить площадь многоугольника.

Формат входных данных

В первой строке вводится количество вершин многоугольника N.

В следующих N строках вводятся координаты вершин многоугольника.

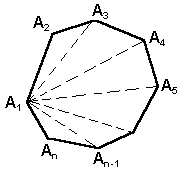
Формат выходных данных

Выводится одно число – площадь многоугольника.

Примеры входных и выходных файлов

|  |  |
| --- | --- |
| input1.txt | output1.txt |
| **5**  **3 2**  **2 5**  **4 7**  **8 5**  **7 1** | **23,5** |

**Комментарии к решению задачи 1.**

Стандартный способ вычисления площади выпуклого многоугольника - разбиение исходного многоугольника на отдельные треугольники с последующим вычислением площадей полученных треугольников и их суммированием.

Площадь отдельного треугольника можно вычислить, например, по формуле Герона, но в данном случае более удобной будет формула расчета площади треугольника по координатам его вершин:

pltrg

Пусть n - число вершин, X(n), Y(n) - массивы, содержащие координаты вершин, тогда основная часть программы для вычисления площади многоугольника будет иметь вид:

s:=0;

for i:=3 to n do

s:=s+0.5\*abs((x[i-1]-x[1])\*(y[i]-y[1])-(x[i]-x[1])\*(y[i-1]-y[1]));

writeln('Площадь многоугольника s=',s);

**Тесты для проверки (4 тестов Х 25 баллов = 100 баллов)**

|  |  |
| --- | --- |
| input1.txt | output1.txt |
| **5**  **3 2**  **2 5**  **4 7**  **8 5**  **7 1** | **23,5** |
| **3**  **2 2**  **4 7**  **9 4** | **15,5** |
| **4**  **1 1**  **3 7**  **8 8**  **10 3** | **35,1** |
| **5**  **2 2**  **3 10**  **6 11**  **11 9**  **14 4** | **55,3** |

Задача 2. Сумма двух чисел (100 баллов).

Заданы три числа: *a*, *b*, *c*. Необходимо выяснить, можно ли так переставить цифры в числах *a* и *b*, чтобы в сумме получилось *c*.

Формат входных данных

Входной файл содержит три целых числа: *a*, *b*, *c* (0 < *a*, *b*, *c* < 109). Числа разделены пробелом.

Формат выходных данных

Если искомая перестановка цифр возможна, необходимо вывести в выходной файл слово YES, в противном случае — выведите слово NO. При положительном ответе необходимо вывести во второй строке выходного файла число *x*, получаемое перестановкой цифр числа *a*, и число *y*, получаемое перестановкой цифр числа *b*, сумма которых равна *c*. Числа *x* и *y* не должны содержать ведущих нулей. Числа в строке разделены пробелом.

Примеры входных и выходных файлов

|  |  |
| --- | --- |
| Input2.txt | output2.txt |
| 12 31 25 | YES  12 13 |
| 12 31 26 | NO |

**Комментарии к решению задачи 2.**

Основная идея решения данной задачи основана на переборе всех перестановок цифр числа *a*. Чтобы это сделать, обозначим полученное число *aperm*. Теперь, для того, чтобы найти число *bperm*, которое необходимо получить перестановкой цифр числа *b*, достаточно вычесть из числа *c* число *aperm*. Для проверки возможности перестановки цифр числа *b* таким образом, чтобы получилось число *bperm,* предлагается проверить на равенство мультимножества цифр указанных чисел. Это можно сделать, например, посчитав количество нулей, единиц, двоек … девяток в каждом из сравниваемых чисел.

При оценивании решений участников рекомендуется учитывать частичное решение, которое перебирает перестановки обоих чисел (*a* и *b*).

**Тесты для проверки. (5 тестов Х 20 баллов = 100 баллов)**

|  |  |
| --- | --- |
| input2.txt | output2.txt |
| 39 14 107 | YES  93 14 |
| 0 15 51 | YES  0 51 |
| 0 15 15 | YES  0 15 |
| 15 15 30 | YES  15 15 |
| 15 15 32 | NO |

Задача 3. Детский праздник (100 баллов).

Организаторы детского праздника планируют надуть для него  воздушных шариков. С этой целью они пригласили  добровольных помощников, -й среди которых надувает шарик за  минут, однако каждый раз после надувания  шариков устает и отдыхает  минут. Теперь организаторы праздника хотят узнать, через какое время будут надуты все шарики при наиболее оптимальной работе помощников, и сколько шариков надует каждый из них. (Если помощник надул шарик, и должен отдохнуть, но больше шариков ему надувать не придется, то считается, что он закончил работу сразу после окончания надувания последнего шарика, а не после отдыха).

Формат входных данных

На первой строке входного файла находятся числа  и  (, ). Следующие  строк содержат по три целых числа - ,  и  соответственно (, ).

Формат выходных данных

Выведите в выходной файл на первой строке число  - время, за которое будут надуты все шарики. На второй строке выведите  чисел – сколько шариков надует каждый из приглашенных помощников. Разделяйте числа пробелами. Если распределений шариков несколько, выведите любое из них.

Примеры

|  |  |
| --- | --- |
| input3.txt | output3.txt |
| 10 3  1 2 3  3 10 3  2 4 3 | 8  4 2 4 |
| 1 3  1 1 100  2 1 100  3 1 100 | 1  1 0 0 |

**Комментарии к решению задачи 3.**

Для решения этой задачи можно применить жадный алгоритм – будем добавлять шарики по очереди и давать шарик для надувания тому помощнику, который закончит этот процесс раньше всех. Действительно – предположим, что на каком-то этапе быстрее всех надует шарик помощник X, но выгодно дать шарик другому помощнику – Y. Тогда, поскольку все шарики одинаковые, X больше вообще не следует надувать шариков. Но тогда, отобрав шарик у Y и передав его X, получим, что максимум времени окончания работы среди этих двух помощников уменьшится, а значит общее время работы не увеличится.

**Примерный текст программы:**

program children\_party;

var

i, j, k, m, n, min, time: longint;

done, next, t, y, z: array [0..100] of longint;

begin

assign(input, 'input.txt');

reset(input);

assign(output, 'output.txt');

rewrite(output);

{ Ввод данных из файла }

read(m, n);

for i := 1 to n do

begin

read(t[i], z[i], y[i]);

{ next – массив содержит момент окончания надувания помощником

следущего шарика }

next[i] := t[i];

end;

{ done – массив содержит количество шариков, надутых помощником }

fillchar(done, sizeof(done), 0);

{ Время работы }

time := 0;

for i := 1 to m do

begin

{ Ищем того, кто закончит раньше }

min := maxlongint;

for j := 1 to n do

if next[j] < min then

begin

k := j;

min := next[j];

end;

{ Обновляем время работы }

time := next[k];

{ Добавляем помощнику шарик }

inc(done[k]);

{ Обновляем время окончания им работы }

next[k] := next[k] + t[k];

{ Учитываем отдых }

if done[k] mod z[k] = 0 then

next[k] := next[k] + y[k];

end;

{ Выводим ответ }

writeln(time);

for i := 1 to n do

write(done[i], ' ');

end.

**Тесты для проверки (4 тестов Х 25 баллов = 100 баллов):**

|  |  |
| --- | --- |
| input3.txt | output3.txt |
| 1 1  1 1 3 | 1  1 |
| 1 1  1 1 101 | 0 |
| 1 25  1 1 3 | 0 |
| 1 2  1 1 3  0 0 0 | 5  2 0 |

**Задача 4 Такси (100 баллов).**

На конференцию приехало N человек. Для их перевоза выделили автомобили вместимостью К и М человек (без водителя). К гостинице автомобили подаются в таком порядке: сначалавместимостью К человек, потом - М человек, после этого опять - К человек, потом М человек и так далее. В автомобиле можно перевозить не больше максимально допустимого количества пассажиров.

Определите, сколько всего необходимо автомобилей для перевозки всех делегатов конференции.

**Входные данные**

Входные данные содержат содержит три числа: N - количество делегатов, К и М – вместительность автомобилей.

**Выходные данные**

Выводиться одно число - количество автомобилей, необходимых для перевозки всех делегатов.

**Примеры входных и выходных файлов:**

|  |  |
| --- | --- |
| input4.txt | output4.txt |
| 20 3 4 | 6 |

**Комментарии к решению задачи 4.**

Пусть T – количество автомобилей. Делим целочисленно q= N\(К+М). Определяем разность p=(N - (К+М)\*q). Если p<=K, то T=2\*q-1, иначе T=2\*q.

**Тесты для проверки (4 тестов Х 25 баллов = 100 баллов):**

|  |  |
| --- | --- |
| input4.txt | output4.txt |
| 20 3 4 | 6 |
|  |  |
| 25 5 5 | 5 |
|  |  |
| 12 2 2 | 6 |
|  |  |
| 100 2 1 | 67 |