10 Класс

Максимальная продолжительность – 210 мин.
Максимально возможное количество баллов – 500

Задача 1 (Время – 1 сек., память – 16 Мб, 100 баллов)

У некоторых школьников возникают затруднения при переводе единиц измерения информации. Ученик 10 класса решил написать им в помощь программу, которая по указанному информационному объёму определяет его значение в битах. Помогите ученику написать такую программу.

Входные данные: *с*трока, состоящая из двух значений, разделённых одним пробелом. Первое значение A ($0 < A \leq 105$) – целое число – количество информации, второе – текстовое – единица измерения информации. Возможные обозначения единиц B (байт), KiB (килобайт), MiB (мегабайт), GiB (гигабайт), PiB (петабайт).

Выходные данные: одно число - количество информации в битах.

Пример работы программы:

ввод	**вывод**
12 MiB	100663296

Решение.

Предлагаемая задача на простую обработку строки текста, преобразование типов данных, знание единиц измерения информации.

//Pascal

```
program solve01;
Uses sysutils, Math;
Var s: string; a,k: integer;
begin
readln(s);
k := pos(' ',s);
a := StrToInt(copy(s, 1, k-1));
s := copy(s, k+1, length(s)-k);
a := a * 8;
if s = 'B' then write(a);
if s = 'KiB' then write(a*power(2,10):0:0);
if s = 'MiB' then write(a*power(2,20):0:0);
if s = 'GiB' then write(a*power(2,30):0:0);
if s = 'PiB' then write(a*power(2,40):0:0);
end.
```

#Python.

```
a = open ('VINF.IN')
s,b = a.readline().strip().split()
a.close()
s = int(s)
if b == 'B':
s = s*2**13
elif b == 'MiB':
s = s*2**23
elif b == 'GiB':
s = s*2**33
elif b == 'PiB':
s = s*2**43
```

```
a = open('VINF.OUT')
a.write(str(s))
a.close()
```

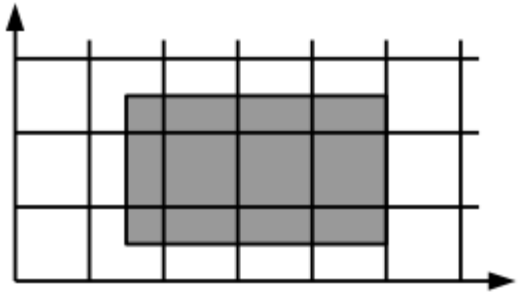
Критерии оценивания: за программу без ошибок начисляется 100 баллов, программа содержит несущественные ошибки, не влияющие на правильность решения – 50 баллов, в ином случае – 0 баллов.

Задача 2 (Время – 1 сек., память – 16 Мб, 100 баллов)

Стена покрыта квадратной плиткой со стороной M см. На стену повесили картину, известны координаты (X и Y) левого нижнего угла картины, её ширина и высота (W и H). Определите количество плиток, которые оказались частично или полностью закрыты картиной.

Входные данные: M — сторону плитки,. X и Y — координаты левого нижнего угла картины, W и H — ширина и высота картины. Ось OX направлена вправо, ось OY направлена вверх. Все числа целые, не превосходящие 2×10^9, числа M, W, H — положительные, числа X и Y — положительные или равны 0.

Выходные данные: количество плиток, полностью или частично закрытых картиной. Плитка считается закрытой картиной, если пересечение картины и плитки имеет ненулевую площадь, то есть касание картины и плитки не считается перекрытием.



Решение, правильно работающее только для случаев, когда все входные числа не превосходят 100, будет оцениваться в 40 баллов. Решение, правильно работающее только для случаев, когда все входные числа не превосходят 10^5, будет оцениваться в 70 баллов.

Решение:

Разобьём плоскость на плитки (квадраты со стороной M) и пронумеруем столбцы и строки квадратов начиная с 0. Сначала определим, в какой столбец плиток попадёт левая сторона картины, в какой столбец попадёт правая сторона картины, в какой ряд плиток попадёт нижняя сторона картины, в какой ряд плиток попадёт верхняя сторона картины. Для этого нужно координаты сторон плиток поделить на M. Левая сторона картины имеет координату X, правая — $X + W$, нижняя — Y, верхняя — $Y + H$. При этом при определении номера столбца левой стороны и номера строки нижней стороны нужно делить на M с округлением вниз, а при определении номера столбца правой стороны и номера строки верхней стороны нужно делить на M с округлением вверх. Затем определяем, количество столбцов и строк, которое занимает картина и перемножаем два полученных числа. Отметим, что в языках Pascal, C++, Java для получения полного балла необходимо проводить вычисления с использованием 64-битных целых чисел, т. к. при использовании обычных целых чисел происходит переполнение целочисленной переменной при вычислении.

Пример решения:

```
#Python
M = int(input())
X = int(input())
Y = int(input())
W = int(input())
H = int(input())
left = X // M
right = (X + W - 1) // M
bottom = Y // M
top = (Y + H - 1) // M
print((right - left + 1) * (top - bottom + 1))
```

Критерии оценивания: решение, правильно работающее только для случаев, когда все входные числа не превосходят 100, будет оцениваться в 40 баллов. Решение, правильно работающее только для случаев, когда все входные числа не превосходят 10^5, будет оцениваться в 70 баллов. Правильное решение - 100 баллов

Задача 3 (Время – 1 сек., память – 16 Мб, 100 баллов)

Требуется написать программу, определяющую наименьшее общее кратное (НОК) чисел a и b.

Входные данные: два натуральных числа A и B через пробел, не превышающих 46340.

Выходные данные: целое число — НОК чисел A и B.

Примеры работы программы:

№	*ввод*	*вывод*
1	36 27	108
2	39 65	195

Для нахождения НОК удобно использовать следующее свойство: для любых натуральных чисел a и b верно равенство НОД(a,b)*НОК(a,b)=a*b , откуда получаем, что НОК(a,b)=a*b/НОД(a,b).

В условиях данной задачи можно НОД найти перебором, но более универсально использовать алгоритм Евклида:

Примеры решения:

```
//Pascal
read(a,b);
a1:=a; b1:=b;
while a*b > 0 do
  if a >= b then a = a mod b else b = b mod a;
write(a1*b2/(a+b));
```

//C++:

```
#include <stdio.h>
#include <iostream.h>
```

```
int a,b;

int main(){
  cin >> a >> b;
  while(b) b^=a^=b^=a%=b;
  cout << a;
  return 0;
```

Критерии оценивания: программа без ошибок, используется эффективный алгоритм - начисляется 100 баллов; программа выдает верный результат, используется неэффективный алгоритм – 50 баллов, в ином случае – 0 баллов.

Задача 4 (Время – 1 сек., память – 16 Мб, 100 баллов)

Известно, что некоторый автомобиль начал свое движение со скоростью v_0. После чего он двигался t_1 секунд с ускорением a_1, а затем – t_2 секунд с ускорением a_2. Необходимо найти максимальную скорость автомобиля на описанном выше промежутке его движения

Входные данные: целые числа v_0, t_1, a_1, t_2, a_2 ($0 <= v_0 <= 10^6$, $0 < t_1, t_2 <= 10^6$, $|a_1| <= 10^6$, $|a_2| <= 10^6$). Гарантируется, что на данном промежутке движения автомобиль не двигался в обратном направлении.

Выходные данные: число – ответ на задачу.

Пример работы программы:

№	ввод	вывод
1	1 2 3 4 5	27
2	1 2 3 4 -1	7

Решение:

$\mathbf{v_0}$ – скорость в начале пути,
$\mathbf{t_1}$, $\mathbf{a_1}$ – время и ускорение при первом равноускоренном движении,
$\mathbf{t_2}$, $\mathbf{a_2}$ – время и ускорение при втором равноускоренном движении.
Известна формула из основ механики, которая позволяет вычислять скорость $\mathbf{v_t}$ после равноускоренного движения материальной точки по прямой с ускорением **a** в течении времени **t** и при начальной скорости $\mathbf{v_0}$: $\mathbf{v_t = v_0 + a * t}$.

Поскольку при равноускоренном движении скорость меняется равномерно, то максимальное и минимальное ее значение достигается в одной из двух точек: начале или конце пути. Поэтому нам достаточно рассмотреть скорость в начале пути (v_0), после первого ускорения ($v_0 + a_1 \times t_1$) и после второго ускорения ($v_0 + a_1 \times t_1 + a_2 \times t_2$). Ответом будет служить максимальное среди этих значений. Алгоритмическая реализация:

```
read(v, t1, a1, t2, a2)
write(max(v, v + t1*a1, v + t1*a1 + t2*a2))
```

В силу ограничений на значения времен и ускорений здесь наиболее эффективно использовать 8-байтовый целый тип (long long в C++ или int64 в Pascal).

Примеры решений:

```
//C++11
#include <bits/stdc++.h>
```

```
using namespace std;
int main(){
long long v0,t1,a1,t2,a2;
cin >> v0 >> t1 >> a1 >> t2 >> a2;
cout << max({v0, v0 + a1*t1, v0 + a1*t1 + a2*t2});
return 0;
}
```

```
//Free Pascal
uses Math;
var v0, t1, t2, a1, a2 : int64;
begin
read(v0, t1, a1, t2, a2);
write(max(v0, max(v0 + t1*a1, v0 + t1*a1 + t2*a2)))
end.
```

```
#Python 3
v0, t1, a1, t2, a2 = map(int, input().split())
print(max(v0, v0 + t1*a1, v0 + t1*a1 + t2*a2))
```

Программа работает верно – 100 баллов; программа содержит ошибки, но ход рассуждений верный – 20 баллов, в ином случае – 0 баллов.

Задача 5 (Время – 1 сек., память – 16 Мб, 100 баллов)

Напишите программу, которая генерирует изображение размером N×N по следующему правилу: квадратную область заполняется цифрами в порядке возрастания, «змейкой», начиная с нижнего левого угла вверх (вправо-вверх-влево-вверх и т.д.). После достижения последней цифры она начинается заново с 0. И так до тех пора, пока не получалась квадратная область со стороной *N* клеток.

```
5432
8901
7654
0123
```

Например, для *N = 4* получался рисунок, приведённый на изображении.

Входные данные: число N (0 < N < 1000), – сторона квадрата изображения.

Выходные данные: сгенерированное изображение в виде матрицы N строк по N цифр без пробелов.

Пример работы программы:

ввод	***вывод***
3	678 543 012
4	5432 8901 7654 0123

Решение.

Предлагаемая задача на моделирование процессов согласно описанному алгоритму, либо поиск закономерностей.

Пример решения:

//Pascal.

```
program solve03;
var i, j, n, k, p, t: integer;
a: array of array of byte;
begin
read(n);
```

//Создадим динамический массив для хранения данных

```
setLength(a, n); k := 0; t := 0; p := 1;
for i := 0 to n-1 do setLength(a[i], n);
```

//Смоделируем заполнение в обратном направлении, отразив результат по вертикали

```
for i := 0 to n-1 do beginfor j := 0 to n-1 do begin
a[i,k] := t;
inc(t); if t = 10 then t := 0;
k := k + p;
end;
```

//Чередование направления заполнения массива

```
p := -1 * p;
k := k + p;
end;
```

//Осуществим вывод в файл в правильном направлении

```
for i := n-1 downto 0 do begin
for j := 0 to n-1 do write(a[i,j]);
writeln;
end;
end.
```

Программа работает верно – 100 баллов; программа содержит ошибки, но ход рассуждений верный – 30 баллов, в ином случае – 0 баллов.