ГЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

2
6904

1	2	3	4	Σ
6	10	13	∅	29

заполняется жюри!

58

ПИСЬМЕННАЯ РАБОТА УЧАСТНИКА ОЛИМПИАДЫ ШКОЛЬНИКОВ СПбГУ 2016–2017

заключительный этап

Предмет (комплекс предметов) Олимпиады ИНФОРМАТИКА (10-11 КЛАССЫ)

Город, в котором проводится Олимпиада Казань

Дата 21.03.2017

****

Вариант 10

****

Задача №1.

На планете «Лютые лютики» имеется свой набор символов для использования в именах жителей. Алфавит состоит из $F > 10$ символов. Имена всех жителей планеты состоят из $D > 4$ букв. Правитель планеты требует, чтобы кто-нибудь смог создать таблицу всех возможных имен, при учете, что никакая буква в имени не повторяется более или равно 4 раза.

Входные данные: Размер алфавита, Алфавит строкой, длина имени

Выходные данные: строки имен (в файл или на экран).

****

Требования к оформлению задач по программированию:

3) Программы должны быть написаны на одном из языков: C, C++, Pascal

4) Полностью оформленная задача должна содержать:

- *программу, выполняющую необходимые операции для всех допустимых данных;*
- *операции с файлами входных и выходных данных **или** понятный пользователю интерфейс ввода исходных данных;*
- *комментарии к тексту программы, облегчающие ее понимание.*

Невыполнение вышеуказанных требований влечет за собой снижение получаемых за задачи баллов

Задача №4.

На каждой клетке игрового поля размером $M \times N$ клеток можно поместить либо дозорную башню с лучниками, либо стену, либо оставить его пустым. Лучники на башне, находящейся по координатам (x, y), могут атаковать другие башни по координатам (x_{-2}, y_{-2}), (x_{-2}, y_2), (x_2, y_{-2}) и (x_2, y_2), в том случае, когда они не разделены стеной.

На вход подается карта, на который отмечены свободные поля и поля с размещенными на них объектами. Ваша задача – разместить на карте как можно больше дополнительных башен, лучники на которых ***не будут*** атаковать друг друга.

Входные данные:

На вход может быть подано несколько тестов, каждый из которых имеет следующую структуру:

- Первая строка содержит два положительных целых числа M и N, разделенных пробелом
- Следующие M строк содержат N символов без пробелов, описывающих игровое поле, где
  - F – свободное поле
  - G – поле с башней
  - P – поле со стеной

Ввод данных считается законченным, когда M и N равны 0.

Выходные данные:

Целые числа, соответствующие количеству дополнительных башен.

Пример:

Ввод:	***Вывод:***
3 4 *FPFP* *PFPP* *GFGF* 5 3 *FPF* *FFF* *FGG* *PFP* *FPF* **0 0**	3 6

****

Требования к оформлению задач по программированию:

1) Программы должны быть написаны на одном из языков: C, C++, Pascal

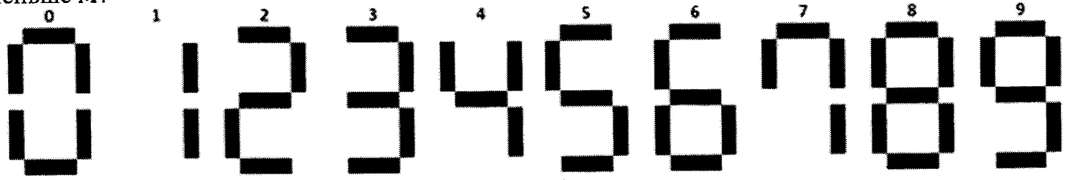
2) Полностью оформленная задача должна содержать:

- *программу, выполняющую необходимые операции для всех допустимых данных;*
- *операции с файлами входных и выходных данных **или** понятный пользователю интерфейс ввода исходных данных;*
- *комментарии к тексту программы, облегчающие ее понимание.*

Невыполнение вышеуказанных требований влечет за собой снижение получаемых за задачи баллов

Задача №2.

Цифровое табло, состоящее из семи-сегментных дисплеев, используется для вывода числовых значений. Вы можете ***деактивировать уже включенные*** сегменты дисплея, но ***включать*** выключенные сегменты не можете. Необходимо написать программу или алгоритм на языках C, C++, Pascal для определения минимального числа, которое может быть выведено на дисплей и не будет меньше ***M***.



Входные данные:

Целое число и ограничение M. Количество дисплеев равно количеству цифр во введенном числе.

Выходные данные:

Целое число. Количество цифр в конечном варианте должно быть равно количеству цифр в начальном.

Пример:

Ввод: 86 10

Вывод: 15

****

Требования к оформлению задач по программированию:

5) *Программы должны быть написаны на одном из языков: C, C++, Pascal*
6) *Полностью оформленная задача должна содержать:*
   - *программу, выполняющую необходимые операции для всех допустимых данных;*
   - *операции с файлами входных и выходных данных **или** понятный пользователю интерфейс ввода исходных данных;*
   - *комментарии к тексту программы, облегчающие ее понимание.*

Невыполнение вышеуказанных требований влечет за собой снижение получаемых за задачи баллов

Задача №3.

Существует такая комната прямоугольной формы, с целочисленными длиной **l** и шириной **w**, для которых верно равенство **l = Aw + B**, где **A** и **B** – целочисленные постоянные. Количество единиц площади этой комнаты таково, что если прибавить к нему **C**, то результат будет делиться без остатка на простое число **P**. Найдите все возможные значения **w**.

Входные данные:

Первая строка содержит количество тестовых случаев, каждая следующая – тестовый случай с разделенными пробелом значениями **A**, **B**, **C** и **P**.

Выходные данные:

Для каждого случая с новой строки выводится результат решения, который содержит общее количество решений и полученные значения ширины в порядке возрастания. Все значения разделяются пробелом

Ограничения:

- $2 \le P < 10^6$
- $0 < A < P$
- $0 \le B$
- $C < P$

Пример:

Ввод:

```
2
1102
1223
```

Вывод:

```
201
0
```

****

Требования к оформлению задач по программированию:

7) *Программы должны быть написаны на одном из языков: C, C++, Pascal*
8) *Полностью оформленная задача должна содержать:*
   - *программу, выполняющую необходимые операции для всех допустимых данных;*
   - *операции с файлами входных и выходных данных **или** понятный пользователю интерфейс ввода исходных данных;*
   - *комментарии к тексту программы, облегчающие ее понимание.*

Невыполнение вышеуказанных требований влечет за собой снижение получаемых за задачи баллов

ЧИСТОВИК

№3

Пусть S - площадь комнаты.

$S = l \cdot w = (Aw + B)w = Aw^2 + Bw$

Пусть $S' = (\text{Площадь комнаты} + C \text{ ед.}) \Rightarrow S' = Aw^2 + Bw + C = S'(w)$

Так как по условию S' делится на P без остатка, то P — один из корней многочлена $S'(w)$; обозначим как w_0 другой его корень.

Тогда $S'(x) = A(w-P)(w-w_0) = Aw^2 + w(-P-w_0) + APw_0 =$

$= Aw^2 + Bw + C$

Тогда w_0 — единственное решение (за исключением вырожденных случаев с $l=0$ или $w=0$)

Тогда: $\begin{cases} B = A(-P-w_0) \\ C = APw_0 \end{cases} \Rightarrow \begin{cases} w_0 = -\frac{B}{A} - P \\ w_0 = \frac{C}{AP} \end{cases}$ — решения должны совпадать.

Программа на C++:

```
#include <iostream>

int main(void)
{
    unsigned long long n; //переменная для кол-ва тестов
    std::cin >> n;
    unsigned long long a, b, c, p;
    double w1, w2; //решения первого и второго ур-я системы
    for(unsigned long long counter = 0; counter++; counter < n) //для каждого тестового случая:
        { // ввод значений:
        std::cin >> a;
        std::cin >> b;
        std::cin >> c;
        std::cin >> p;
        // расчет w1 и w2:
        w1 = (-1)*(b/a) - p;
        w2 = c/(a*b);
        int flag = ((a*0 + b*0 + c) == 0); // flag будет содержать 0, если 0 - корень
                                           // многочлена S'(x)
    // вывод количества реш-й и самих реш-й:
```

```
if ((flag==0) && (w1==w2))
  {std::cout <<"2 "<<"0 "<<w1<<std::endl;}
else if ((flag!=0) && (w1!=w2))
  {std::cout <<"0"<<std::endl;}
else if (flag==0)
  {std:: cout <<"1 0"<<std::endl;}
else {std:: cout <<"1 "<< w1 <<std::endl;}
 }
return 0;
}
```

[redacted]

№1 БУДЕМ ВЫВОДИТЬ ВСЕ ВОЗМОЖНЫЕ ИМЕНА В ЛЕКСИКОГРАФИЧЕСКОМ ПОРЯДКЕ, ОТБРАСЫВАЯ ТЕ, В КОТОРЫХ КАКАЯ-ЛИБО БУКВА ПОВТОРЯЕТСЯ более 3 РАЗ.

ПРОГРАММА НА C++11:

```
#include<iostream>
#include <string>
#include <algorithm>
int main (void)
  {
    unsigned long f, d ; // f и d из условия задачи
    std:: string al ; // строка, содержащая алфавит
    std:: cin >> f ;
    std:: cin >> al ;
    std:: cin >> d;
    std::string prev ; // строка, хранящая предыдущее выведенное имя
    unsigned long cnt[] = new [f] int ; //Вспомогательный массив, хранящий на
                                        //[redacted] i-той позиции количество
                                        // i-тых символов алфавита в текущем
                                        // имени
    std::string stt, stp ;//максимальные и минимальные [redacted] в лексикографическом
                          // порядке строки из данного алфавита
  for (unsigned long long l=d+1 ; d++ ; d<= f) // только имена длины l :
   {for (unsigned long long counter=0 ; counter++ ; counter < d)
      { stt. append (al. charAt (0));
        stp. append (al. charAt (f-1));
      }
    [redacted] prev = stt;
    while (prev< stp)
```

№1, продолжение

Чистовик

Санкт-Петербургский государственный университет

```
{ //генерация следующего имени:
    for (unsigned long long pos = l-1; pos --; pos >= 0)
      { prev[pos] = next(prev[pos]); // next(a) возвращает следующий за a
        if (prev[pos+1] == al[0]) break;   символ алфавита или первый,
      }                                    если a - последний.
  // подсчет букв
    for (unsigned long long pos = 0; pos++; pos < l)
      { cnt[find(al.begin(); al.end(); prev[pos])] += 1; }
  // Если нет таких букв, которые встречаются более 3 раз, вывод имени
    for (unsigned long long pos = 1; pos++; pos < l) // проверка как для числа от 3 до l
      { if (find(*cnt; *(cnt+f+1); pos) == *(cnt+f+1))
          { std::cout << prev << std::endl; }
      }
  //обнуление вспомогательного массива
    for (unsigned long long pos = 0; pos++; pos < f)
      { cnt[pos] = 0; }
  }
}
delete[] cnt;
return 0;
}

char next(char a; std::string al)
{ if (find(al.begin(); al.end(); a) == al.end() - 1)
    {return *(al.begin());}
  else return *(find(al.begin(); al.end(); a) + 1);
}
```

№2. Заметим, что из цифры 0 можно получить цифры 0, 1, 7;
из 1 — 1; из 2 — 2; из 3 — 1, 3; из 4 — 1, 4; из 5 — 5;
из 6 — 5, 6; из 7 — 1, 7; из 8 — все другие и 8; из 9 — 1, 3, 5, 7, 9.
Тогда будем перебирать все числа A (A > M), пока не встретим такое, какое можно получить указанными преобразованиями.

программа на C++ 11:

```
#include <iostream>
#include <algorithm>
#include <string>
#include <cstdlib>
int main(void)
{ unsigned long long num, m; // числа из условия
std::cin >> num;
std::cin >> m;
std::string snum(itoa(num)); // строковое представление num
std::string sm (itoa(m));
// создаём sm, начиная с самого младшего;
unsigned long long pos;
for (pos = sm.length()-1; pos--; pos >=0)
{ // если самая маленькая цифра, которую можно получить из текущей, больше ей-
  меняем, если меньше — меняем и переходим к след. разряду
    if (minn(sm[pos]) > sm[pos]) // сравнивать символы вместо цифр можно, т.к.
                                 меньшей цифре соответствует меньший
                                 код символа
      { sm[pos] = minn(sm[pos]); break; }
    else {sm[pos] = minn(sm[pos]);}
}
// все оставшиеся цифры меняем на самые маленькие из возможных
for (; pos--; pos >=0)
  {sm[pos] = minn sm[pos];}
// если в M меньше разрядов, чем в исходном числе, заполняем необходимые
                разряды минимально возм. цифрами
if (sm.length() != snum.length())
  {for (pos = (snum.length() - sm.length()); pos--; pos>=0)
    { snum[pos] = minn(snum[pos]);}
// копируем из sm в snum
for (pos = (snum.length() - sm.length()); pos++; pos < sm.length())
  {snum[pos] = sm[pos - (snum.length() - sm.length())];}
} else snum = sm;
// выводим
std::cout<<snum<< std::endl;
return 0;
}
```

10

№2, продолжение

Листовик


```
char minn (char a)
{ switch (a){
        case "0": return "0";
        case "1": return "1";
        case "2": return "2";
        case "3": return "1";
        case "4": return "1";
        case "5": return "5";
        case "6": return "5";
        case "7": return "1";
        case "8": return "0";
        case "9": return "1";
  };
}
```

№4

Условие некорректно.
Непонятно, что имеется в виду под координатами
$(X_{-2}; Y_{-1})$; $(X_{-2}; Y_2)$; $(X_2; Y_{-2})$; $(X_2; Y_2)$ относительно координат клетки $(X; Y)$