Суходолов Олег Андреевич

Балл: 140

Задача №1

Условие

Петя наблюдает, как строители копают фундамент для нового дома. Землю они вывозят одинаковыми самосвалами. По каким-то своим причинам всю выбранную землю вывозят со стройки в тот же день, в который ее выбрали, то есть даже за одним кубометром все равно вечером приедет самосвал.

Пете известно, сколько времени копали фундамент, какой объем земли вывозили каждый день и сколько кубометров помещается в самосвал. Он задался вопросом: насколько меньше рейсов понадобилось бы совершить, если бы можно было вывозить землю не в тот же самый день, в который ее извлекли?

Напишите программу, которая решит эту задачу за Петю.

В первой строке на вход программы подаются два натуральных числа: количество дней t, в течение которых копали фундамент, $1 < t \leq 10^3$ и объем земли v, который вмещается в один самосвал, $1 < v \leq 10^2$. На следующей строке на вход программы поступает n натуральных чисел: $a_1, a_2, \ldots, a_n$. a_i – сколько земли выбрали в i-й день, $1 < a_i \leq 10^5$.

Выведите одно целое число – разницу между количеством проделанных рейсов и минимальным количеством рейсов, за которые можно вывезти этот объем земли.

Пример

Входные данные	Выходные данные
2 4 6 5	1

Комментарий к примеру. Строители работали два дня, в первый день они извлекли 6 кубометров грунта, во второй – 5. Самосвал принимает 4 кубометра грунта, следовательно, понадобится 4 рейса. Можно было обойтись за три рейса, следовательно, разница – 1.

Исходный код

```
#include <iostream>

int main()
{
 int n;
 int v;
 std::cin >> n >> v;
 int sum_v = 0;
 int sum_am = 0;
 for (int i = 0; i < n; ++i)
 {
  int a;
  std::cin >> a;
  sum_v += a;
  sum_am += a / v + (a % v ? 1 : 0);
 }
 std::cout << sum_am - (sum_v / v + (sum_v % v ? 1 : 0));
 return 0;
}
```

Задача №2

Условие

В кассе, продающей билеты по цене 500 р., нет возможности расплатиться картой, а в начале работы нет даже сдачи. Перед открытием у кассы собрались посетители, у каждого человека в очереди есть только

одна купюра. У части – 500 р., у части – 1000 р. Сколькими способами можно выстроить посетителей в очередь так, чтобы к моменту обслуживания посетителя с купюрой в 1000 рублей у кассира всегда была сдача?

Напишите программу, которая решит эту задачу.

На вход программы подаются два целых неотрицательных числа:

n - количество человек с купюрой 500 р,

k - количество человек с купюрой 1000 р

Каждое число меньше 10.

Выведите одно целое число – сколькими способами можно выстроить очередь.

Пример

Входные данные	Выходные данные
2 1	4

Комментарий к примеру. Пусть есть посетители **A**, **B**, и **C**. У **A** и **B** по 500 рублёвой купюре, а у **C** – купюра в 1000 рублей. Тогда их можно выстроить так:

ABC – можно

ACB – можно

BAC – можно

BCA – можно

CAB – нельзя, т.к. в кассе к началу работы нет купюр на сдачу, а у первого посетителя купюра 1000р.

CBA – нельзя, т.к. в кассе к началу работы нет купюр на сдачу, а у первого посетителя купюра 1000р.

Есть только 4 возможных варианта.

Исходный код

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

//F(i, j) = F(i - 1, j)*(n - i + 1) + (1-can(i, j)) * F(i, j - 1) * (k - j + 1)
bool can(int i, int j)
{
 return i >= j;
}
int main()
{
 long long int n, k;
 std::cin >> n >> k;
 std::vector<std::vector<long long int>> amounts(n + 1, std::vector<long long int>(k + 1, 0));
 amounts[0][0] = 1;

 for (int i = 0; i <= n; ++i)
 {
  if (i == 0)
   continue;
  for(int j = 0 + (n == 0); j <= k; ++j)
  {
   if (j == 0)
   {
    amounts[i][j] = amounts[i - 1][j] * (n - i + 1);
    continue;
   }
   amounts[i][j] = amounts[i - 1][j] * (n - i + 1) + (can(i, j) ? 1 : 0) * amounts[i][j - 1]*(k - j  + 1);
  }
 }
 /*for (auto &it : amounts)
  {
   for (auto &it1 : it)
   std::cout << ' ' << it1;
   std::cout << '\n';
 }*/
 std::cout << amounts[n][k];
```

```
 return 0;
}
```

Задача №3

Условие

Исполнитель получает на вход натуральное число X (не превышающее 10^6). По этому числу, точнее по его представлению в шестеричной системе счисления строится новое число Y по следующим правилам.

В шестеричном представлении числа X предпоследняя цифра увеличивается на 1 (гарантируется, что в шестеричном представлении X числа больше 2-х цифр). Например, $749_{10} = 32\mathbf{4}5_6$ -> $32\mathbf{5}5_6 = 755_{10}$.

Если предпоследняя цифра 5, тогда предпоследняя цифра становится 0, а последняя изменяется по следующему принципу: четная увеличивается н 1, а нечетная уменьшается на 1. Например, последняя цифра нечетная $751_{10} = 32\mathbf{5}\boldsymbol{1}_6$ -> $32\mathbf{0}\boldsymbol{0}_6 = 720_{10}$, последняя цифра четная $752_{10} = 32\mathbf{5}\boldsymbol{2}_6$ -> $32\mathbf{0}\boldsymbol{3}_6 = 723_{10}$.

Введем понятие расстояния

Oh = | Исходное_число – Полученное_число |

Напишите программу, которая будет считать наибольшее расстояние Oh для чисел из заданного интервала [A,B] и наибольшее исходное число, для которого оно было вычислено.

На вход программы подаётся два целых числа A и B ($10 \le A \le B \le 1\,000\,000$), записанных через пробел.

Программа должна вывести два числа наибольшее расстояние Oh и через пробел исходное число, для которого оно было посчитано.

Входные данные	Вывод	Примечание
748 752	31 751	|748 - 754| = 6 ($32\mathbf{4}4_6 – 32\mathbf{5}4_6$) |749 - 755| = 6 ($32\mathbf{4}5_6 – 32\mathbf{5}5_6$) |750 - 721| = 29 ($32\mathbf{50}_6 – 32\mathbf{01}_6$) |751 - 720| = 31 ($32\mathbf{51}_6 – 32\mathbf{00}_6$) |752 - 723| = 29 ($32\mathbf{52}_6 – 32\mathbf{03}_6$)
747 749	6 749	|747 - 753| = 6 ($32\mathbf{4}3_6 – 32\mathbf{5}3_6$) |748 - 754| = 6 ($32\mathbf{4}4_6 – 32\mathbf{5}4_6$) |749 - 755| = 6 ($32\mathbf{4}5_6 – 32\mathbf{5}5_6$)

Исходный код

```
#include <iostream>
#include <vector>
/*void out(std::vector<int> & v)
{
 for (auto &it : v)
  std::cout << it << ' ';
 std::cout << std::endl;
}*/

void reverse(std::vector<int>  vec)
{
 for (int i = 0, j = vec.size() - 1; i < j; ++i, --j)
 {
  int t = vec[i];
  vec[i] = vec[j];
  vec[j] = t;
  //std::swap(vec[i], vec[j]);
 }
}
```

```
std::vector<int> toSixth(int n)
{
 std::vector<int> res;
 while (n > 0)
 {
  res.push_back(n % 6);

  n /= 6;
 }
 //reverse(res);

 return res;
}
std::vector<int> complete(std::vector<int>  num)
{
 if (num[1] == 5)
 {
  num[1] = 0;
  if ( num[0] % 2)
   --num[0];
  else
   ++num[0];
 }
 else
  ++num[1];
 return num;
}
int toInt(std::vector<int> num)
{
 int sum = 0;
 int six_deg = 1;
 for (int i = 0; i < num.size(); ++i)
 {
  sum += num[i] * six_deg;
  six_deg *= 6;
 }
 return sum;
}
int absol(int n)
{
 return n > 0 ? n : -n;
}

int main()
{
 int a, b;
 std::cin >> a >> b;
 int max = 0;
 int num_of_max = 0;
 for (int i = a; i <= b; ++i)
 {

  int deg = absol(i - toInt(complete(toSixth(i))));
  if (deg >= max)
  {
   max = deg;
   num_of_max = i;
  }
 }
 std::cout << max << ' ' << num_of_max;
 //system("pause");
 return 0;
}
```

Задача №4

Условие

Зонд передает данные с орбиты Юпитера во время сильной магнитной бури. Информация передается по каналу связи в виде пакетов. Каждый пакет представляет собой целое положительное число в двоичной системе счисления. Для обнаружения помех последний разряд в пакете подбирают таким образом, чтобы количество единиц в разрядах пакета было четным. В каждом пакете на практике никогда не искажается больше одного разряда.

Напишите программу, которая по распечатке пакетов, записанных в десятичной системе счисления, найдет самое большое значение, прошедшее без искажений. Известно, что как минимум один пакет прошел без искажений.

Формат ввода

В строке вводится сначала целое число ***n*** – количество пакетов (***n ≤ 1000***), затем ***n*** натуральных чисел, все числа отделены друг от друга одним или несколькими пробелами.

Формат вывода

Вывести одно целое число – самое большое значение, прошедшее без искажений.

Пример

Входные данные	Выходные данные
4 1025 496 882 1056	1056

Исходный код

```
#include <iostream>
/*bool evenBytes(long long int n)
{
 bool even = true;
 for (long long int i = 0; i < 64; ++i)
 {

  if (n & (1 << i))
   {
    std::cout << (1 << i) << ' ';
    even = !even;
    //std::cout << 1;
  }
  //else std::cout << 0;
 }
 return even;
}*/
bool evenBytes(int n)
{
 bool even = true;
 for (int i = 0; i < 32; ++i)
 {

  if (n & (1 << i))
   {
    even = !even;
  }
 }
 return even;
}
int main()
{
 int n;
 std::cin >> n;
 long long int max = 0;
 for (int i = 0; i < n; ++i)
 {
  int a;
  std::cin >> a;
  if (evenBytes(a))
  {
   if (max < a)
    max = a;
  }
 }
 std::cout << max;
 //system("pause");
 return 0;
}
```

Задача №5

Условие

Вася придумывает пароль для каждой новой учетной записи, которую он заводит на каком-то из своих устройств. Время от времени он изменяет пароль, дописывая к нему новые символы. Предыдущий пароль никогда не будет началом пароля для новой учетной записи Васи. После того, как Вася заводит новую учетную запись, он перестает менять пароль на старой.

Зная все Васины пароли в хронологическом порядке, напишите программу, которая найдет, какое наибольшее количество раз Вася менял пароль для одной учетной записи.

Формат ввода

В первой строке вводится сначала целое число ***n*** – количество слов (***n ≤ 1000***), затем в ***n*** следующих строках записано по слову. Слова состоят только из строчных латинских букв.

Формат вывода

Вывести одно целое число – какое наибольшее количество раз Вася менял пароль для одной учетной записи.

Примеры

Входные данные	Выходные данные
3 abd abdc bvd	1
4 bcd bcdf bcdfe abc	2

Исходный код

```
#include <iostream>
#include <string>

bool isSubPass(std::string last, std::string new_p)
{
 if (new_p.size() <= last.size())
  return false;
 for (int i = 0; i < last.size(); ++i)
 {
  if (last[i] != new_p[i])
   return false;
 }
 return true;
}
int main()
{
 int n;
 std::cin >> n;
 std::string last_password = "";
 std::string new_password;
 int amount = -1;
 int max_amount = 0;
 for (int i = 0; i < n; ++i)
 {
  std::cin >> new_password;
  if (isSubPass(last_password, new_password))
  {
   ++amount;
   if (max_amount < amount)
    max_amount = amount;
  }
  else
   amount = 0;
  last_password = new_password;
 }
 std::cout << max_amount;
 //system("pause");
 return 0;
}
```

Задача №6

Условие

Исследовательский аппарат на поверхности Марса может выполнять команды «Фотографирование», «Пробное бурение», «Взятие образцов грунта», «Анализ атмосферы». Из-за конструктивных особенностей на аппарат наложен ряд ограничений. Нельзя выполнять команду «Анализ атмосферы» после команды «Пробное бурение». Нельзя выполнить команду «Фотографирование» после команды «Взятие образцов грунта». Команду «Взятие образцов грунта» можно выполнять только следующей после команды «Пробное бурение». Никакую команду, кроме команды «Взятие образцов грунта», нельзя выполнить подряд дважды.

Напишите программу, которая определит, сколько существует выполнимых последовательностей команд длиной n, если до начала выполнения программы аппарат выполнил команду «Пробное бурение».

На вход программе подается натуральное число n ($n \leq 15$) – количество команд.

Вывести целое число – количество выполнимых последовательностей команд длиной n.

Пример

Ввод	Вывод
2	5

Исходный код

Задача №7

Условие

В августе 1944 года радисты из СМЕРШ прослушивают эфир в окрестностях Шиловичского лесного массива. В лесу скрывается две группы немецких диверсантов, каждая из которых несколько раз выходила на связь. Частоты, на которых работали немецкие рации, пронумерованы цифрами от 0 до 9. Для каждой из вражеских групп выписаны частоты их сеансов связи в хронологическом порядке.

В СМЕРШ считают, что часть этих сеансов связи диверсанты провели со своим командованием. Контрразведчики знают, что порядок использования частот для сеансов связи с командованием одинаков для обеих групп.

Напишите программу, которая определит, какое наибольшее количество сеансов связи с командованием противника могло быть у обеих групп.

На вход программе подаются две строки, каждая из которых состоит из цифр от 0 до 9 – выписанные в хронологическом порядке частоты сеансов связи каждой группы диверсантов.

Вывести одно целое число – наибольшее возможное количество сеансов связи обеих групп со своим командованием.

Пример

12345 2135	3

Исходный код

```
#include <iostream>
#include <vector>
#include <string>
int imax(int a, int b)
{
 return a > b ? a : b;
}
int main()
{
 std::string str1, str2;
 std::cin >> str1 >> str2;
 std::vector<std::vector<std::pair<int, int>>> sequences(str1.size());
 std::vector<std::vector<int>> letters_places(10);
 for (int i = 0; i < str2.size(); ++i)
  letters_places[str2[i] - '0'].push_back(i);

 for (int i = 0; i < str1.size(); ++i)
```

```
 {
  if (letters_places[str1[i]-'0'].size() == 0)
   continue;
  for (int j = 0; j < i; ++j)
  {
   for (int k = 0; k < sequences[j].size(); ++k)
   {
    int start = sequences[j][k].second + 1;
    /*for (int u = start; u < str2.size(); ++u)
    {
     if (str2[u] == str1[i])
     {
      if (sequences[i].size() > 0 && sequences[i].back().second == u)
       sequences[i].back().first = imax(sequences[i].back().first, sequences[j][k].first + 1);
      else
       sequences[i].push_back(std::pair<int, int>(sequences[j][k].first + 1, u));
     }
    }*/
    for (int u = 0; u < letters_places[str1[i] - '0'].size(); ++u)
    {
     if (letters_places[str1[i] - '0'][u] < start)
      continue;
     if (sequences[i].size() > 0 && sequences[i].back().second == letters_places[str1[i]-'0'][u])
      sequences[i].back().first = imax(sequences[i].back().first, sequences[j][k].first + 1);
     else
       sequences[i].push_back(std::pair<int, int>(sequences[j][k].first + 1, letters_places[str1[i]-'0'][u]));
    }
   }
  }
  if (sequences[i].size() == 0 || sequences[i][0].second != letters_places[str1[i] - '0'][0])
   sequences[i].push_back(std::pair<int, int>(1, letters_places[str1[i] - '0'][0]));

 }
 int max_seq = 0;
 for (int i = 0; i < sequences.size(); ++i)
 {
  for (int j = 0; j < sequences[i].size(); ++j)
   max_seq = imax(max_seq, sequences[i][j].first);
 }
 std::cout << max_seq;
 //system("pause");
 return 0;
}
```