# ВОРОПАЕВ Иван Алексеевич

## Балл: 114

### Задача №1

#### Условие

Обычно Петя добирается до школы на автобусе. Иногда приходится ехать в битком набитом автобусе, иногда наоборот, в почти пустом. Петя заметил, что он не один, кто предпочитает не сидеть, а стоять. По его подсчётам предпочитает стоять каждый пятый пассажир. Петя составил таблицу, которая показывает, при какой заполненности автобуса сколько пассажиров предпочитают стоять.

| Общее количество пассажиров в автобусе | Количество пассажиров, предпочитающих сидеть | Количество пассажиров, предпочитающих стоять |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 2 | 0 |
| 3 | 2 | 1 |
| 4 | 3 | 1 |
| 5 | 4 | 1 |
| 6 | 5 | 1 |
| 7 | 6 | 1 |
| 8 | 6 | 2 |
| 9 | 7 | 2 |
| … | … | … |

Три пассажира, уже ближе к пяти, поэтому среди них найдётся один, кто предпочтёт стоять.

Петя считает сиденье бесполезным, если на нём никто не сидит. Общей бесполезностью Петя называет суммарное количество остановок, которое пустовало каждое сидение. Например, если от начала поездки до конца, нужно ехать 3 остановки, и каждый раз пустовало каких-то 4 сидения, то общая бесполезность за время поездки равнялось 12. Каждый раз, когда Петя едет в школу, он считает общую бесполезность. Напишите программу, которая делает это вместо Пети.

Сначала на вход программы подаются два числа: количество сидячих мест $s$, $1 < s \leq 10^3$ и количество остановок $n$, $1 < n \leq 10^5$ (на первой пассажиры могут только заходить, на $n$-ой все выходят). Далее на вход программы поступает ещё $n$-1 число: $a_1$, $a_2$, …, $a_{n-1}$. $a_i$ – это на сколько пассажиров увеличилось общее количество пассажиров в автобусе после $i$-ой остановки. $a_i < 0$ означает, что на $i$-ой остановке больше пассажиров вышло, чем вошло. Гарантируется, что число пассажиров в автобусе никогда не оказывается отрицательным и не оказывается больше 1000.

Выведите одно натуральное число – общую бесполезность сидений в автобусе.

Пример

| Входные данные | Выходные данные |
|---|---|
| 10 3 8 -2 | 9 |

Комментарий к примеру. На первой остановке в автобус вошли 8 человек, шестеро из них успешно сели, 4 сидения остались бесполезными. На следующей остановке двое вышли, осталось 6 человек, из которых сидели пятеро, бесполезными оказалось 5 сидений. На третьей остановке все вышли, т.к. это была конечная. Общая бесполезность сидений оказалась равна девяти.

#### Исходный код

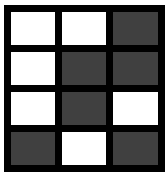
```
#include<bits/stdc++.h>
using namespace std;
int main(){
long long int max,n,sid=0,lisn=0,a,pass=0;
```

```
cin>>max>>n;
for(int i=0;i<n-1;i++)
{
 cin>>a;
 pass=pass+a;

 sid=pass/5;
 if(pass%5>2)
 sid++;

 if(max-(pass-sid)>0)
 lisn=lisn+max-(pass-sid);

}
cout<<lisn;
return 0;
}
```

## Задача №2

### Условие

Петя недавно узнал, что для записи чисел достаточно использовать только нолики и единички. Числа получаются длиннее, зато их можно изображать, закрашивая клеточки тетради, используя только карандаш или только ручку. Петя решил, что он будет закрашивать клеточку, когда встречается цифра один, и оставлять пустой, если цифра ноль. Например, вот такой рисунок Пети



означает число $001\ 011\ 010\ 101_2 = 512 + 128 + 64 + 16 + 4 + 1 = 725$.

Петя считает, что каждый рисунок должен обладать разнообразностью, т.е. не должно быть рисунков, в которых сильно преобладает какой-то один цвет. Лучше, когда пустых и закрашенных клеточек приблизительно поровну. Теперь Петя задался вопросом, сколько разных чисел можно изобразить такими картинками.

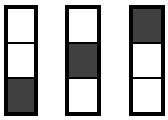
На вход программе подаются три натуральных числа ширина и высота картинки в клеточках, а также количество клеточек, которые должны быть закрашены, чтобы картинки обладали разнообразностью. Общее количество клеточек в изображении не превышает 30, количество клеточек, которые должны быть закрашены меньше общего количества клеточек.

Выведите, сколько разных чисел можно изобразить заданным образом.

Пример

| Входные данные | Выходные данные |
|---|---|
| 1 3 1 | 3 |

Пояснение к примеру. С одной закрашенной клеточкой разных рисунков размера 3 x 1 может быть только три:



### Исходный код

```
#include<bits/stdc++.h>
using namespace std;
int main(){
long long int a,b,c,n,nf=1,c3,Nf=1,kf=1;
cin>>a>>b>>n;
c=a*b;
```

```
c3=a*b;
while(c3!=1)
{
 nf=nf*c3;
 c3--;
// cout<<c1<<' ';
}
//cout<<endl<<nf;
c=c-n;
//cout<<endl;
if(c==0)
{
 cout<<1;
 return 0;
}
while(c!=1)
{
 Nf=Nf*c;
 c--;

}
//cout<<endl<<Nf;

while(n!=1)
{
 kf=kf*n;
 n--;
}
//cout<<endl<<kf;

cout<<nf/kf/Nf;
return 0;
}
```

## Задача №3

### Условие

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

В семеричном представлении числа X зачеркиваются последние две цифры.

1. Если получившееся число нечетное, то в конец полученной цепочки цифр дописывается две цифры – сумма оставшихся цифр (тоже в семеричном представлении, причем если сумма состоит из одной цифры, то перед ней надо вписать 0).

   Например, $2305_{10} = 6502_7$. После зачеркивания остается $65_7$ – нечетное число. Сумма оставшихся цифр $11_{10}=14_7$. Новое число $Y = 6514_7 = 2314_{10}$.

   Другой пример, $841_{10} = 2311_7$, после зачеркивания остается нечетное число $23_7$. Сумма оставшихся цифр $5_{10} = 5_7$. Новое число $Y= 2305_7 = 838_{10}$.

2. Если получившееся после зачеркивания последних двух цифр число четное, то в конец полученной цепочки цифр дописывается одна цифра – количество оставшихся цифр. Например, $2302_{10} = 6466_7$. После зачеркивания остается $64_7$ – четное число. Количество оставшихся цифр $2_{10}=2_7$. Новое число $Y = 642_7 = 324_{10}$.

Напишите программу, которая будет считать для скольких чисел из заданного интервала [A,B] полученное число будет больше исходного.

На вход программы подаётся два целых числа A и B (50<= A<=B<=1 000 000), записанных через пробел.

Программа должна вывести одно целое число - вычисленное значение

### Исходный код

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

```
using namespace std;
int main(){
long long int A,B, a,b,c,d,n,count=0,end,ost,sum=0,ost10,otv10;
string s,s1,s2,s3;
cin>>A>>B;

for(int i=A;i<=B;i++)
{

a=i;

while(a!=0)
{
 b=a%7;
 s1=char(int('0')+b);
 s.insert(0,s1);
 a=a/7;
 s1.clear();
}
//cout<<s<<endl;

end=s.size()-1;
if(s[end-2]%2==1)
{

c=int(s[end])-48;
d=int(s[end-1])-48;
ost=d*10+c;
//cout<<"dve posl="<<ost;

for(int i=0;i<end-1;i++)
{
 sum=sum+int(s[i])-48;
}
//cout<<endl<<"sum cifr="<<sum;

while(sum!=0)
{
 b=sum%7;
 s2=char(int('0')+b);
 s3.insert(0,s2);
 sum=sum/7;
 s2.clear();
}
otv10=atoi(s3.c_str());
//cout<<endl<<"otv10="<<otv10<<endl<<endl;

if(otv10>ost)
{
count++;
//cout<<endl<<"otv10="<<otv10<<endl;
//cout<<"dve posl="<<ost<<endl;

}



}
else
count++;

s1.clear();
s.clear();
s2.clear();
s3.clear();
sum=0;

}
cout<<count;
return 0;
}
```

## Задача №4

### Условие

С клавиатуры вводится несколько слов. Под словом подразумевается последовательность строчных латинских букв. Точное количество слов неизвестно, но может быть очень велико.

Слова можно собирать в цепочку по следующему правилу: у всех слов в цепочке совпадают первая и последняя буквы. Начинать цепочку можно с любого слова. Менять местами слова в последовательности нельзя.

Необходимо определить максимальную длину цепочки, которую можно собрать из введённых слов по этим правилам.

*Формат ввода*

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

*Формат вывода*

Вывести одно целое число – количество слов в самой длинной цепочке.

**Примеры**

| Входные данные | Выходные данные |
|---|---|
| 3<br>a<br>aba<br>c | 2 |
| 4<br>abd<br>abcd<br>abcf<br>abcfde | 2 |

**Исходный код**

```
#include<bits/stdc++.h>
using namespace std;
int main(){
long long int a,b,n,nf=1,c3,Nf=1,kf=1,x,y,max=0;
string s,s1;
char c,d;

map<string,int> mp;

cin>>n;
for(int i=0;i<n;i++)
{
 cin>>s;
 c=s[0];
 b=s.size();
 d=s[b-1];
 s1=s1+c;
 s1=s1+d;

 mp[s1]++;

s1.clear();
}

for(int i=0;i<26;i++)
for(int j=0;j<26;j++)
{
 c=char('a'+i);
 d=char('a'+j);
 s1=s1+c;
 s1=s1+d;
 if(mp[s1]>max)
 max=mp[s1];
 s1.clear();
 }
 cout<<max;

return 0;
}
```

## Задача №5

### Условие

На вход программе подаются натуральные числа. Точное количество чисел неизвестно, но может быть очень велико. Требуется найти среди введенных данных наибольшее число, двоичная запись которого оканчивается тремя нулями. Гарантируется, что хотя бы одно такое число в последовательности есть.

*Формат ввода*

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

*Формат вывода*

Вывести одно целое число – наибольшее число, двоичная запись которого оканчивается тремя нулями.

## Пример

| Входные данные | Выходные данные |
|---|---|
| 5 30 1000 35 6 39 | 1000 |

### Исходный код

```
#include<bits/stdc++.h>
using namespace std;
int main(){
long long int n,max=0,a;
cin>>n;

for(int i=0;i<n;i++)
{
 cin>>a;
 if(a%8==0)
 {
  if(a>max)
  max=a;
 }
}
cout<<max;
return 0;
}
```

## Задача №6

### Условие

В одной из электрических цепей бронетранспортера «Носорог» используются три резистора: первый и второй соединены последовательно, третий подключен к ним параллельно.

На вход программе подаются значения сопротивлений первых двух резисторов и возможные сопротивления третьего. Требуется выбрать среди возможных сопротивлений такое, чтобы общее сопротивление цепи было максимальным.

*Формат ввода*

В первой строке вводятся два положительных числа ***R1*** и ***R2*** – сопротивления первого и второго резистора.

В следующей строке сначала вводится целое положительное число ***n*** – количество возможных сопротивлений. Далее идет n положительных чисел.

*Формат вывода*

Вывести одно число – сопротивление резистора, который надо включить на место третьего.

## Пример

| Входные данные | Выходные данные |
|---|---|

| | |
|---|---|
| 10 10<br><br>2 15 20 | 20 |

**Исходный код**

```
#include<bits/stdc++.h>
using namespace std;
int main(){
long double a,b,c,n,bb,bbb;
long double max=0;
string s,sm,tmp,musor;
//cin>>a>>b;
int start;

//cin>>bb>>bbb;
 getline(cin,musor);
 getline(cin,s);
 for(int i=0;i<s.size();i++)
 {
  if(s[i]==char(32))
  {
   start=i+1;
   break;
  }
 }
// cout<<start<<endl;

 for(int i=start;i<s.size();i++)
 {
  if(s[i]==char(32))
  {
   if(atof(tmp.c_str())>max)
   {
    max=atof(tmp.c_str());
    sm=tmp;
   }

//   cout<<atoi(tmp.c_str())<<'   ';
   tmp.clear();
  }
  else
  tmp=tmp+s[i];
 }
// cout<<atoi(tmp.c_str())<<'   ';
 if(atof(tmp.c_str())>max)
   {
    max=atof(tmp.c_str());
    sm=tmp;
   }



cout<<sm;
return 0;
}
```

## Задача №7

**Условие**

Автоматика, управляющая грузовым лифтом, оперирует только четырьмя командами: «Открыть двери», «Закрыть двери», «Поехать на второй этаж», «Поехать на первый этаж». Из-за недоработок в системе нельзя два раза подряд выполнить команду «Закрыть двери». Остальные попытки обрабатываются системой корректно.

Сколько существует последовательностей команд длиной n, не приводящих к ошибке и не содержащих двух последовательных перемещений между этажами, если лифт в начальный момент времени стоял на втором этаже?

На вход программе подается натуральное число n (N≤20) – количество команд.

Вывести целое число – количество последовательностей команд длиной n, не приводящих к ошибке и не содержащих двух одинаковых команд подряд.

Пример

| Ввод | Вывод |
|---|---|
| 2 | 7 |

**Исходный код**

```
#include<bits/stdc++.h>
using namespace std;
int main(){
long long int  o,z,Z,p1,p2,n,O,P1,P2,count=0;
cin>>n;
o=1;
z=1;
p1=0;
p2=0;

for(int i=0;i<n-1;i++)
{
 O=0;
 Z=0;
 P1=0;
 P2=0;
 O=o+z+p1+p2;
 Z=o+p1+p2;
 P1=o+z;
 P2=o+z;

 o=O;
 z=Z;
 p1=P1;
 p2=P2;
}
cout<<o+z+p1+p2;

return 0;
}
```