Државно такмичење из програмирања, Београд – 2. април 2016.
I категорија (5. и 6. разред)

У сваком задатку временско ограничење је 1 секунда, а меморијско ограничење је 64 MB.

1. Посматрајмо бесконачан низ 1, 1, 2, 1, 2, 3, 1, 2, 3, 4, 1, 2, 3, 4, 5... који се се формира на следећи начин: први члан тог низа је 1, потом следе сви природни бројеви од 1 до 2, потом следе природни бројеви од 1 до 3, потом сви природни бројеви од 1 до 4 и тако даље. Написати програм (конзолну апликацију) **NIZN** који ће за дати број ***n*** исписати ***n***-ти члан описаног низа. На пример, за n=55, потребно је исписати члан који се у низу налази на позицији 55, а то је број 10. Позиције се броје почев од 1 (а не почев од 0).
УЛАЗ: У једином реду стандардног улаза се налази један природан број ***n*** ($1 \leq n \leq 1\ 000\ 000$)
ИЗЛАЗ: У једином реду стандардног излаза исписати ***n***-ти члан описаног низа.
ПРИМЕР

УЛАЗ	ИЗЛАЗ	УЛАЗ	ИЗЛАЗ	УЛАЗ	ИЗЛАЗ
2	1	55	10	56	1

Решење

Позицију *n* умањимо за 1, тј. прескочимо први блок. Пронађимо унутар ког блока је *n*-ти члан. Зато најпре од броја *n* одузмемо 1, потом одузмемо 2, онда одузмемо 3 и тако даље наставимо док не добијемо негативну вредност за број *n*. Број извршених одузимања једнак је броју блока, а позиција унутар блока је последња ненегативна вредност коју ћемо добити.

```
#include <iostream>
using namespace std;

int main() {
  long long n;
  cin >> n;

   n--;
   for (long long i = 1; i <= n; i++) {
      n -= i;
    }
    cout << n + 1 << endl;

    return 0;
}
```

2. На почетку рада машина за прављење папирних новчаница садржи тачно 1 новчаницу, а за дати број N потребно је да машина направи тачно N новчаница. На машини постоје две команде: „НАПРАВИ“ или „СЕЦКАЈ“. Ако користите команду „НАПРАВИ“, машина ће креирати још M новчаница. Ако користите команду „СЕЦКАЈ“, машина ће исецкати, тј. уништити Z новчаница које се налазе у њој. Команде можете бирати произвољним редом, али у машини број папирних новчаница мора увек бити већи или једнак 1. Написати програм (конзолну апликацију) **NOVAC** који ће израчунати најмањи број команди које треба употребити да би машина произвела тачно N новчаница. Претпоставите да подаци на улазу су такви да се N новчаница може добити са не више од 1 000 000 позива на неку од команди.
Улаз: У првом реду стандардног улаза се налази се природни број N (1 ≤ N ≤ 1 000 000), број новчаница које мора машина да направи;
У другом реду стандардног улаза се налази се природни број M (1 ≤ M ≤ 1000), број новчаница које машина направи једним притиском на команду „НАПРАВИ“ ;
У трећем реду стандардног улаза се налази се природни број Z (1 ≤ Z ≤ 1000), број новчаница које машина исецка једним притиском на команду „СЕЦКАЈ“ ;

Излаз: У једином реду стандардног излаза исписати један број, тражени најмањи могући укупни број команди машине тако да се произведе тачно N новчаница.

Примери

Улаз	**Улаз**
10 11 1	987654 999 1000
Излаз	**Излаз**
3	1705

Објашњење 1. примера: У машини је на почетку 1 новчаница. Командом „НАПРАВИ“ добије се још 11 новчаница и укупно их има 12. Командом „СЕЦКАЈ“ исецка се 1 новчаница, те их је укупно 11. Поновимо команду „СЕЦКАЈ“, исецка се 1 новчаница, те их је укупно 10. Укупан број команди је 3.

Решење:

```
program novac;
var n, m, z, i : longint;
begin
   readln (n);
   readln (m);
   readln (z);
   {potrebno je napraviti n - 1 novcanica}
   n := n - 1;
   i := 0;
   {
   trazimo najmanji broj i za koji vazi:
   i * m - n >= 0 i
   i * m - n celobrojno podeljeno sa z daje ostatak 0 - (i * m - n) mod z = 0
   }
   while (i * m - n < 0) or ((i * m - n) mod z <> 0) do
      i := i + 1;
   {
   broj pritisaka na komandu NAPRAVI: i
   broj pritisaka na komandu ISCEPKAJ: (i * m - n) / z
   }
   writeln (i + (i * m - n) div z);
end.
```

3. Дат је цео позитиван број ***N*.** Означимо производ узастопних природних бројева **1, 2, 3, 4, ..., *N*** са ***N*!= 1*2*3* *4* ...*N***. На пример, 4! = 24. Напишите програм (конзолну апликацију) **FAKTOR** која за дати број ***N*** штампа збир експонената (изложилаца степена) простих бројева који се добијају при растављању броја *N*! на просте чиниоце. На пример, број 4! = 24=2^3*3^1, те је тражени збир 3+1=4.

Улаз: У једином реду стандардног улаза дат је природан број N (1 < N < 9999).

Излаз: У једином реду стандардног излаза исписати тражени збир експонената.

ПРИМЕР

УЛАЗ	**ИЗЛАЗ**	**УЛАЗ**	**ИЗЛАЗ**	**УЛАЗ**	**ИЗЛАЗ**
4	**24**	**6**	**7**	**20**	**36**

Објашњење 2. примера: 6! = 720 = 2^4 3^2 5^1, те је тражени збир 4+2+1=7.

Објашњење 3. примера: 20!= 2432902008176640000=$2^{18}3^8 5^4 7^2 11^1 13^1 17^1 19^1$, те је тражени збир 18+8+4+2+1+1+1+1=36.

Решење:

```
#include <iostream>
using namespace std;
const int fin=10000;
int a[fin+1];

int main()
```

```
{ int n;
  cin >> n;

  for(int i=2;i<=n;i++)
  {
          int cinilac=i;
          int pfaktor=2;
          while(cinilac>1)
          {
            while(cinilac%pfaktor==0) {cinilac /=pfaktor; a[pfaktor]++;}
            pfaktor++;
          }
  }

  int br=0;
  for(int i=1;i<=fin;i++) br += a[i];

  cout << br << endl;


}
```

Корисне напомене у вези прегледања задатака

- Оставите себи времена како бисте проверили да ли сте у кореном директоријуму креирали датотеке чији назив је исти као и назив задатка у формулацији коју сте добили (нпр. NIZN,…)
- У том фолдеру се памте искључиво .pas, .c .cpp,... изворни (source) кодови чија имена морају бити као у формулацији задатка.
- За решавање задатака, такмичари могу да користе програмске језике C, C++, Pascal, Basic, C#.
- Такмичари обавезно креирају конзолне апликације због аутоматског прегледа задатака (C, C++, Pascal, C# tj. FreePascal, gcc, g++ компајлер) и полуатоматског прегледа задатака (Basic).
- Подаци се читају/исписују преко стандардног улаза и излаза - немојте користити додатне датотеке!
- Излазни подаци морају бити тачно у облику датим у опису задатка. Немојте исписивати додатне поруке “Решење је...”.
- На крају програма обавезно уклонити наредбе које неки од Вас користе како би задржали излазни екран са исписаним резултатом:

Pascal	**C/C++**	**C#**

readln;	for(;;);
	system("pause")
	getchar();
	getch();
		Console.ReadLine();
		Console.Read();
		Console.ReadKey();
		for (;;);

- Уколико је потребно користити 64-битне бројеве, користите int64 у Pascal-у, односно long long у C/C++-у; обратите пажњу да long у C/C++-у не мора увек бити 64-битни тип.
  Уколико за учитавање/испис 64-битних бројева у C/C++-у користите функције scanf/printf, потребно је употребити спецификатор %lld.
- У C/C++ кодовима, користити <iostream> а не <iostream.h>. Такође, морате експлицитно include-овати све бибилиотеке чије функције користите (нпр. <cstring>, <cstdlib>, <algorithm>).

У неким окружењима (DevC++) ваш код ће радити и без тога али не и на званичном систему! Слично је и са укључивањем неких библиотека из алата Microsoft Visual C++ (нпр. <stdafx.h>,...) које нису подржане у званичном компајлеру.

- У C/C++ кодовима функција main мора бити декларисана као “int main()” а не као “void main()” или “main()”. Такође, ова функција мора враћати вредност, тј. морате имати “return 0;”

ПОНЕДЕЉАК 04.04.2016.

До 15ч – привремена ранг листа на сајту Рачунарске гимназије и Друштва математичара Србије

ПЕТАК 08.04.2016.

До 15ч – подношење приговора на резултате привремене ранг листа е-поштом на адресу takmicenjeinf@gmail.com

Претходно се региструјте на платформи BubbleBee.org и сами тестирајте број тест примера који Вам успешно пролазе. Слично, можете превући тест примере и са сајта Друштва математичара Србије и ручно проверити рад Ваших програма.

ПОНЕДЕЉАК 11.04.2016.

Објава коначних резултата на сајту Друштва математичара Србије