Разбор задач тренировки № 6

A. Напротив

(Время - 1 сек., память - 16 Мб)

Школьники одного класса встали по кругу на одинаковом расстоянии друг от друга и в порядке их номеров в журнале. Сколько человек в классе, если школьник с номером *a* стоит напротив школьника с номером *b*?

Входные данные

В единственной строке входного файла INPUT.TXT записаны два натуральных числа – номера двух школьников, стоящих напротив друг друга. Числа не превышают 1000000000.

Выходные данные

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно натуральное число — количество человек в классе. Если такое невозможно, то вывести “No”.

Примеры

№	input.txt	output.txt
1	1 2	2
2	3 1	4
3	2 3	No

Разбор

Пусть a<b. Если это не так, то поменяем у них значения. Между a и b стоит b-a-1 школьников. Если считать по другую сторону, то там не меньше a-1 школьников, и их должно быть не больше чем на другой стороне. То есть, b-a-1≥a-1, отсюда имеем, что b≥2a. В этом случае всего школьников будет 2(b-a-1)+2=2(b-a).

Программа на Паскале

```
var
   a, b, c : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(a,b);
  if a>b then begin c:=a; a:=b; b:=c end;
  if b>=2*a then write(2*(b-a)) else write('No')
end.
```

B. Фибоначчиева последовательность

(Время - 1 сек., память - 16 Мб)

Последовательность чисел a_1, a_2, …, a_i,… называется фибоначчиевой, если для всех $i \geq 3$ верно, что $a_i=a_{i-1}+a_{i-2}$, то есть каждый член последовательности (начиная с третьего) равен сумме двух предыдущих.

Ясно, что задавая различные числа a_1 и a_2 мы можем получать различные такие последовательности, и любая фибоначчиева последовательность однозначно задается двумя своими первыми членами.

Будем решать обратную задачу. Вам будет дано число n и два члена последовательности: a_n и a_{n+1}. Вам нужно написать программу, которая по их значениям найдет a_1 и a_2.

Входные данные

В единственной строке входного файла input.txt записаны: число n и значения двух членов последовательности: a_n и a_{n+1} ($1 \le n \le 30$, члены последовательности — целые числа, по модулю не превышающие $2 \cdot 10^9$).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести два числа — значения первого и второго членов этой последовательности.

Пример

№	input.txt	output.txt
1	4 3 5	1 1

Разбор

Рассмотрим следующую стандартную схему вычисления чисел Фибоначчиевой последовательности по заданным a_1 и a_2.

a_1	a_2	$a_3 = a_1 + a_2$	…	…	$a_n = a_{n-2} + a_{n-1}$	$a_{n+1} = a_{n-1} + a_n$
Заданы		→	→	→	→	
a	b	c=a+b				
	a=b	b=c	c=a+b			

А теперь, зная a_n и a_{n+1} развернём вычисления в обратном порядке.

$a_1 = a_3 - a_2$	$a_2 = a_4 - a_3$		$a_{n-2} = a_n - a_{n-1}$	$a_{n-1} = a_{n+1} - a_n$	a_n	a_{n+1}
	←	←	←	←	*Заданы*	
				c=b-a	a	b
			c=b-a	c=b	b=a	

После этого уже легко написать программу вычислений.

Программа на Паскале

```
var
  n, i : integer;
  a, b, c : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n,a,b);
  for i:=n-1 downto 1 do begin
    c:=b-a; b:=a; a:=c
  end;
  write(a,' ',b);
  close(output)
end.
```

C. Рациональный множитель

(Время - 2сек., память - 16 Мб)

Дано натуральное число n. Разрешается умножать его на рациональное число $\frac{p}{q} < 1$, (p – простое число или единица, q – простое число), такое, что произведение опять натуральное. С получившимся в результате этого действия числом можно проделать такую же операцию (возможно, с другой дробью).

Найти количество операций в одной из наиболее длинной последовательности таких действий.

Входные данные

В единственной строке входного файла INPUT.TXT записано одно натуральное число n ($1 \le n \le 1000000$).

Выходные данные

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно натуральное число — количество операций.

Примеры

№	input.txt	output.txt
1	2	1
2	3	2

Разбор

Задача из книги С.Г. Волчёнкова с соавторами «Ярославские олимпиады по информатике. Сборник задач с решениями», изданной издательством «БИНОМ. Лаборатория знаний» в 2010 году. Она предлагалась в 1996-97 учебном году участникам городской олимпиады на теоретическом туре. Условие немного изменено и добавлены технические ограничения.

Если n – простое число, то максимальное количество действий равно его номеру в ряду простых чисел 2, 3, 5, 7, 11, 13, 17, 19, …. Если же n - составное, то надо просуммировать номера его простых сомножителей. Для определения простоты чисел и их номеров воспользуемся программой из книги Н. Вирта «Систематическое программирование. Введение», хоть и изданной издательством «Мир» в 1977 году, но доступной в электронном виде в интернете.

Программа на Паскале

```
Var
  n, otvet, x, i, k, lim : longint;
  prim : boolean;
  p : array [1..2000] of longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n);
  otvet:=0;
  while n mod 2=0 do begin
    otvet:=otvet+1; n:=n div 2
  end;
  while n mod 3=0 do begin
    otvet:=otvet+2; n:=n div 3
  end;
  x:=3; lim:=1; i:=1; p[1]:=3;
  while x*x<n do begin
    repeat
      x:=x+2;
      if sqr(p[lim])<=x then lim:=lim+1;
      k:=1; prim:=true;
      while prim and (k<lim) do begin
        prim:=x mod p[k]<>0; k:=k+1
      end
    until prim;
```

```
    i:=i+1; p[i]:=x;
    while n mod x=0 do begin
      otvet:=otvet+i+1; n:=n div x
    end
  end;
  while x<n do begin
    repeat
      x:=x+2;
      if sqr(p[lim])<=x then lim:=lim+1;
      k:=1; prim:=true;
      while prim and (k<lim) do begin
        prim:=x mod p[k]<>0; k:=k+1
      end
    until prim;
    i:=i+1
  end;
  if n>1 then otvet:=otvet+i+1;
  write(otvet);
  close(output)
end.
```

D. Ленточка

(Время: 1 сек. Память: 16 Мб)

Расположенную вертикально прямоугольную бумажную ленточку с закрепленным нижним концом стали складывать следующим образом:

- на первом шаге ее согнули пополам так, что верхняя половина легла на нижнюю либо спереди (P - сгибание) либо сзади (Z - сгибание),

- на последующих n-1 шагах выполнили аналогичное действие с получающейся на предыдущем шаге согнутой ленточкой, как с единым целым.

Затем ленточку развернули, приведя ее в исходное состояние. На ней остались сгибы - ребра от перегибов, причем некоторые из ребер оказались направленными выпуклостью к нам (K - ребра), а некоторые - от нас (O - ребра). Ребра пронумеровали сверху вниз числами от 1 до 2^n-1.

Требуется написать программу, которая по заданным строке символов из прописных букв "P" и "Z", определяющей последовательность типов сгибаний, и номерам рёбер сообщает тип этих ребер, получившийся после этой последовательности сгибаний.

Входные данные

Входной файл input.txt содержит в первой строке число n – количество сгибаний ленточки (n не более 60), во второй строке - набор n символов из прописных латинских букв "P" и "Z". Третья строка содержит в начале число k – количество рассматриваемых рёбер (не более 10), а далее их номера (числа от 1 до 2^n-1).

Выходные данные

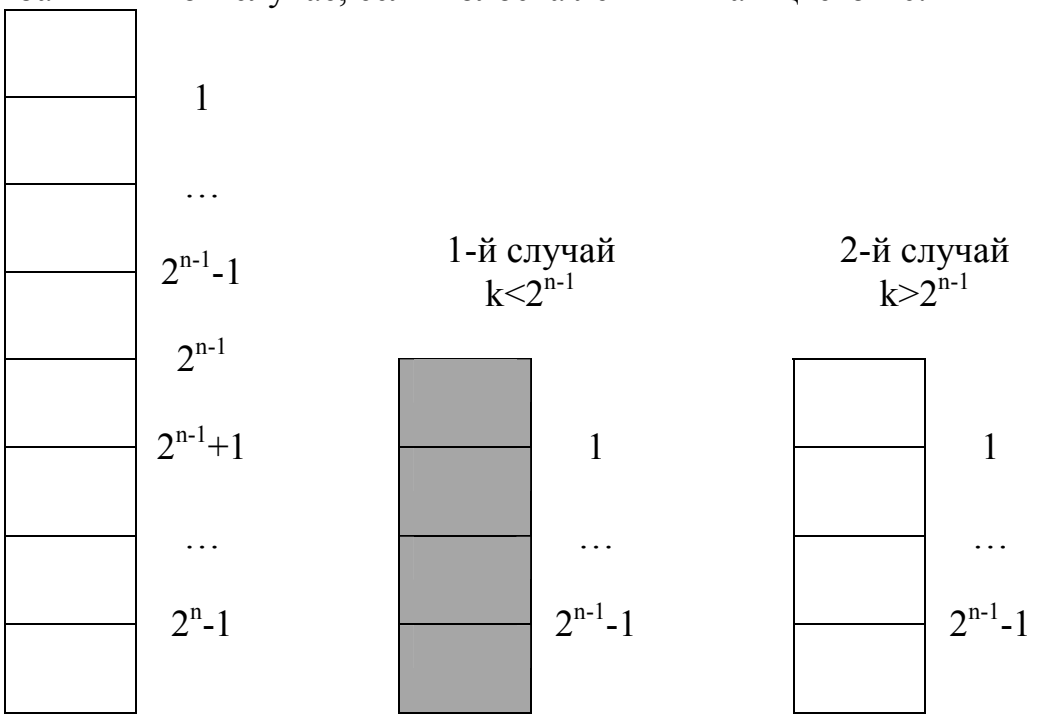
В единственную строку выходного файла output.txt нужно вывести k символов (прописные латинские буквы "K" или "O") – типы рассматриваемых ребра.

Примеры

№	input.txt	output.txt
1	2 PP 1 1	K
2	2 ZZ 2 1 2	OK

Разбор

Решим задачу методом уменьшения размера – одним из распространённых методов решения задач в математике и информатике. Для этого будем считать, что бумажная ленточка с одной стороны окрашена 1-м цветом (белым), а с другой – 2-м (тёмным). Обозначим через Z(n, k, c) решение задачи поиска типа k-го ребра при n сгибаниях в том случае, если полоска лежит к нам цветом c.



Из рисунка видно, что если $k= 2^{n-1}$, то тип сгиба можно определить по имеющейся информации о 1-м сгибании. Если $k<2^{n-1}$, то задача Z(n, k, c) имеет такое же решение как задача $Z(n-1, 2^{n-1}-k, 3-c)$. Если $k>2^{n-1}$, то задача Z(n, k, c) имеет такое же решение как задача $Z(n-1, k-2^{n-1}, c)$. Таким образом, мы или уже имеем решение задачи или свели её по 1-му параметру на единицу меньшему, по 2-му не менее вдвое меньшему и, возможно, третьему изменённому (с 1 на 2 или наоборот). Продолжим сведение задач и получим решение максимум за n таких сведений.

Программа

```
var
  k, st, t : int64;
  n, cv, i, j : integer; s : string;
begin
  assign(input, 'input.txt'); reset(input);
  assign(output, 'output.txt'); rewrite(output);
  readln(n); readln(s); read(j);
  st:=1; for i:=2 to n do st:=st*2;
```

```
  while j>0 do begin
    j:=j-1; t:=st; i:=1; cv:=1;
    read(k);
    while k<>t do begin
      if k>t then k:=k-t else begin cv:=3-cv; k:=t-k end;
      t:=t div 2; i:=i+1
    end;
    case cv of
      1 : if s[i]='P' then write('O') else write('K');
      2 : if s[i]='P' then write('K') else write('O');
    end
  end;
  close(output)
end.
```