Computer
Programming
Olympiad

A project of the Institute of IT Professionals South Africa

Ph: 021-448 7864 • Fax: 021-447 8410 • PO Box 13013, MOWBRAY, 7705 • info@olympiad.org.za • www.olympiad.org.za

PROGRAMMING OLYMPIAD 2015: Round 1

1. WELL ORDERED WORDS

Introduction:

A word is well ordered when its letters are in alphabetical order. The word **act** is a well ordered word; **cat** is not as **a** comes before **c** in the alphabet.

Task:

Write a program that will ask for a word (any word) as input and produce, as output, the word “**True**” if the word is well ordered or the word “**False**” if the word is not well ordered.

Example:

Input: `will`

Output: `False`

Input: `all`

Output: `True`

Test:

Test your program with the following words; and enter your answers on the website for Q1(a), Q1(b) and Q1(c).

(a) `dog`
(b) `abbozzo`
(c) `effort`

[Adapted from the ICPSC]

2. BUYING MARBLES

Introduction:

John has marbles of different colours and would like to know how many marbles he must buy so that he has the same number of marbles of each colour.

Task:

Write a program that asks how many colours and how many marbles of each colour John has, and that will then calculate the least number of marbles he has to buy in order to have the same number of marbles of each colour. The first line of the input gives the number of colours; the second line the number of marbles of each colour. You may design your program in such a way that the inputs are read individually with successive prompts or as one or two longer lines of input.

Examples:

Input:
```
3
1 2 2
```

Output: `1`

Input:
```
4
3 4 5 3
```

Output: `5`

Supported by Oracle and the University of Cape Town.

ORACLE ACADEMY

Sponsored by

Standard Bank

Explanation:

In the first example John has three colours; one marble of the first colour, two marbles of a second colour, and two marbles of a third colour. In order to have the same number (two) of each colour, he has to buy one more marble.

Test:

Test your program with the following values, and enter your answers on the website for Q2(a), Q2(b) and Q2(c).

(a)
```
2
3 3
```
(b)
```
3
4 2 3
```
(c)
```
5
1 2 7 3 4
```

[Darren Roos, past Bronze Medal winner]

3. COUNTING LETTERS

Task:

Write a program that will ask for a word (any word) as input and provide the letters of that word as output, listing each letter only once and in the order in which the letters first appear in the word, but indicating after each letter how many times it appears in the word. Do not leave any spaces between letters and numbers.

Examples:

Input: `floor`

Output: `f1l1o2r1`

Input: `robot`

Output: `r1o2b1t1`

Test:

Test your program with the following words; and enter your answers on the website for 3(a), Q3(b) and Q3(c).

(a) `weed`

(b) `asthma`

(c) `mississippi`

[Adapted from the ICPSC]

4. ALIEN NUMBERS

Introduction:

The Froogons have made contact with Earth! Their advanced technology has made communication easy, except for their weird number system. They use factorials. *N* factorial, written as *N*!, is equal to 1 x 2 x 3 x … x *N*. For example, 3! = 1 x 2 x 3 = 6 and 1! = 1. The Froogons write a number as a sequence where the first number from the left indicates the number of 1!s, the second number from the left indicates the number of 2!s, the third number indicates the number of 3!s, etc, The *i*-th number in the sequence is at most *i* and represents how many *i*!s are included in the number. For example, the 3rd number in the sequence is at most 3 and represents how many 3!s are included in the number.

Task:

Write a program that asks for a decimal number as input and outputs its Froogon representation.

Input: A single positive integer *N*, in decimal notation.

Output: *N* written in Froogon notation. Your answer should be on a single line with a single space separating adjacent numbers in the sequence. Leading zeros must be shown.

Examples

Input: `13`

Output: `1 0 2`

Input: `17`

Output: `1 2 2`

Input: `24`

Output: `0 0 0 1`

In Froogon, 13 is written as 1 0 2 (i.e. 1 x 1! + 0 x 2! + 2 x 3!).

In Froogon, 17 is written as 1 2 2 (i.e. 1 x 1! + 2 x 2! + 2 x 3!).

In Froogon, 24 is written as 0 0 0 1 (i.e. 0 x 1! + 0 x 2! + 0 x 3! + 1 x 4!).

Test:

Test your program with the following numbers; and enter your answer on the website for Q4(a), Q4(b) and Q4(c).

(a) `18`

(b) `719`

(c) `2100100100`

[Sean Wentzel, past Bronze, Silver and Gold Medal winner]

5. PRIME GENERATING INTEGERS

Introduction:

Consider the factors of 30: 1, 2, 3, 5, 6, 10, 15, 30

It can be seen that for every factor (*f*) of 30, *f*+30/*f* is prime.

(A prime number is a whole number greater than 1, whose only two whole-number factors are 1 and itself. The first few prime numbers are 2, 3, 5, 7, 11, 13, 17, 19, 23, and 29.)

Task:

Write a program that asks for a number *n* as input and then calculates the **sum of all positive integers smaller than or equal to *n*,** such that for every factor (*f*) of *n*, (*f*+*n*/*f*) is prime.

Example 1:

Input: `6`

Output: `9`

Explanation:

- The positive integers smaller than or equal to 6 are 1, 2, 3, 4, 5 and 6.
- The only factor of 1 is 1. Apply the formula (*f*+*n*/*f*) to this factor. 1 + 1/1 = 2; 2 is a prime number and therefore the integer 1 is part of your sum.
- The factors of 2 are 1 and 2. Apply the formula (*f*+*n*/*f*) to each factor. 1 + 2/1 = 3; 2 + 2/2 = 3. 3 is a prime number and therefore the integer 2 is part of your sum.
- The factors of 3 are 1 and 3. 1 + 3/1 = 4. 4 is not a prime number, so there is no need to go further
- The factors of 4 are 1, 2 and 4. 1 + 4/1 = 5; 2 + 4/2 = 4. 4 is not a prime number, so there is no need to go further
- The factors of 5 are 1 and 5. 1 + 5/1 = 6. 6 is not a prime number, so there is no need to go further.
- The factors of 6 are 1, 2, 3, and 6. 1 + 6/1 = 7; 2 + 6/2 = 5; 3 + 6/3 = 5; 6 + 6/6 = 7. Both 7 and 5 are prime numbers. Therefore the integer 6 is part of your sum.
- The answer is therefore 1 + 2 + 6 = 9

Example 2:

Input: `10`

Output: `19`

Test:

Test your program with the following values; and enter your answers on the website for Q5(a), Q5(b) and Q5(c).

(a) *n* = `20`
(b) *n* = `10000`
(c) *n* = `100000000`

Hint:

If your program runs for more than 3 minutes, rather abandon the run.

[Adapted from Project Euler Problem 357]

6. DODGEBALL

Introduction:

Umar has finished programming his hit video game, Dodgeball, and he needs your help to check if the levels are not too hard.

Dodgeball takes place on a court that is *N* blocks wide with a player character, called Aphiwe, who is *K* blocks wide. Every second, two things happen. First, a ball is fired at a predetermined position on the court. Second, in order to try to avoid the ball, Aphiwe either moves one block left or one block right, or stays where she is The goal is for Aphiwe to avoid getting hit by the balls for as long as possible. . Aphiwe knows the entire sequence in which the balls will be fired, and thus can make choices in the early moves that will avoid her getting hit by later balls. Positions are numbered from 1 on the left of the court to *N* on the right, and at the start of the game Aphiwe takes up the *K* leftmost blocks of the court (those numbered from 1 to *K*). A level consists of *L* seconds.

Task:

You will be given *N, K* and *L,* and the positions at which the balls will be fired at each second. Write a program that will ask for the inputs and will then determine the longest time, in seconds, before Aphiwe is hit by a ball.

Input:

The first line of input will contain three space-separated integers, *N, K* and *L.* The second line of input will contain *L* space-separated integers. The *i*-th of these indicates at which position a ball will be fired in the *i*-th second.

Output:

A single integer, the maximum number of seconds that Aphiwe can survive before being hit by a ball. If Aphiwe can survive the whole level, output “Complete" instead.

Examples:

Input:
```
4 2 2

 4 2
```
Output:
```
Complete
```

Input:
```
5 3 6

 4 5 2 1 3 2
```
Output:
```
5
```

Explanation:

At the beginning, Aphiwe is on the left of the court, occupying blocks 1 to 3. The first ball gets fired at block 4 and the second ball at block 5. . In order to dodge these balls, Aphiwe need not move at all, BUT then she will have no way of avoiding the third ball fired at position 2 and she would only last two seconds without being hit.

As she know the entire sequence of balls beforehand, a better strategy would be to move one block right when the first ball is fired and right again when the second ball is fired; so Aphiwe then occupies blocks 3 to 5. The third ball is fired at block 2, so Aphiwe must stay where she is to avoid being hit. The fourth ball is fired at block 1 and Aphiwe can stay where she is again, The fifth ball is fired at block 3 and there is no way Aphiwe can dodge it. So with this strategy Aphiwe lasts 5 seconds before she is hit .

Hint:

This problem has some long inputs. In order to make sure you don't make a mistake entering them into your program, it is recommended you copy and paste them.

Test:

Test your program with the following numbers; and enter your answers on the website for Q6(a), Q6(b) and Q6(c).

(a)
```
10 4 8

6 1 2 4 8 6 4 2
```

(b)
```
20 9 16

16 1 20 1 2 4 16 4 8 1 2 4 8
16 8 16
```

(c)
```
40 5 36

40 30 20 10 1 2 3 4 6 5 7 9
17 10 16 11 1 40 14 13 16 11
18 9 20 7 15 23 19 26 5 8 15
30 12 33
```

[Sean Wentzel, past Bronze, Silver and Gold Medal winner]