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

COMPUTER PROGRAMMING OLYMPIAD

2015

POSSIBLE SOLUTIONS

Supported by Oracle and the University of Cape Town.

ORACLE ACADEMY

Sponsored by

Standard Bank

NOTE: Solutions to the problems have been tested using the following programming languages:

Java solutions:	jGrasp 2.0.1_06
Pascal solutions:	Lazarus 1.4.2 (FPC 2.6.4)
Python solutions:	PyCharm Community Edition 4.5 (Python v2.7)
Scratch solutions:	Scratch 2 Offline Editor

CONTRIBUTORS:

Max Brock	IT Curriculum Adviser in the Western Cape
Shamiel Dramat	IT Curriculum Adviser in the Western Cape
OER Foss	Educator and open-source advocate
Sean Wentzel	Past Bronze, Silver and Gold Medal winner; IOI 2015 Team Leader

QUESTION 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 Cases:

Test your program with the following words; enter your answer in the space provided below and save.

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

[Adapted from the ICPSC]

Answers:

(a) False
(b) False
(c) True

JAVA SOLUTION

```
import java.util.Scanner;

public class WellOrdered {

   public static boolean inOrder (String word) {
      boolean ordered = true;
      for (int  i = 0; i < word.length()-1; i++) {
         if (word.charAt(i) > word.charAt(i+1))
            ordered = false;
      }
      return ordered;
   }

   public static void main(String[]   args) {
      WellOrdered  pgn = new WellOrdered();
      Scanner keybd = new Scanner(System.in);
      System.out.print("Enter word: ");
      String userInput = "";
      userInput = keybd.next();

      if (inOrder(userInput))
         System.out.println("True");
      else
         System.out.println("False");
   }
}
```

[Adapted by OER Foss from original solution produced by Max Brock]

PASCAL SOLUTION

```
program WellOrdered;

Uses Sysutils;

var
   count : integer;
   len : integer;
   ordered : boolean;
   userInput : string;

begin
   ordered := TRUE;
   write('Enter word: ');
   readln(userInput);
   len := length(userInput);
   for count := 1 to len-1 do
      if userInput[count] > userInput[count+1] then
         begin
            ordered := FALSE;
```

```
            break;
         end;
   if ordered then
         writeln('True')
   else
      writeln('False');

   readln;
end.
```

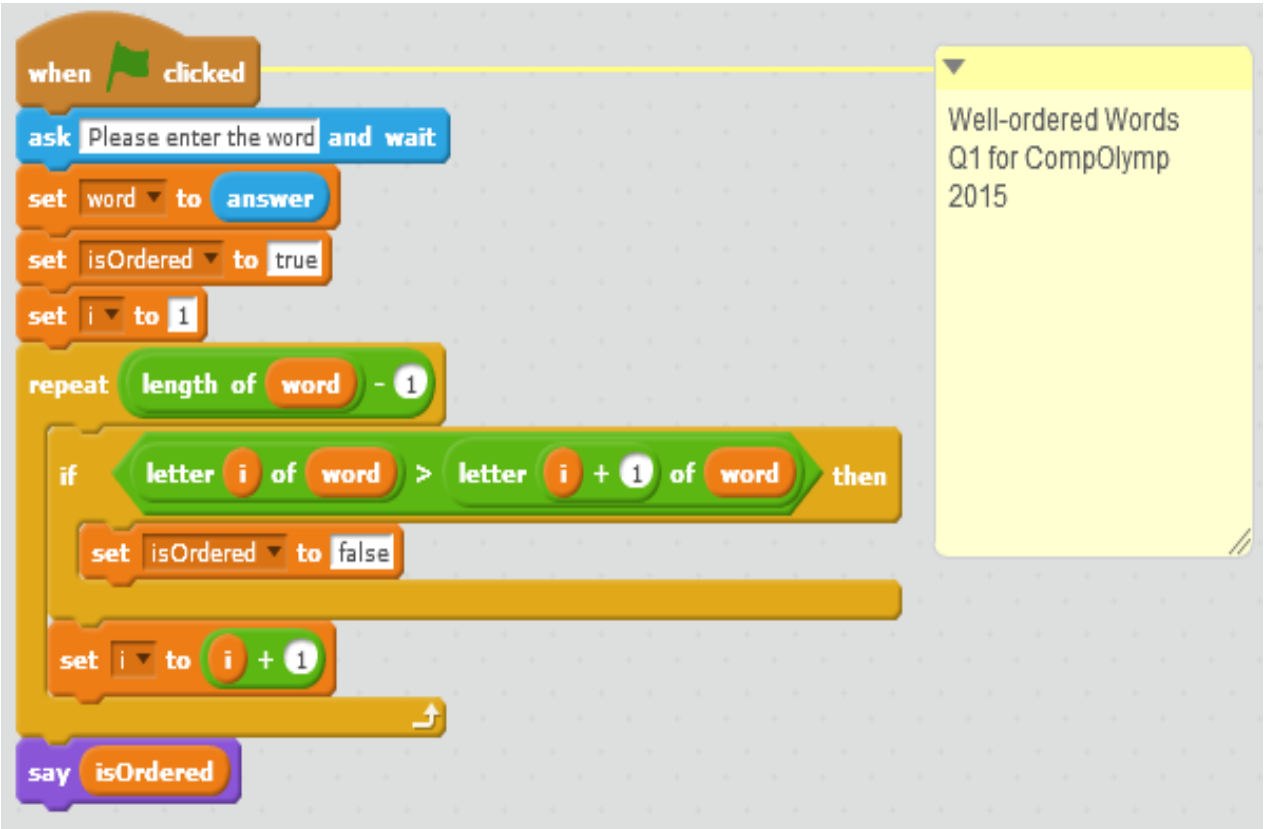
[Solution produced by OER Foss]

PYTHON SOLUTION

```
print 'Enter word:'
word = list(raw_input())
copy = word[:]
copy.sort()
print (word == copy)
```

[Solution produced by Sean Wentzel]

SCRATCH SOLUTION



[Solution produced by Max Brock]

QUESTION 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
```

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 Cases:

Test your program with the following values, enter your answer in the space provided below and save.

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

[Darren Roos, past Bronze Medal winner]

Answers:

(a) 0
(b) 3
(c) 18

JAVA SOLUTION

```
import java.util.Scanner;
import java.lang.Integer;

public class BuyingMarbles {

   public static void main (String[] args){

   Scanner sc = new Scanner(System.in);

   System.out.print("Enter the number of different colours: ");
   int numColours = sc.nextInt();

   System.out.print("Enter the number of each colour separated by spaces: ");
   String input = sc.nextLine();

   int temp;
   int max = 0;
   int sum = 0;

   for(int x = 0;x < numColours;x++){
      temp = Integer.parseInt((String)sc.next());
      sum += temp;
      if (temp > max) {
         max = temp;
         }
      }
   System.out.print(numColours*max-sum);
   }
}
```

[Adapted by OER Foss from original solution produced by Max Brock]

PASCAL SOLUTION

```
program Marbles;

Uses Sysutils;

var
  count : integer;
  max : integer;
  number : integer;
  numColours : integer;
  numMarbles : integer;

begin
  max := 0;
  numMarbles := 0;
  write('Enter number of different colours: ');
  readln(numColours);

  for count := 1 to numColours do
      begin
           write('Enter number of marbles for colour ', count, ': ');
           readln(number);
           if (number > max) then
              max := number;
           numMarbles := numMarbles + number;
      end;

  writeln(numColours*max - numMarbles, ' marbles required.');
  readln;
end.
```

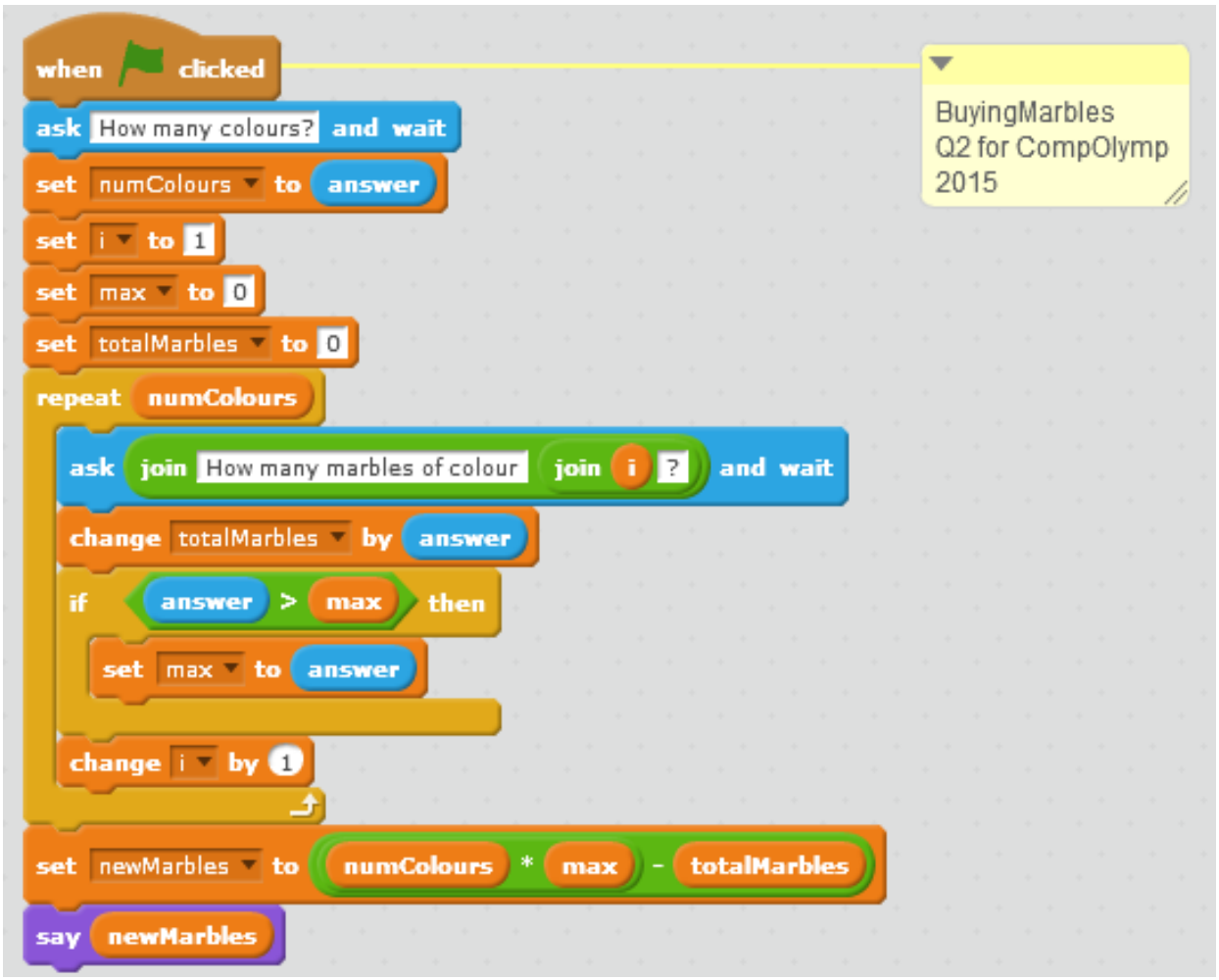
[Solution produced by OER Foss]

PYTHON SOLUTION

```
n = int(raw_input())
marbles = map(int, raw_input().split())
print n * max(marbles) - sum(marbles)
```

[Solution produced by Sean Wentzel]

SCRATCH SOLUTION



[Solution produced by Max Brock]

QUESTION 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 Cases:

Test your program with the following words; enter your answer in the space provided below and save.

(a) `weed`
(b) `asthma`
(c) `mississippi`

[Adapted from the ICPSC]

Answers:

(a) w1e2d1
(b) a2s1t1h1m1
(c) m1i4s4p2

JAVA SOLUTION

```
import java.util.Scanner;
import java.util.ArrayList;

public class CountingLetters {

   public static void main (String[] args){
      ArrayList<String> lettersUsed = new ArrayList();
      Scanner keybd = new Scanner(System.in);
      String input;
      String output = "";

      System.out.print("Enter the word: ");
      input = keybd.next();
      int count = 1;
      for (int j = 0; j < input.length(); j++) {
         for (int k = j + 1; k < input.length(); k++) {
            if (input.charAt(j) == input.charAt(k)) {
               count++;
            }
```

```
      }
      if (!lettersUsed.contains("" + input.charAt(j))) {
         output += "" + input.charAt(j) + count;
         }
         count = 1;
         lettersUsed.add("" + input.charAt(j));
      }
      System.out.println(output);
   }
}
```

[Adapted by OER Foss from original solution produced by Max Brock]

PASCAL SOLUTION

```
program Letters;

Uses Sysutils;

Var
  array_count : integer=1;
  array_increment : integer;
  count : integer;
  index : integer;
  letter : char;
  letter_array : array[1..10] of char;
  num_array : array[1..10] of integer;
  word : string;

begin

  for count := 1 to 10 do
     begin
       letter_array[count] := ' ';
       num_array[count] := 0;
     end;

  write('Enter word: ');
  readln(word);

  for count := 1 to length(word) do
     begin
        letter := word[count];
        array_increment := 0;
        for index := 1 to array_count do
           begin
              if letter_array[index] = ' ' then
                 begin
                    letter_array[index] := letter;
                    inc(num_array[index]);
                    array_increment := 1;
                 end
```

```
               else if letter_array[index] = letter then
                  begin
                     inc(num_array[index]);
                     break;
                  end;
            end;
         array_count := array_count + array_increment;
      end;

  for count := 1 to array_count-1 do
     write(letter_array[count], num_array[count]);

  readln;
end.
```

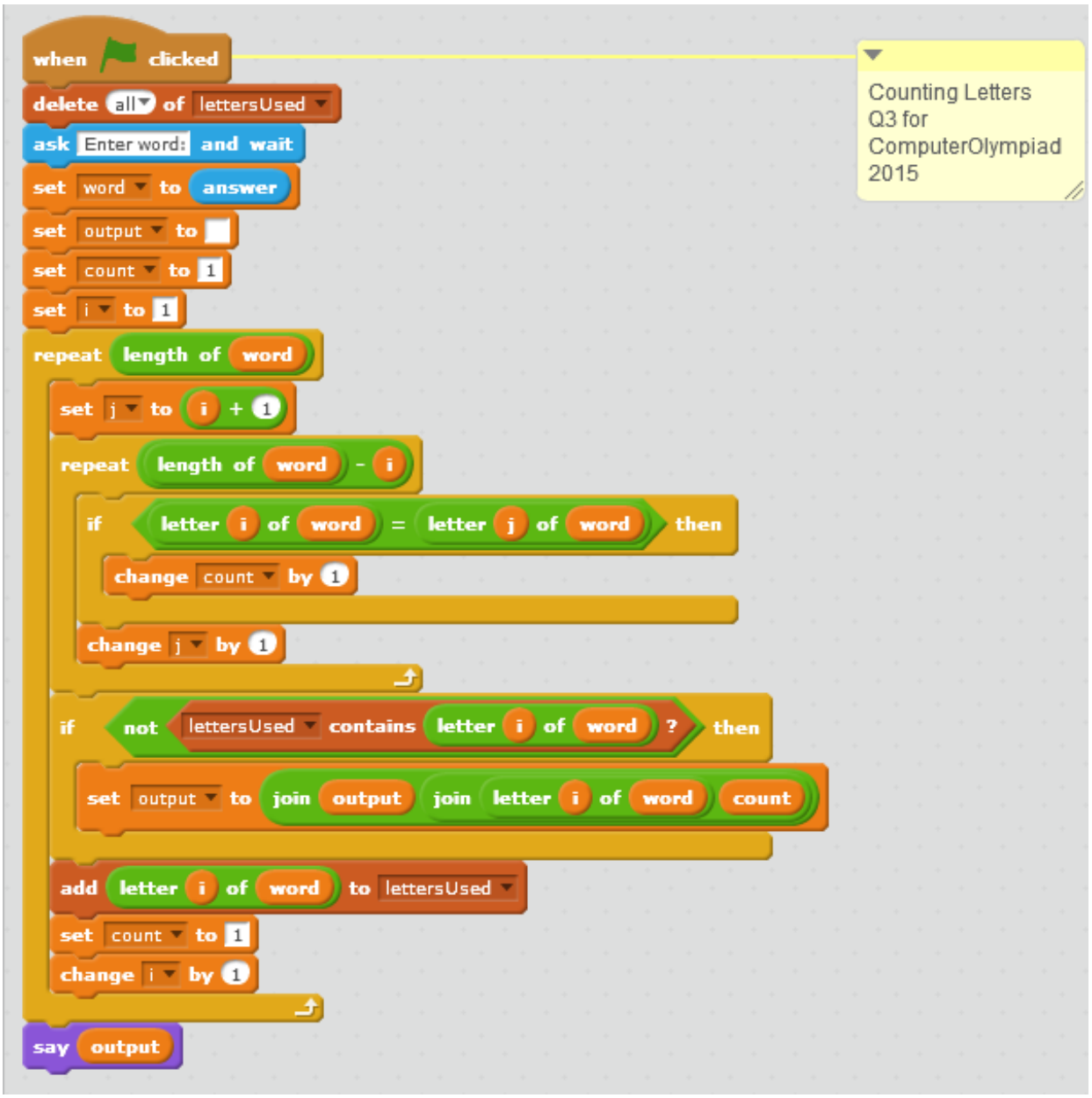
[Solution produced by OER Foss]

PYTHON SOLUTION

```
word = raw_input().strip()
already_seen = set()
ans = []
for letter in word:
    if letter in already_seen:
        continue
    ans.append(letter + str(len([same for same in word if same == letter])))
    already_seen.add(letter)
print ''.join(ans)
```

[Solution produced by Sean Wentzel]

SCRATCH SOLUTION



[Solution produced by Max Brock]

QUESTION 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 Cases:

Test your program with the following numbers; enter your answer in the space provided below and save.

(a) `18`
(b) `719`
(c) `2100100100`

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

Answers:

(a) 0 0 3
(b) 1 2 3 4 5
(c) 0 1 3 0 4 3 6 2 7 6 4 4

JAVA SOLUTION

```
import java.util.Scanner;

public class AlienNumbers {

   public static String froogonNum(long num) {
      long i = 1;
      for (i = 1; i < num; i++) {
         long fac = 1;
         for (int j = 1; j <= i; j++) {
            fac = fac * j;
            }
         if (fac > num) {
            break;
            }
      }
      String froog = "";
      for (long j = i - 1; j > 0; j--) {
         long fac = 1;
         for (int k = 1; k <= j; k++) {
            fac = fac * k;
            }
         long dig = num / fac;
         num = num - dig * fac;
         froog = dig + " " + froog;
         i++;
         }
      return froog;
      }

   public static void main (String[] args){
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter the number: ");
      long number = sc.nextLong();
      System.out.println(froogonNum(number));
   }
}
```

[Adapted by OER Foss from original solution produced by Max Brock]

PASCAL SOLUTION

```
program Froogon;

Uses Sysutils;

Var
   froog : string;
   i, j : integer;
   intdiv : integer;
   number : Int64;
   numpos : integer;

function factorial(n: integer): Int64;

var
   i: integer;
   product: Int64;

begin
   product := 1;
   for i := 1 to n do
      product := product*i;
   factorial := product;
end;

begin
   froog := '';
   write('Number = ');
   readln(number);

   for j := 1 to 100 do
      if factorial(j) > number then
         break;
   numpos := j-1;
   for i := 1 to numpos do
      begin
         intdiv := number div factorial(j-i);
         froog := inttostr(intdiv)+' '+froog;
         number := number - intdiv * factorial(j-i);
      end;
   writeln(froog);

   readln;
end.
```

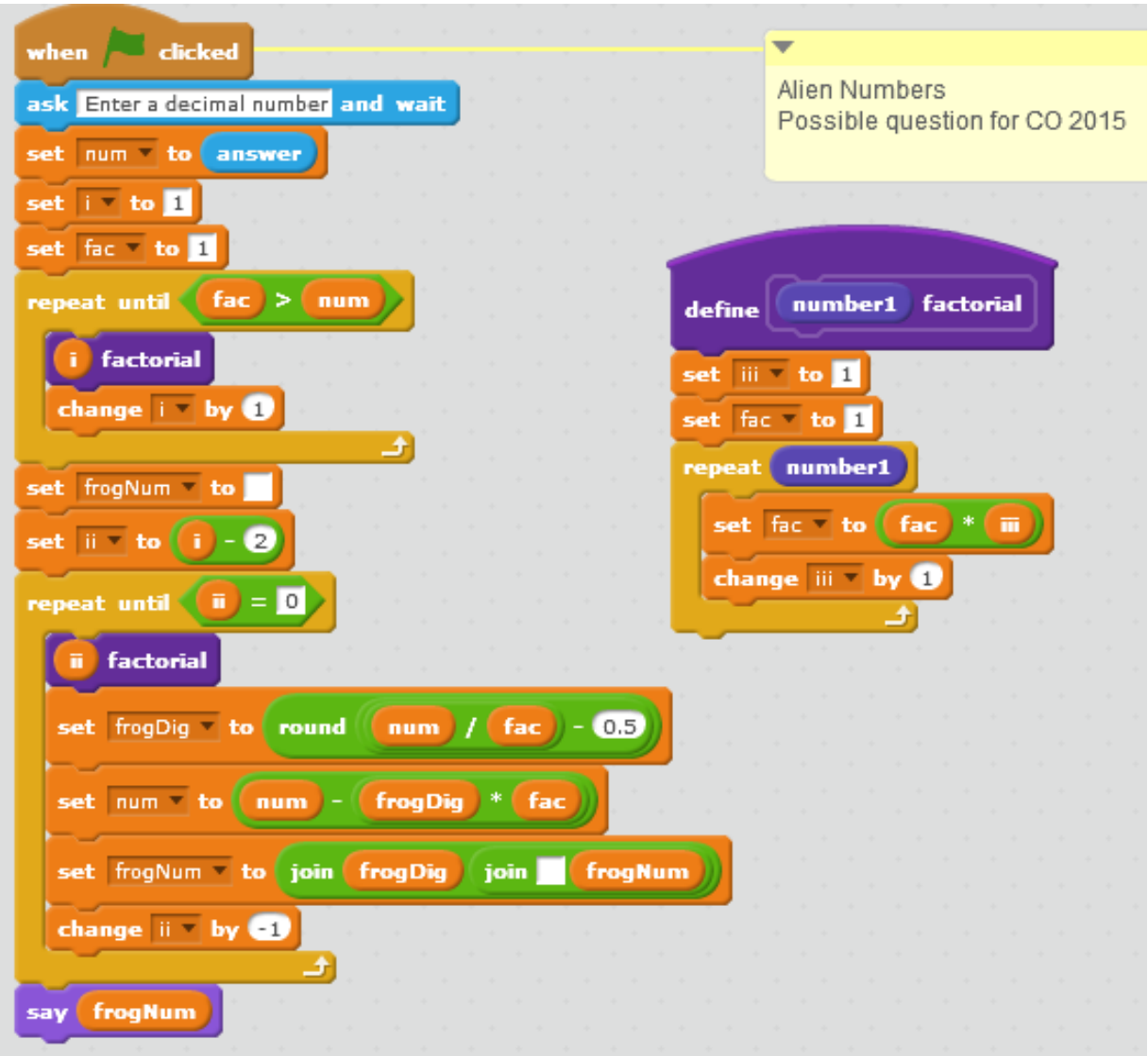
[Solution produced by OER Foss]

PYTHON SOLUTION

```
factorials = [1, 1] #0!, 1!
for i in xrange(2, 25):
    factorials.append(factorials[-1] * i)
n = int(raw_input())
a = 1 #index of smallest factorial > n
while n >= factorials[a]:
    a += 1
foo = [0 for i in xrange(a)]
for i in xrange(a-1, -1, -1): # for every factorial less than a!
    while(n >= factorials[i]): # represent as much of n as possible
        foo[i] += 1
        n -= factorials[i]
print " ".join(map(str,foo[1:]))
```

[Solution produced by Sean Wentzel]

SCRATCH SOLUTION



[Solution produced by Max Brock]

QUESTION 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 Cases:

Test your program with the following values; enter your answer in the space provided below and save.

(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]

Answers:

(a) 19
(b) 262615
(c) 1739023853137

JAVA SOLUTION

```
import java.util.Scanner;

public class PrimeGenNumbers {

   private boolean isPrime(int number) {
      double sqrt = Math.sqrt(number);
      if (number == (int) sqrt * (int) sqrt) {
         return false;
         }
      for (int i = 3; i <= (int) sqrt; i += 2) {
         if (number % i == 0) {
             return false;
            }
         }
         return true;
      }

   public static void main(String[] args) {
      PrimeGenNumbers pgn = new PrimeGenNumbers();
      long sum = 3;
      Scanner keybd = new Scanner(System.in);
      System.out.print("Enter number: ");
      Double maxNum = keybd.nextDouble();

      loop:
      for (int number = 4; number <= maxNum; number += 2) {
         if (!pgn.isPrime(number + 1)) {
             continue loop;
            }
            int divisor = 2;
            while (number / divisor >= divisor) {
                if (number % divisor == 0) {
                   if (divisor % 2 == 0 && number / divisor % 2 == 0 ) {
                       continue loop;
                    }
                    if (!pgn.isPrime(divisor + number / divisor)) {
                       continue loop;
                    }
                }
                divisor++;
            }
            sum += number;
        }
```

```
        System.out.println(sum);
    }
}
```

[Adapted by OER Foss from original solution produced by Max Brock]

PASCAL SOLUTION

```
program PrimeGenInt;

Uses Sysutils;

Var
  at : LongInt;
  good : boolean;
  j,cur : LongInt;
  N : Int64;
  Sieve : Array[2..100000002] Of Integer;
  Sum : Int64;

function isFactor(i,j: Int64): boolean;

begin
   isFactor := False;
   if(i mod j = 0 ) then
      isFactor := True;
end;

begin
   write('Enter value: ');
   readln(N);
   sum := 0;
   for at := 2 to N+1 do
      begin
         sieve[at] := 1;
      end;
   at := 2;
   while(at*at<=N+1) do
      begin
         if(sieve[at] = 1) then
            begin
               j:=at*at;
               while(j<= N+1) do
                  begin
                     sieve[j] := 0;
                     if(j div at mod at = 0) then
                        begin
                           sieve[j]:= -1;
                        end;
                  j+=at;
                  end;
```

```
            end;
            at+=1;
      end;
   cur:=2;
      while(cur<=N+1) do
         begin
            good:=True;
            if((sieve[cur]=-1) or (sieve[cur+1]<>1)) then
               begin
                  cur+=4;
                  continue;
               end;
            j := 2;
            while(j*j<=cur) do
               begin
                  if(isFactor(cur,j)) then
                     begin
                        if(sieve[(cur div j) + j] <> 1) then
                           begin
                              good:=False;
                              break;
                           end;
                     end;
                  j+=1;
               end;
            if(good) then
               begin
                  sum += cur;
               end;
            cur+=4;
         end;
   writeln('Sum = ', sum+1);

  readln;
end.
```

[Solution produced by Sean Wentzel]

PYTHON SOLUTION

```
from math import sqrt
EPS=1e-9
# returns an array where nums[i] is 1 if i is prime, 0 if it is not prime
# but squarefree, and -1 if it is not squarefree
def sieve(n):
    nums=[1 for i in xrange(n+2)]
    for i in xrange(2, int(sqrt(n+1)+EPS) + 1):
        if nums[i] != 1:
            continue
        j = i*i
        while j <= n+1:
            if nums[j] == 1:
                nums[j] = 0
            if (j/i) % i == 0 :
                nums[j] = -1
            j += i
    return nums
n = int(raw_input())
nums = sieve(n)
ans = 0
for i in xrange(2, n+1, 4):
# we only need consider numbers congruent to 2 mod 4. If a number x is odd,
# then x/1+1 certainly isn't prime and if 4|x then x/2+2 certainly isn't.
    good = True
    # if p^2|i then p|i/p+p so i must be squarefree, and i/1+1 must be prime
    if nums[i] == -1 or nums[i+1] != 1:
        continue
    for j in xrange(2, int(sqrt(n)+EPS) + 1):
        if i%j != 0:
            continue
        if nums[i/j + j] != 1:
            good = False
            break
    if good:
        ans += i
print ans + 1
```

[Solution produced by Sean Wentzel]

QUESTION 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 5
4 5 2 1 3
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 Cases:

Test your program with the following numbers; enter your answer in the space provided below and save.

(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]

Answers:

(a) 6
(b) 14
(c) 32

JAVA SOLUTION

```
import java.util.Scanner;

public class Dodgeball {
   int court;
   int[] balls;
   int prblock;
   int result = 0;

   public Dodgeball() {
   getInputs();
   play(0, 1, prblock);
   play(0, 2, prblock +   1);
   if(result == balls.length){
      System.out.println("Complete");
      }
      else{
      System.out.println(++result);
      }
   }

   public boolean  canMoveRight(int r) {
      return r <   court;
      }

   public boolean  canMoveLeft(int l) {
      return l  > 1;
      }
```

```
   public boolean  isHit(int l, int r, int  ball) {
      return (ball >= l   && ball <= r);
      }

   public void  getInputs()  {
      Scanner kb = new Scanner(System.in);
      System.out.print("Enter the field values <court player balls>: ");
      court  = kb.nextInt();
      prblock   = kb.nextInt();
      balls =   new int[kb.nextInt()];
      System.out.print("Enter balls separated by spaces: ");
      for (int  i = 0; i  < balls.length; i++)   {
         balls[i] = kb.nextInt();     }
         }

   public void  play(int  index, int left, int   right) {
      if (index <  balls.length && !isHit(left, right, balls[index]))   {
         index++;
         play(index,  left, right);
         if (canMoveRight(right)) {
            play(index,  left + 1, right + 1);
            }
         if (canMoveLeft(left)) {
            play(index, left- 1, right - 1);
            }
         }
         else {
            if  (result < index) {
               result = index; }
            }
      }

   public static void main(String[]   args) {
      new Dodgeball();
      }
}
```

[Recursive solution produced by Shamiel Dramat]

PASCAL SOLUTION

```
program DodgeBall;

Uses sysutils;

Type
  BallArray = Array[1..50] of integer;

Var
  ball : BallArray;
  ballpos : string;
  courtWidth, playerWidth, numBalls : integer;
  i, j : integer;
  NKL : string;
  result : integer = 0;

function canMoveRight(r, courtWidth: integer): boolean;
  begin
    canMoveRight := (r < courtWidth);
  end;

function canMoveLeft(l: integer): boolean;
  begin
     canMoveLeft := (l > 1);
  end;

function isHit(l, r, ballInPlay: integer): boolean;
  begin
    isHit := ((ballInPlay >= l) and (ballInPlay <= r));
  end;

procedure play(index, left, right: integer);
  begin
    if (index < numBalls)and (not(isHit(left, right, ball[index]))) then
      begin
        index += 1;
        play(index,   left, right);
        if (canMoveRight(right, courtWidth)) then
          play(index,  left + 1, right +    1);
        if (canMoveLeft(left)) then
          play(index, left   - 1, right - 1);
      end
    else
      begin
        if   (result < index) then
          result := index;
      end;
  end;
```

```
procedure getInputs;

  begin
    write('Enter values for N, K and L : ');
    readln(NKL);
    if (NKL = '') then
      NKL := '40 5 36';
    write('Enter ball positions: ');
    readln(ballpos);
    if (ballpos = '') then
      ballpos := '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';
    i := pos(' ', NKL);
    courtWidth := strtoint(copy(NKL, 1, i-1));
    NKL := copy(NKL, i+1, length(NKL));
    i := pos(' ', NKL);
    playerWidth := strtoint(copy(NKL, 1, i-1));
    NKL := copy(NKL, i+1, length(NKL));
    numBalls := strtoint(NKL);
    for i := 1 to numBalls-1 do
      begin
        j := pos(' ', ballpos);
        ball[i] := strtoint(copy(ballpos, 1, j-1));
        ballpos := copy(ballpos, j+1, length(ballpos));
      end;
    ball[i+1] := strtoint(ballpos);
  end;

begin
  getInputs;
  play(1, 1, playerWidth);
   play(1, 2, playerWidth + 1);
   if(result = numBalls)then
      writeln('Complete')
      else
      writeln(result);
  readln;
end.
```

[Solution produced by OER Foss, based on the recursive algorithm used in the Java solution]

PYTHON SOLUTION

```
import sys
n, k, l=map(int,raw_input().split())
shots = map(int,raw_input().split())
pos=[[0 for i in xrange(n)]] # pos[i][j]=1 iff Aphiwe can be at position j
# after the ith second
pos[0][0] = 1
print "pos : ", pos
for i in xrange(l):
    shots[i] -= 1
    print "shots : ", shots
    new = [0 for j in xrange(n)] # the next row of pos
    print "new : ", new
    for j in xrange(n):
        if(pos[-1][j] == 1):
            new[j] = 1
            if j > 0:
                new[j-1] = 1
            if j < n - k:
                new[j+1] = 1
    for j in xrange(max(0, shots[i] - k + 1),min(n, shots[i] + 1)):
        new[j] = 0
    pos.append(new)
for i in xrange(1, l+1):
    if sum(pos[i]) == 0: # if pos[i] is all 0s
        print i
        sys.exit(0)
print "Complete"
```

[Solution produced by Sean Wentzel]