# Worksheet 4 Testing Answers

**Task 1**

1. (a) Write an algorithm to allow a user to input the maximum and minimum daily temperatures for a number of days until a maximum temperature of 999 is entered.

The program then calculates the average maximum daily temperature and outputs the number of days that this temperature was above average. It also outputs the number of days that the temperature was negative.

maxTemp = USERINPUT

initialise maxArray as an empty array

initialise minArray as an empty array

totalMaxTemp = 0

numDays = 0

daysAboveAverage = 0

daysNegative = 0

WHILE maxTemp <> 999

minTemp = USERINPUT

append maxTemp to maxArray

append minArray to minArray

numDays = numDays + 1

totalMaxTemp = totalMaxTemp + maxtemp

IF minTemp < 0 THEN

daysNegative = daysNegative + 1

ENDIF

maxTemp = USERINPUT

ENDWHILE

averageMaxTemp = totalMaxTemp / numDays

daysAboveAverage = 0

FOR n = 1 to numDays

IF maxArray(n) > averageMaxTemp THEN

daysAboveAverage = daysAboveAverage + 1

ENDIF

ENDFOR

OUTPUT (“Days max temp above average ”,daysAboveAverage)

OUTPUT (“Days min temperature below zero ”, daysNegative)

(b) Write a test plan to test the program.

**Test plan**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | Test data | Reason for test | Expected resut | Actual result |
| 1 | Max:10,12, 6, 0,-1,999  Min: -1, -2, 3, 0, 5 | Test that 0 is processed correctly  Test that average max is calculated correctly  Test that minimums are counted correctly | Average = 27/5=5.4  Days above average = 3  Days below zero = 2 | Days above average = 3  Days below zero = 2 |
| 2 | Max: 5, 5, 5, 5, 999  Min: 1, 2, 3, 4 | Test for all max temps the same  Test for no mins <0 | Average = 5  Days above average = 0  Days below zero = 0 | Days above average = 0  Days below zero = 0 |
| 3 | Max = 0, 0, 0, 5, 999  Min = -1, -1, -2, -2 | Test all temps below zero | Days above average = 1  Days below zero = 4 | Days above average = 1  Days below zero = 4 |
| 4 | Max: 999 | Test for first temp = 999 | Program crashes | Program crashes, division by 0 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Note that invalid data is not tested here as validation is not being tested.

**Task 2**

2. Use the trace table below to help you answer (a), (b) and (c) below.

What would be the values of integer variables x, y and z after execution of these statements if the initial values of x and y are

(a) 2 and 7 2 200 47

(b) -4 and -4 16 6 -4

(c) 27 and 3 27 200 9

**Trace table**

|  |  |  |
| --- | --- | --- |
| **x** | **y** | **z** |
| 2 | 7 | 2 |
|  | 49 | 47 |
|  | 200 |  |
| -4 | -4 | -4 |
| 16 | 6 | -4 |
| 27 | 3 | 27 |
|  |  | 9 |
|  | 200 |  |

z = x

IF x = y THEN

x = x \* x

y = (x + y) / 2

ELSE

IF x < y THEN

y = y \* y

z = y - x

ELSE

IF x > 0 THEN

z = x/y

ENDIF

ENDIF

y = 200

ENDIF

OUTPUT x, y, z

3. Use a trace table to determine the output from the following algorithm.

x = 5

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **K** | **Sum** | **OUTPUT** |
| 5 | 10 | 45 |  |
|  |  | 55 | 10 |
|  | 15 | 70 | 15 |
|  | 20 | 90 | 20 |
|  | 25 |  | Sum = 90 |

k = 10

sum = 45

WHILE sum < 75

sum = sum + k

OUTPUT k

k = k + x

ENDWHILE

OUTPUT sum

(See Python program **Worksheet 4 trace tables**)

4. Study the following algorithm and fill in the trace tables below to discover what it does.

y = 2

z = 1

OUTPUT ("Please enter a positive integer: ")

x = USERINPUT

WHILE z<>0

z = x mod y

IF z <> 0 THEN

y = y + 1

ENDIF

ENDWHILE

IF x = y

print (x, " is in category 1")

ELSE

print (x, " is in category 2")

ENDIF

|  |  |  |
| --- | --- | --- |
| **x** | **y** | **z** |
| 25 | 2 | 1 |
|  | 3 | 1 |
|  | 4 | 1 |
|  | 5 | 0 |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **x** | **y** | **z** |
| 7 | 2 | 1 |
|  | 3 | 1 |
|  | 4 | 3 |
|  | 5 | 2 |
|  | 6 | 1 |
|  | 7 |  |

(i) If the user inputs the integer 25, what is output? OUTPUT: x is in category 2

(ii) If the user enters the integer 7, what is output? OUTPUT: x is in category 1

(iii) What are “category 1” and “category 2”? What is the purpose of the program?

The program tests if the number input is a prime number. “Category 1” is prime, “category 2 is non-prime.

(iv) Suggest ways in which the program could be made easier to understand.

The program would be easier to understand if comments were added to explain the purpose of the program and the way it works. Instead of “category 1” and “category 2” the output should say, “This is a prime number” or “This is not a prime number”.

Meaningful variable names would also help – y is *divisor*, x mod y could be put in a variable called *remainder*.

(v) This is a “brute force” algorithm. Suggest how the algorithm could be made more efficient.

Once you have found that the number is not divisible by 2, there is no point testing for other even numbers. Likewise, once you have tested for any other number like 3, 5, etc there is no point testing for multiples of that number.

See Python program Worksheet 4 primes.py