Anglia Ruskin University

Practical: Loops and Arrays

1. Design and write a program to read int values from the keyboard until a value of -999 is entered. The program should calculate the average (as a double) of all the numbers except the terminating value (the -999 terminating value is known as a *sentinel*). You will need both a loop and an if/else structure.

2. Write a program utilising a loop structure to solve a factorial number input by the user. For example, if the user input 5, the factorial of 5 (symbolized as 5!) would be 5x4x3x2x1 = 120. The solution is a quite short bit of coding but it is tricky to implement. Hint: it's easier if you can identify the algorithm to do this first before writing the program.

3. One important operation in data processing is the ability to sort data. There are various ways of doing this. One such way is a method called BubbleSort. BubbleSort works by repeatedly looking at each element in an array of initially unsorted data sequentially from left-to-right, and if the current number is bigger than the next number then the numbers are swapped around. This means that if a large number is present towards the left of the array then it will be repeatedly shifted to the right until it lies before a larger number. Repeated scans through the array in this manner cause the numbers to 'bubble up' from left–to-right until they are in size order. The algorithm (a recipe to do something) is given below; note this is given in pseudocode (not real code):

```
data[ ] // values in data[ ] are unsorted
length = length of array
for i=0 to length
{
    for j=0 to length-1
    {
        if ( data[ j ] > data[ j+1 ] )
        {
            temp = data[ j ]
            data[ j ] = dat [ j+1 ]
            data[ j+1 ] = temp
        }
    }
}
// values in data[ ] are now sorted
```

The algorithm uses a 'nested' loop. In the inner loop the 'j' value refers to a position in the array, whilst the 'i' value in the outer loop refers to a number reflecting the current processing of the whole array. Another way of explaining this is> ‘Scan the array a number of times, and for each scan compare adjacent pairs of values swapping them as necessary’.

a) Write a program that initialises an integer array containing the numbers 15, 68, 4, 19, 99, 52, 53, 36, 74, 1, 85.

b) Then add a loop which passes through the array and prints each number on one line separated by two spaces. Note Console.WriteLine() implements a newline at the end of output, whilst Console.Write() leaves the cursor on the same line.

c) When b) is working, copy the code of b) and paste it a few lines below. So now when the code is run it should print to screen two rows, both identical and both reflecting the contents of the array.

d) Now add code between the code of b) and c) that will sort the data in the array into ascending order based on the BubbleSort algorithm. When implemented correctly the first row that prints to screen will show the data unsorted whilst the second row will show the numbers sorted following the processing by the BubbleSort algorithm. Since the BubbleSort algorithm above is only an approximation of computer code (it is pseudocode) you will need to modify it so that it is in a correct C# form. Note a common error is to be ‘out-by-one’ when parsing (reading) through the elements of an array.

DEMO COMPLETED QUESTION 3 TO TUTOR