QUESTION 1 **(Total 10 marks)**

(a) A year is a "leap year" if it is divisible by 4, but *not* if it is divisible by 100 unless it is also divisible by 400.

This means century years such as 1800, 1900 and 2100 **are not** leap years (because they are divisible by 4 and 100 but **not** divisible by 400)

Century years 2000 and 2400 **are** leap years (because they are divisible by 4, 100 and 400)

Years 2012 and 2016 **are** leap years (because they are divisible by 4, but not divisible by 100)

Years 2013, 2014 and 2015 **are not** leap years (because they are not divisible by 4).

Write a method to determine whether a given year is a leap year, using the following method signature:

```
public static bool IsLeapYear(int year)
```

(4 marks)

(b) (i) Write an algorithm (a set of clearly defined, unambiguous steps) to solve the following problem:

Given a non-empty array of integers with no duplicates (ie all values are unique), find out if the array elements are in either ascending OR descending sorted order.

Some test data with expected results is given below:

```
{1, 3, 6, 7, 8}  - true
{8, 7, 6, 2}     - true
{8 , 2, 3, 4, 6} - false
{1, 5, 4, 6, 2}  - false
```

Remember that a collection of only 1 or 2 values is already sorted.

(2 marks)

(ii) Now implement your algorithm by writing the method using the following heading:

```
public static bool IsArraySorted(int[] numbers)
```

(4 marks)

QUESTION 2 **(Total 10 marks)**

(a) Write a method to reverse the elements in an array between a lower index position and an upper index position. Use the method signature below:

```
// pre: start and finish are valid indexes
// within the bounds of the array
public void Reverse(double[] values, int start, int finish)
```

Given the following array declaration:

```
    double[] data = { 8.5, 12.9, 23.2, 18.1, 15.5, 5.2, 10.5 };
```

following a call to the method:

```
    Reverse(data, 2, 5);
```

the contents of `data` would be:

```
    { 8.5, 12.9, 5.2, 15.5, 18.1, 23.2, 10.5 }
```

Assume you have already written a method called `swap` that swaps two elements in an array; the elements identified by the two index values passed as parameters. This method has the following method signature:

```
public void Swap(double[] array, int oneIndex, int otherIndex)
```

You should *use* this method in your solution. **(5 marks)**

(b) Consider the following C# declarations and initialisations, representing the results for one particular student over four semesters of study. Each array in the array of arrays (or 'jaggered array') represents one semester's results. You will notice that the student did not complete the same number of units each semester.

```
        // An array of arrays to store grades for 4 semesters
        int[][] grades = new int[4][];
        // record the 4 semester results (grades out of 7)
        grades[0] = new int[] { 6, 5, 4, 7 };
        grades[1] = new int[] { 5, 5, 6 };
        grades[2] = new int[] { 4, 4, 5, 6 } ;
        grades[3] = new int[] { 6 };
```

Write code to calculate the GPA (grade point average) for this student. The GPA is calculated by adding each mark and dividing by the number of marks. For example, using the above values, the GPA for this student would be calculated to be 5.25. However, your code must work for ANY student results for any number of semesters.

(5 marks)

QUESTION 3 **(Total 10 marks)**

(a) The following UML diagram describes a class `Animal` which is to be used as part of an inventory system at a pet shop. Assume this class has already been implemented in C#.

Animal (abstract)
-age: int

+Animal(age: int)
+Age: int {property}
+HaveBirthday()
+ToString(): string

The following UML diagram describes a class `Dog` which is to be used in the same inventory system. This class extends the above `Animal` class.

Dog {extends **Animal**}
-inventoryNumber : int -breed : string -colour : string -totalNumberOfDogs : int

+Dog()
+Dog(typeOfBreed : string, age : int, colour : string)
+InventoryNumber : int {read-only property}
+Breed : string {read-only property}
+Colour : string {read+write property}
+ToString() : string

Using the UML diagrams, your task is to complete the `Dog` class in C#.

Further information:

The purpose of `totalNumberOfDogs` is twofold:

- it stores the current number of `Dog` objects created;
- it is used to generate the inventory number for the next `Dog`.

Before any `Dog`s are created, the `totalNumberOfDogs` should be 0. When the first `Dog` is created, that number should be incremented, and used as the `inventoryNumber` of the new `Dog`.

The default constructor is to generate an exception of type `NotImplementedException` with an appropriate message to say that it is not permissible to create a `Dog` object without specifying the breed and age of the dog).

In addition, `breed` and `inventoryNumber` will never change.

(5 marks)

QUESTION 3 (cont'd)

(b) After the following C# code has been executed, what is the value of `num`? Provide a trace table to show the changing values of the variables after each line of code is executed.

```
...
int[] myArray = {200, 0, 100, -100, 0, 300};
int num = 5;
bool found = false;

while (!found) {
     if (myArray[num] == 0) {
          found = true;
     }
     num--;
     }
```

(2 marks)

(c) Consider the following method named `Mystery`. In one or two sentences describe what this method does. Do NOT give a line by line description of the code. Instead provide a summary of the method's overall purpose. For example in what circumstances might the method be useful? Your answer should be similar to that expected in a method comment.

In your answer, provide a test call to the method, and a trace table to show the changing values of the variables after each line of code is executed.

```
public int Mystery(int[] data, int x ){
    int index;
    int found = -1;

    for (index = 0; index < data.Length; index++ ){
        if(data[index] == x){
            found = index;
        }
    }
    return found;
}// end Mystery
```

(1 mark)

(d) In the following loop, how many times is the output statement executed? Provide a trace table to show the changing values of the variables after each line of code is executed.

```
for (int i = 0; i < 4; i++) {
    for (int j = i; j < 4 - i; j++) {
        Console.WriteLine("output"); // output statement
    }
}
```

(2 marks)

END OF PAPER

EXTRA PRACTICE QUESTION **(Total 5 marks)**

(a) Errors in keyboard data entry are quite common. Some sources of errors are:

- typing errors ; pressing 8 instead of 7
- transcription errors: reading a 7 as a 1, a 5 as a 6,
- transposition errors: entering 12 instead of 21

A commonly used approach to the validation of numeric data is the **modulo-11** method.

Given a 9 digit number, eg 374839018, the first digit (3) is multiplied by 9, the second digit (7) is multiplied by 8, and the third digit (4) is multiplied by 7, and so on. All of these products are added together and the total is divided by 11 (hence the name “modulo-11). If the remainder is zero the number is valid.

For the number, 374839018, the calculation is:

```
3 x 9 = 27
7 x 8 = 56
4 x 7 = 28
8 x 6 = 48
3 x 5 = 15
9 x 4 = 36
0 x 3 =  0
1 x 2 =  2
8 x 1 =  8
Total   220
```

220 % 11 == 0. As the remainder is zero, this number passes the **modulo-11** validation test.

Write a Boolean method which, given a nine (9) digit number, determines if the number passes this **modulo-11** validation check. Use the following method heading. (There is no need to include the XML comment in your answer).

```
/// <summary>
/// Performs a modula-11 validation check on number
/// </summary>
/// <param name="number">a 9 digit positive integer</param>
/// <returns> true if modula-11 check is successfully
///           otherwise returns false
///</returns>
static bool NumberValid(int number){
```

(5 marks)