Exercise 1

Apply the basis path technique to design the minimum test case required to testing the following method. This static method applies the binary search to find a character c in an array of characters v. The method returns 1 if it found the character, 0 otherwise. The input vector is sorted ascending by the ASCII code.

a) draw the correctly labelled flow graph
b) calculate the cyclomatic complexity:
   I. provide the number of regions
   II. provide the number of nodes
   III. provide the number of predicate nodes
   IV. provide the number of edges
c) specify the independent paths
d) provide the test cases associated with the independent paths

```
static public int search(char c, char []v)
        {
            int a, z, m;
            a = 0;
            z = v.Length - 1;
            while (a <= z)
            {
                m = (a + z) / 2;
                if (v[m] == c) {
                    return 1;
                }
                else if(v[m] < c)
                {
                    a = m + 1;
                }
                else
                {
                    z = m - 1;
                }
            }
            return 0;
        }
```

Exercise 2

Apply the basis path technique to design the minimum test case required to testing the following method. This static method sorts an array of integers in an ascending way.

a) draw the correctly labelled flow graph
b) calculate the cyclomatic complexity:
   I. provide the number of regions
   II. provide the number of nodes
   III. provide the number of predicate nodes
   IV. provide the number of edges
c) specify the independent paths
d) provide the test cases associated with the independent paths

```
static public void sort(int[] testArray)
{
            int tempValue;
            int i = 0;
            bool isSwapped = true;
            while (isSwapped)
            {   isSwapped = false;
                i++;
                Console.Out.WriteLine("Before "+i+" iteration :");
                Console.Out.WriteLine("");
                for (int j = 0; j < testArray.Length - i; j++)
                {
                    if (testArray[j] > testArray[j + 1])
                    {
                        tempValue = testArray[j];
                        testArray[j] = testArray[j + 1];
                        testArray[j + 1] = tempValue;
                        isSwapped = true;
                    }
                }
            }
}
```