20CAPF0 - Programming in C# using .Net – Assignment II

Name : BALAJI J

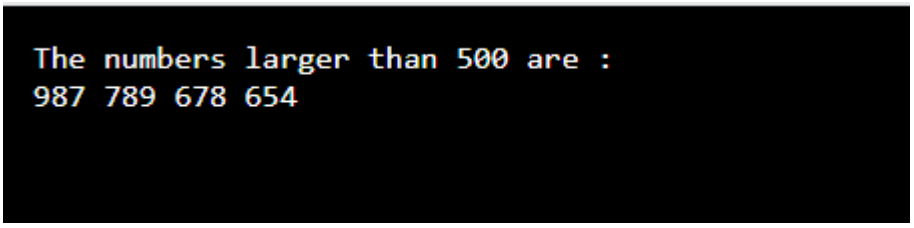
Reg. no : 20Y005

1. Assume that you have an array of integers. Apply the concept of LINQ to retrieve those numbers which are greater than a user-given input value. Also print the result in the descending order of the values.

   Program:

```
using System;
using System.Linq;
class Lin{
static void Main()
{
        int[] Arr = { 1, 234, 456, 678, 789, 987, 654, 345 };

        var numbers = from number in Arr
              where number > 500
              orderby number descending
              select number;

        Console.WriteLine("The numbers larger than 500 are :");
        foreach(int n in numbers)
        {
               Console.Write(n + " ");
        }
}
}
```

Output:



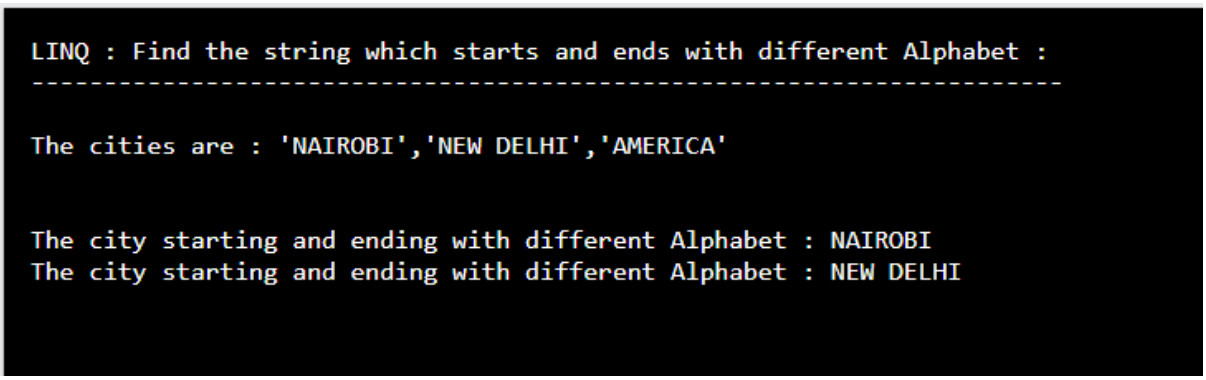
2. You are given an array of strings are required to retrieve those strings start with one alphabet and end with yet another alphabet. The user is expected to supply those values. Also, arrange the result in ascending order. Apply the concept of linq for your operation.

Program:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

class Lin
  {
     static void Main(string[] args)
   {
        string[] cities =
        {
           "NAIROBI","NEW DELHI","AMERICA"
        };

        Console.Write("\nLINQ : Find the string which starts and
ends with different Alphabet : ");
```

```
        Console.Write("\n----------------------------------------------------------\n");
        Console.Write("\nThe cities are : 'NAIROBI','NEW DELHI','AMERICA'\n");

        var _result = from x in cities
        where x.StartsWith('N')
        where x.EndsWith('I')
        orderby x ascending
        select x;
        Console.Write("\n\n");
        foreach(var city in _result)
        {
           Console.Write("The city starting and ending with different Alphabet : {0} \n", city);
        }

        Console.ReadLine();
     }
  }
```

Output:



3. Illustrate the concept of Generic classes and develop a c# application for the implementation of queue

Concepts of Generic classes:

Generic classes have type parameters. Separate classes, each with a different field type, can be replaced with a single generic class.

A generic class introduces a type parameter (often specified as the letter T). This becomes part of the class definition itself. Generic methods can also be designed.

Generic class example. To start, we specify a generic type. These types have type parameters. When compiled, the type parameters refer to the type specified.

Ex: Test<int>
T = int

The letter T denotes a type that is only known based on the calling location. The program can act upon T like it is a real type. We use an int type parameter with the Test class. The T is substituted with an int.

Program:

```
using System;
using System.Collections.Generic;

class QueueEx
{
public static void Main(String[] ar)
{
```

```
        /*Creating a Queue<T> of type int i.e. Queue<int> to hold int values. Where each int value is implicitly converted to an Object.*/

        Queue<int> dq = new Queue<int>();

        /*Calling the Enqueue<T>() method to push elements into the Queue<int>. New element is always added to the end of the Queue<int>.*/

        dq.Enqueue(10);
        dq.Enqueue(23);
        dq.Enqueue(16);
        dq.Enqueue(5);
        dq.Enqueue(29);

        //Printing the contents of Queue<int>
        Console.WriteLine("The contents of Queue<int>: ");
        foreach(int element in dq)
                Console.WriteLine(element);


        //Calling the Dequeue() method
        Console.WriteLine("\nRemoving the front element = "+ dq.Dequeue());
        Console.WriteLine("Removing the next front element = "+ dq.Dequeue());


        //Calling the Peek() method
```

```
        Console.WriteLine("\nPeeking at the current front
element = "+ dq.Peek());


        //Printing the updated contents of Queue<int>
        Console.WriteLine("\nUpdated contents of Queue<int>:
");
        foreach(int element in dq)
                Console.WriteLine(element);
}
}
```

Output: