1) Create a custom exception class named StackException. The Push()and Pop() method should throw object of StackException when the stack is full or empty respectively.

StackException.cs

```
using CCA;
using System;

public class demo
{
static int MAX = 5;
int[] st = new int[MAX];
int top = -1;
bool isempty()
{

if (top == -1)
return true;
else
return false;
}

public void Push(int data)
{
try
{
if (top > MAX)
{
throw new StackException("Stack Overflow");

}
else
{
st[++top] = data;
top++;
}
}
catch(Exception e)
{
Console.WriteLine(e.Message);
}
}

public void Pop()
{
int data;
try
{
if (!isempty())
{
data = st[top];
top = top - 1;
}
else
{
throw new StackException("Stack Underflow");

}
}
catch(Exception e)
{
Console.WriteLine("Stack Underflow");
}

}

public static void Main(String[] args)
{
//Employee emp = new Employee();
//emp.EmployeeDetails();

int ch;
demo se = new demo();

do
{
Console.WriteLine("\n1.Push\n2.Pop\n3.Exit");
Console.WriteLine("\nEnter the operation");
ch = Convert.ToInt32(Console.ReadLine());
switch (ch)
{
case 1:
Console.WriteLine("Enter the value");
int data = Convert.ToInt32(Console.ReadLine());
se.Push(data);
break;
case 2:
se.Pop();
break;
case 3: break;
}
}
while (ch != 3);}}
```

program.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Linq;
```

```
namespace CCA
{
public class StackException : Exception
{
public StackException() { }

public StackException(string message) : base(String.Format(message))
{

}
}
}
```