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

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication1

{

class Stack : ICloneable

{

private int[] ele;

private int top;

private int max;

public Stack(int size)

{

ele = new int[size];

top = -1;

max = size;

}

public void push(int item)

{

try

{

ele[++top] = item;

}

catch (Exception Stackexception)

{

throw new StackOverflowException(GetType().Name);

}

}

public int pop()

{

try

{

Console.WriteLine("Poped element is: " + ele[top]);

return ele[top--];

}

catch (Exception Stackexception)

{

throw new InsufficientExecutionStackException("stack underflow exception");

}

}

public void printStack()

{

if (top == -1)

{

Console.WriteLine("Stack is Empty");

return;

}

else

{

for (int i = 0; i <= top; i++)

{

Console.WriteLine("Item[" + (i + 1) + "]: " + ele[i]);

}

}

}

public object Clone()

{

return this;

}

}

class Program

{

public static void Main()

{

try

{

Console.WriteLine("enter length of the Stack");

int length = Convert.ToInt32(Console.ReadLine());

Stack S = new Stack(length);

again:

Console.WriteLine("do you want to push or pop");

string choice = Console.ReadLine();

{

if (choice.ToUpper() == "POP")

{

S.pop();

goto again;

}

else if (choice.ToUpper() == "PUSH")

{

int num = Convert.ToInt32(Console.ReadLine());

S.push(num);

goto again;

}

else

{

S.printStack();

}

}

S.printStack();

S.Clone();

}

catch (Exception ex)

{

Console.WriteLine(ex.GetType().Name);

}

}

}

}

