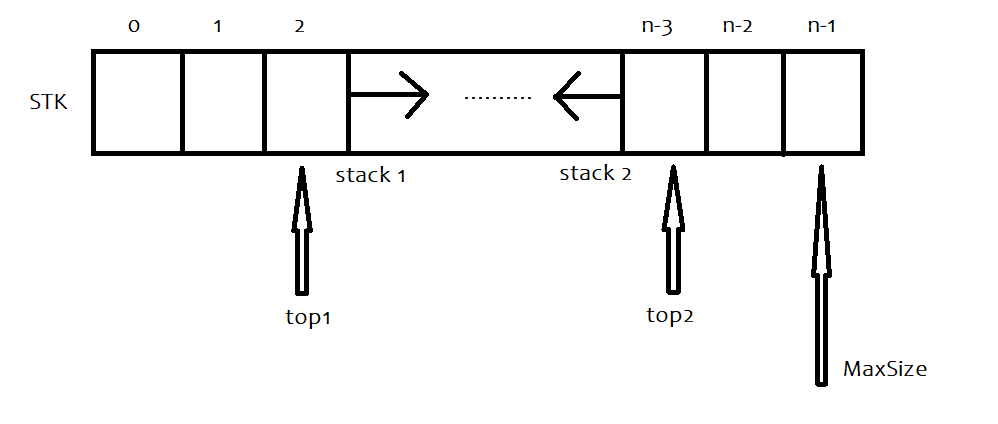
**I Assess(Stack)**

**Question:**

1.Write a C  program to implement push and pop operation on stack and to display the contents of the stack.  
  


Create a data structure twoStacks that represents two stacks. Implementation of twoStacks should use only one array, i.e., both stacks should use the same array for storing elements. Following functions must be supported by twoStacks.

push1(int x) –> pushes x to first stack  
push2(int x) –> pushes x to second stack

pop1() –> pops an element from first stack and return the popped element  
pop2() –> pops an element from second stack and return the popped element  
  
display1() -> displays the contents of first stack  
display2() -> displays the contents of second stack.  
  
printMenu() -> displays the choices.

Consider implementing a dynamically sized stack using an array.

**Create a structure**

struct twoStacks

{

    int \* arr;

    int top1, top2;

    int size;

};

The structure stack has a pointer 'arr' to a dynamically allocated array (used to hold the contents of the stack), an integer 'size' that holds the size of this array (i.e the maximum number of data that can be held in this array) and an integer 'top1' which stores the array index of the top element in the stack1 and an integer 'top2' which stores the array index of the top element in the stack2.  
  
In the initstack function intialize the value of top1 to -1 and top2 to size, and initialize the value of size.

Print the message “Stack1 is full” in the push1 function when an attempt is made to push a data into a full stack1.  
Print the message “Stack2 is full” in the push2 function when an attempt is made to push a data into a full stack2.

Print the message “Stack1 is empty” in the pop1 function and return the value -1000 when an attempt is made to pop data from an empty stack1.  
Print the message “Stack2 is empty” in the pop2 function and return the value -1000 when an attempt is made to pop data from an empty stack2.  
Print the contents of Stack1 "The contents of the stack1 are" in the display1 function, and when stack is empty display "{}".  
Print the contents of Stack2 "The contents of the stack2 are" in the display2 function, and when stack is empty display "{}".

**Function specifications:**

|  |
| --- |
| void initstack(struct twoStacks \* p, int size) |
| void push1(struct twoStacks \* p, int item) |
| void push2(struct twoStacks \* p, int item) |
| int pop1(struct twoStacks \* p) |
| int pop2(struct twoStacks \* p) |
| void display1(struct twoStacks p) |
| void display2(struct twoStacks p) void printMenu() |

**Input and Output Format:**

Refer sample input and output for formatting specifications.

**Note: All text in bold corresponds to input and the rest corresponds to output.**

**Sample Input and Output:**

Enter the maximum size of the stack

**5**

Choice 1 : Push1  
Choice 2 : Push2  
Choice 3 : Pop1  
Choice 4 : Pop2  
Choice 5 : Display1  
Choice 6 : Display2  
Any other choice : Exit  
Enter your choice

**1**

Enter the element to be pushed in stack1

**5**

Choice 1 : Push1

Choice 2 : Push2  
Choice 3 : Pop1  
Choice 4 : Pop2  
Choice 5 : Display1  
Choice 6 : Display2  
Any other choice : Exit

Enter your choice

**2**

Enter the element to be pushed in stack2

**10**

Choice 1 : Push1

Choice 2 : Push2  
Choice 3 : Pop1  
Choice 4 : Pop2  
Choice 5 : Display1  
Choice 6 : Display2  
Any other choice : Exit

Enter your choice

**2**

Enter the element to be pushed in stack2

**15**

Choice 1 : Push1

Choice 2 : Push2  
Choice 3 : Pop1  
Choice 4 : Pop2  
Any other choice : Exit

Enter your choice

**1**

Enter the element to be pushed in stack1

**11**

Choice 1 : Push1

Choice 2 : Push2  
Choice 3 : Pop1  
Choice 4 : Pop2  
Choice 5 : Display1  
Choice 6 : Display2  
Any other choice : Exit

Enter your choice

**2**

Enter the element to be pushed in stack2  
**7**

Choice 1 : Push1

Choice 2 : Push2  
Choice 3 : Pop1  
Choice 4 : Pop2  
Choice 5 : Display1  
Choice 6 : Display2  
Any other choice : Exit

Enter your choice

**1**

Enter the element to be pushed in stack1  
**45**  
Stack1 is full

Choice 1 : Push1

Choice 2 : Push2  
Choice 3 : Pop1  
Choice 4 : Pop2  
Choice 5 : Display1  
Choice 6 : Display2  
Any other choice : Exit

Enter your choice

**5**

The contents of the stack1 are 5 11

Choice 1 : Push1

Choice 2 : Push2  
Choice 3 : Pop1  
Choice 4 : Pop2  
Choice 5 : Display1  
Choice 6 : Display2  
Any other choice : Exit

Enter your choice

**6**

The contents of the stack2 are 10 15 7  
Choice 1 : Push1

Choice 2 : Push2  
Choice 3 : Pop1  
Choice 4 : Pop2  
Choice 5 : Display1  
Choice 6 : Display2  
Any other choice : Exit

Enter your choice

**3**  
The popped element from stack1 is 11  
Choice 1 : Push1

Choice 2 : Push2  
Choice 3 : Pop1  
Choice 4 : Pop2  
Choice 5 : Display1  
Choice 6 : Display2  
Any other choice : Exit

Enter your choice

**2**  
Enter the element to be pushed in stack2  
**45**

Choice 1 : Push1

Choice 2 : Push2  
Choice 3 : Pop1  
Choice 4 : Pop2  
Choice 5 : Display1  
Choice 6 : Display2  
Any other choice : Exit

Enter your choice  
**3**  
The popped element from stack1 is 5

Choice 1 : Push1

Choice 2 : Push2  
Choice 3 : Pop1  
Choice 4 : Pop2  
Choice 5 : Display1  
Choice 6 : Display2  
Any other choice : Exit

Enter your choice  
**3**  
Stack1 is empty

Choice 1 : Push1

Choice 2 : Push2  
Choice 3 : Pop1  
Choice 4 : Pop2  
Choice 5 : Display1  
Choice 6 : Display2  
Any other choice : Exit

Enter your choice  
**5**  
The contents of the stack1 are {}

Choice 1 : Push1

Choice 2 : Push2  
Choice 3 : Pop1  
Choice 4 : Pop2  
Choice 5 : Display1  
Choice 6 : Display2  
Any other choice : Exit

Enter your choice  
**4**  
The popped element from stack2 is 45  
Choice 1 : Push1  
Choice 2 : Push2  
Choice 3 : Pop1  
Choice 4 : Pop2  
Choice 5 : Display1  
Choice 6 : Display2  
Any other choice : Exit  
Enter your choice  
**7**

**PROGRAM**

import java.io.\*;  
import java.util.Scanner;  
class Stack1{  
    int top=-1;  
    int size;  
    int[] arr;  
      public Stack1(int size){  
        this.size=size;  
        arr=new int[size];  
         
    }  
    public void push1(int data){  
        if(top==size-1){  
            System.out.println("Stack is Full");  
        }else{  
            top++;  
            arr[top]=data;  
        }  
    }  
    public void pop1(){  
        if(top==-1){  
            System.out.println("Stack is empty");  
        }else{  
            System.out.println("The popped element from stack1 is"+arr[top]);  
            int temp=top;  
            top--;  
            temp=0;  
        }  
    }  
    public void display1(){  
        if(top==-1){  
            System.out.println("Stack is Empty");  
        }else{  
            for(int i=0;i<=top;i++){  
                 
                System.out.println(arr[i]);  
            }  
        }  
    }  
}  
  
class Stack2{  
    int top=-1;  
    int size;  
    int[] arr2;  
      public Stack2(int size){  
        this.size=size;  
        arr2=new int[size];  
           
    }  
    public void push2(int data2){  
        if(top==size-1){  
            System.out.println("Stack is Full");  
        }else{  
            top++;  
            arr2[top]=data2;  
        }  
    }  
    public void pop2(){  
        if(top==-1){  
            System.out.println("Stack is empty");  
        }else{  
            System.out.println("The popped element from stack1 is"+arr2[top]);  
            int temp=top;  
            top--;  
            temp=0;  
        }  
    }  
    public void display2(){  
        if(top==-1){  
            System.out.println("Stack is Empty");  
        }else{  
             
            for(int i=0;i<=top;i++){  
                   
                System.out.println(arr2[i]);  
            }  
        }  
    }  
}  
  
public class Main{  
    public static void main(String args[]){  
        Scanner sc=new Scanner(System.in);  
         int ch,size,data,data2;  
        System.out.println("Enter the maximum size of the stack");  
       size=sc.nextInt();  
        Stack1 s1=new Stack1(size);  
        Stack2 s2=new Stack2(size);  
        do{  
       System.out.print("Choice 1 : Push1\n"+  
                        "Choice 2 : Push2\n"+  
                        "Choice 3 : Pop1\n"+  
                        "Choice 4 : Pop2\n"+  
                        "Choice 5 : Display1\n"+  
                        "Choice 6 : Display2\n"+  
                        "Any other choice : Exit\n"+  
                        "Enter your choice\n");  
        ch=sc.nextInt();  
        switch(ch){  
            case 1:  
                System.out.println("Enter the element to be pushed in stack1");  
                data=sc.nextInt();  
                s1.push1(data);  
                break;  
            case 2:  
                System.out.println("Enter the element to be pushed in stack2");  
                data2=sc.nextInt();  
                s2.push2(data2);  
                break;  
            case 3:  
                s1.pop1();  
                break;  
            case 4:  
                s2.pop2();  
                break;  
            case 5:  
                 System.out.println("The contents of the stack1 are: ");  
                s1.display1();  
                break;  
            case 6:  
                 System.out.println("The contents of the stack2 are: ");  
                s2.display2();  
                break;  
        }  
    }while(ch!=6);  
    }  
}