R. Darshan Nagpal
RVCE24MCA011

2 STACK IN AN ARRAY

Implement 2 stacks using one array, it can be done by the following steps.

Step 1 : Maintain 2 pointers for both the stack top elements.
top1 : for stack1 (starting from 0 index).
top2 : for stack2 (Starting for n-1 index).
1st stack will start from begining.
2nd stack will start from last.

Step 2: Since stacks cannot overlap in one single array, we will have to see whether top1 < top2.

Stacks have the concept of overflow and underflow.

→ program : (C++)

```
#include<iostream.h>
#include<stdlib.h>

class twoStacks
{
    int* arr;
    int size;
    int top1, top2;
```

```
public:
    twoStacks(int n)
    {
        size = n;
        arr = new ~~int~~ int[n];
        top1 = -1;
        top2 = n-1;
    }

    void push1(int x)
    {
        if(top1 < (top2-1))
        {
            top1++;
            arr[top1] = x;
        }
        else
        {
            cout << "Stack Overflow";
            exit(0);
        }
    }

    void push2(int x)
    {
        if(top1 < (top2-1))
        {
            ~~top2--;~~
            top2--;
            arr[top2] = x;
        }
```

```
else
 {

        cout <<" Stack 2 overflow ";
        exit(0);
    }
}

int pop1()
{
        if(top1 >=0)
        {
            int x = arr[top1];
            top1--; // decrement
            return(x);
        }
        else
        {
            cout <<" Stack 1 underflow ";
            exit(0);
        }
}

int pop2()
{
        if(top2 <= size-1)
        {
            int x = arr[top2];
            top2++; //increment
            return(x);
        }
        else
        {
```

```
            cout << "Stack2 underflow";
            exit(0);
        }
    }
};

int main()
{
    twoStacks ts(5);
    ts.push1(5);
    ts.push2(10);
    ts.push1(15);
    ts.push2(11);
    ts.push2(7);
    cout << "Popped element from Stack1 :" << ts.pop1();
    cout << "Popped element from Stack2 is:" << ts.pop2();
    return(0);
}
```

→ Output:

```
Popped element from Stack1 : 15
Popped element from Stack2 is : 7
```

→ Complexities:

- Time : O(1) - for both push and pop
- Space : O(n) - where n is array size.