```
//Created By Ritwik Chandra Pandey on 03/03/21
//183215
//Maintaining multiple stacks in a single array
```

//Time complexities of operations ***push()*** and ***pop()*** is ***O(1)***

//Time complexity of function ***createstack()*** is ***O(n)***

```
#include <stdio.h>
#include <stdlib.h>
//min represents the lower bound for a stack
//max represents the upper bound for a stack
int s[50],top[50],min[50],max[50];
//ns is stack number
//size is size of the stack
int ns,size;
//function to initialize starting values of top,min,max and stack
void init(void)
{
    int i;
    for(i=0;i<50;++i)
    {
        s[i]=min[i]=max[i] = 0;
        top[i]=-1;
    }//end for
}//end function
//function to create the stack
void createstack()
{
    int i ;
    //min and top of 0th stack will be -1
    //and it's max will be at index one lesser than it's size
    min[0]= -1;
```

```
    max[0] = size -1;
    top[0]=-1;

    //min and top of 1,2,3,….th stacks
    for(i=1;i<ns;++i)
    {
       min[i]= min[i-1] + size;
       top[i] = min [i];
    }//end for

    //max of 1,2,3,….th stacks will me min of 2,3,4,….th stack
    for(i=1;i<ns;++i)
    {
       max[i]= min[i+1];

    }//end for

}//end function
//function to push element to stack
//parameters passed will be the item user wants to push and the stack no. to //push
void push(int ele,int k)
{
    //check for stack overflow
    if(top[k-1]==max[k-1])
    {
       printf("Stack no %d is full i.e overflow\n",k);
       return;
    }//end if

    ++top[k-1];
    s[top[k-1]] = ele;

}//end function push
//function to pop an element form stack
//parameters passed is the stack no. from which we want to pop an element
void pop(int k)
{
    //check for underflow
    if(top[k-1]==min[k-1])
```

```
    {
        printf("\nStack no %d is empty i.e underflow\n",k);
        return;
    }//end if

    //else delete the item
    printf("%d from stack %d has been deleted.\n",s[top[k-1]],k);
    --top[k-1];

}//end function pop



//function to display any stack
//parameter passed is the stack number to display
void display(int k)
{
    //first check for stack empty condition
    //variable j is used to iterate through the list
    int j;
    if(top[k-1]==min[k-1])
    {
        printf("\nStack no %d is empty\n",k);
        return;
    }//end if

    //else display the list
    printf("\nStack %d →> ",k);

    for(j=min[k-1]+1;j<=top[k-1];++j)
    {
        printf("%d ",s[j] );
    }//end for

} //end function display
//main function
int main() {
    //variable choice,stack number and item to push is initialized
    int ele,ch,skn;
```

```
    init();//function call
here:
    //input the number of stacks
    printf("\nEnter the number of Stacks (Max 50)\n");
    scanf("%d",&ns);
    if(ns>50){
        printf("Not allowed\n");
        goto here;
    }

    //size of each stack
    //size = size of array/number of stacks
    size = 50/ns;

    createstack();//function call

    printf("\n1.Push\n2.Pop\n3.Display\n4.Exit\n");

    do{
        //ask for users choice
        printf("\nEnter your choice : \t");
        scanf("%d",&ch);

        switch(ch)
        {
            case 1: printf("\nEnter the stack no : \t");
                scanf("%d",&skn);
                printf("\nEnter the element : \t");
                scanf("%d",&ele);
                push(ele,skn);
                break;


            case 2 : printf("\nEnter the stack no to pop : \t");
                scanf("%d",&skn);
                pop(skn);
                break;
```

```
        case 3: printf("\nEnter the stack no to display : \t");
            scanf("%d",&skn);
            display(skn);
            break;


        case 4 : printf("\nProgram Terminating");
            break;


        default : printf("\nInvalid Option\n");
      }//end switch

   }//end do-while loop
   while(ch!=4);

   return 0;

}//end main
```