

// Stack Program A

```
#include<stdio.h>
#include<stdlib.h>
#define Size 4

int Top=-1, inp_array[Size];
void Push();
void Pop();
void show();

int main()
{
    int choice;
    while(1)
    {
        printf("\nOperations performed by Stack");
        printf("\n1.Push the element\n2.Pop the element\n3.Show\n4.End");
        printf("\n\nEnter the choice:");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: Push();
                    break;
            case 2: Pop();
                    break;
            case 3: show();
                    break;
            case 4: exit(0);

            default: printf("\nInvalid choice!!");
        }
    }
}

void Push()
{
    int x;

    if(Top==Size-1)
    {
        printf("\nOverflow!!");
    }
    else
    {
```

```

        printf("\nEnter element to be inserted to the stack:");
        scanf("%d",&x);
        Top=Top+1;
        inp_array[Top]=x;
    }
}

```

```

void Pop()
{
    if(Top== -1)
    {
        printf("\nUnderflow!!");
    }
    else
    {
        printf("\nPopped element: %d",inp_array[Top]);
        Top=Top-1;
    }
}

```

```

void show()
{
    if(Top== -1)
    {
        printf("\nUnderflow!!");
    }
    else
    {
        printf("\nElements present in the stack: \n");
        for(int i=Top;i>=0;--i)
            printf("%d\n",inp_array[i]);
    }
}

```

// Stack Program B

```

#include <stdio.h>
#include <stdlib.h>
#define MAX 10

```

```

int count = 0;

```

```

// Creating a stack

```

```

struct stack {
    int items[MAX];
    int top;
};
typedef struct stack st;

void createEmptyStack(st *s) {
    s->top = -1;
}

// Check if the stack is full
int isfull(st *s) {
    if (s->top == MAX - 1)
        return 1;
    else
        return 0;
}

// Check if the stack is empty
int isempty(st *s) {
    if (s->top == -1)
        return 1;
    else
        return 0;
}

// Add elements into stack
void push(st *s, int newitem) {
    if (isfull(s)) {
        printf("STACK FULL");
    } else {
        s->top++;
        s->items[s->top] = newitem;
    }
    count++;
}

// Remove element from stack
void pop(st *s) {
    if (isempty(s)) {
        printf("\n STACK EMPTY \n");
    } else {
        printf("Item popped= %d", s->items[s->top]);
        s->top--;
    }
}

```

```

    }
    count--;
    printf("\n");
}

// Print elements of stack
void printStack(st *s) {
    printf("Stack: ");
    for (int i = 0; i < count; i++) {
        printf("%d ", s->items[i]);
    }
    printf("\n");
}

// Driver code
int main() {
    int ch;
    st *s = (st *)malloc(sizeof(st));

    createEmptyStack(s);

    push(s, 1);
    push(s, 2);
    push(s, 3);
    push(s, 4);

    printStack(s);

    pop(s);

    printf("\nAfter popping out\n");
    printStack(s);
}

#include<stdio.h>
#include<stdlib.h>

struct stack{
    int size ;
    int top;
    int * arr;
};

```

```
int isEmpty(struct stack* ptr){
    if(ptr->top == -1){
        return 1;
    }
    else{
        return 0;
    }
}
```

```
int isFull(struct stack* ptr){
    if(ptr->top == ptr->size - 1){
        return 1;
    }
    else{
        return 0;
    }
}
```

```
void push(struct stack* ptr, int val){
    if(isFull(ptr)){
        printf("Stack Overflow! Cannot push %d to the stack\n", val);
    }
    else{
        ptr->top++;
        ptr->arr[ptr->top] = val;
    }
}
```

```
int pop(struct stack* ptr){
    if(isEmpty(ptr)){
        printf("Stack Underflow! Cannot pop from the stack\n");
        return -1;
    }
    else{
        int val = ptr->arr[ptr->top];
        ptr->top--;
        return val;
    }
}
```

```
int main(){
    struct stack *sp = (struct stack *) malloc(sizeof(struct stack));
    sp->size = 6;
    sp->top = -1;
```

[illegible]