

```
1: #include<stdio.h>
2: #include<stdlib.h>
3:
4: struct stack{
5:     int size ;
6:     int top;
7:     int * arr;
8: };
9:
10: int isEmpty(struct stack* ptr){
11:     if(ptr->top == -1){
12:         return 1;
13:     }
14:     else{
15:         return 0;
16:     }
17: }
18:
19: int isFull(struct stack* ptr){
20:     if(ptr->top == ptr->size - 1){
21:         return 1;
22:     }
23:     else{
24:         return 0;
25:     }
26: }
27:
28: void push(struct stack* ptr, int val){
29:     if(isFull(ptr)){
30:         printf("Stack Overflow! Cannot push %d to the stack\n", val);
31:     }
32:     else{
33:         ptr->top++;
34:         ptr->arr[ptr->top] = val;
35:     }
36: }
37:
38: int pop(struct stack* ptr){
39:     if(isEmpty(ptr)){
```

```

40:         printf("Stack Underflow! Cannot pop from the stack\n");
41:         return -1;
42:     }
43:     else{
44:         int val = ptr->arr[ptr->top];
45:         ptr->top--;
46:         return val;
47:     }
48: }
49:
50: int peek(struct stack* sp, int i){
51:     int arrayInd = sp->top - i + 1;
52:     if(arrayInd < 0){
53:         printf("Not a valid position for the stack\n");
54:         return -1;
55:     }
56:     else{
57:         return sp->arr[arrayInd];
58:     }
59: }
60:
61: int stackTop(struct stack* sp){
62:     return sp->arr[sp->top];
63: }
64:
65: int stackBottom(struct stack* sp){
66:     return sp->arr[0];
67: }
68:
69: int main(){
70:     // struct stack s;
71:     // s.size = 80;
72:     // s.top = -1;
73:     // s.arr = (int *) malloc(s.size * sizeof(int));
74:
75:     struct stack *sp = (struct stack *) malloc(sizeof(struct stack));
76:     sp->size = 10;
77:     sp->top = -1;
78:     sp->arr = (int *) malloc(sp->size * sizeof(int));

```

```

79:     printf("Stack has been created successfully\n");
80:
81:     push(sp, 1);
82:     push(sp, 23);
83:     push(sp, 99);
84:     push(sp, 75);
85:     push(sp, 3);
86:     push(sp, 64);
87:     push(sp, 57);
88:     push(sp, 46);
89:     push(sp, 89);
90:     push(sp, 6); // ---> Pushed 10 values
91:     // push(sp, 46); // Stack Overflow since the size of the st
92:     printf("After pushing, Full: %d\n", isFull(sp));
93:     printf("After pushing, Empty: %d\n", isEmpty(sp));
94:
95:     printf("Popped %d from the stack\n", pop(sp)); // --> Last in
96:     printf("Popped %d from the stack\n", pop(sp)); // --> Last in
97:     printf("Popped %d from the stack\n", pop(sp)); // --> Last in
98:
99:     // Printing values from the stack
100:    for (int j = 1; j <= sp->top + 1; j++)
101:    {
102:        printf("The value at position %d is %d\n", j, peek(sp, j));
103:    }
104:
105:    printf("The top most value of this stack is %d\n", stackTop(sp));
106:    printf("The bottom most value of this stack is %d\n", stackBot
107:
108:    return 0;
109: }
110:

```