

```

1  #include<stdio.h>
2  #include<stdlib.h>
3
4  struct stackArray
5  {
6      int top1;
7      int top2;
8      int capacity;
9      int *stack1;
10 };
11
12 struct stackArray * createStack(int capacity)
13 {
14     struct stackArray *S=(struct stackArray *)malloc(sizeof(struct stackArray));
15     S->top1=-1;
16     S->top2=capacity;
17     S->capacity=capacity;
18     S->stack1=(int *)malloc(sizeof(int)*capacity);
19     return S;
20 }
21
22 void pushStack1(struct stackArray *s,int value)
23 {
24     if(s->top1==s->top2-1)
25     {
26         printf("stack is full");
27         return;
28     }
29     else
30     {
31         s->top1++;
32         s->stack1[s->top1]=value;
33         return;
34     }
35 }
36
37
38 void pushStack2(struct stackArray *s,int value)
39 {
40     if(s->top2==s->top1+1)
41     {
42         printf("stack is full");
43         return;
44     }
45     else
46     {
47         s->top2--;
48         s->stack1[s->top2]=value;
49         return;
50     }
51 }
52
53
54 int pop1(struct stackArray *s)
55 {
56     if(s->top1==--1)
57     {
58         printf("stack Underflow");
59         return NULL;
60     }
61     else
62     {
63         return s->stack1[s->top1];
64         s->top1--;
65     }
66 }

```

```

67  }
68
69  int pop2(struct stackArray *s)
70  {
71      if(s->top2==s->capacity)
72      {
73          printf("stack Underflow");
74          return NULL;
75      }
76      else
77      {
78          return s->stack1[s->top2];
79          s->top1++;
80      }
81  }
82  }
83
84
85
86  int main()
87  {
88      struct stackArray *s=createStack(5);
89      pushStack1(s,5);
90      pushStack1(s,4);
91      pushStack1(s,3);
92      pushStack2(s,2);
93      pushStack2(s,1);
94      printf("element popped from 1st stack is %d",pop1(s));
95      printf("\nelement popped from 2nd stack is %d",pop2(s));
96
97  }

```