

```

1  #include<stdio.h>
2  #include<stdlib.h>
3
4  struct stack
5  {
6      int top;
7      int capacity;
8      int *array;
9  };
10
11
12  struct advancedStack
13  {
14      struct stack *elementstack;
15      struct stack *minstack;
16  };
17
18  struct stack *createStack(int c)
19  {
20      struct stack *S=malloc(sizeof( struct stack));
21      S->top=-1;
22      S->capacity=c;
23      S->array=malloc(S->capacity*sizeof(int));
24
25      return S;
26  }
27
28  struct advancedStack *createAdvancedStack()
29  {
30      struct advancedStack *As=(struct advancedStack *)malloc(sizeof(struct
advancedStack));
31      As->elementstack=createStack(5);
32      As->minstack=createStack(5);
33      return As;
34  }
35
36  int isEmpty(struct stack *S)
37  {
38      return(S->top== -1);
39  }
40
41  int isFull(struct stack *S)
42  {
43      return(S->top==S->capacity-1);
44  }
45
46
47  void push(struct stack *S,int k)
48  {
49      if(isFull(S))
50      {
51          printf("OverFlow");
52          return;
53      }
54
55
56      else
57
58
59
60
61          S->array[++S->top]=k;
62
63
64  }
65

```

```

66 int pop(struct stack *S)
67 {
68     int data;
69     if(isEmpty(S))
70     {
71         printf("UnderFlow");
72     }
73
74
75     else
76     {
77         data=S->array[S->top--];
78         return data;
79     }
80
81 }
82
83 void pushA(struct advancedStack *S,int data)
84 {
85     if(isFull(S->elementstack))
86     {
87         printf("overflow, cant insert any more values");
88     }
89
90     else
91     {
92         push(S->elementstack,data);
93         if(isEmpty(S->minstack) || data<=S->minstack->array[S->minstack->top])
94             push(S->minstack,data);
95     }
96
97
98
99 }
100
101 void popA(struct advancedStack *S)
102 {
103     int data;
104     if(isEmpty(S->elementstack))
105         printf("underflow, cant delete any more values");
106     else
107     {
108         data=pop(S->elementstack);
109     }
110
111     if(S->minstack->array[S->minstack->top]==data)
112         pop(S->minstack);
113 }
114
115 int getMinimum(struct advancedStack *S)
116 {
117     return S->minstack->array[S->minstack->top];
118 }
119
120 void main()
121 {
122     struct advancedStack *S=createAdvancedStack();
123     pushA(S,5);
124     pushA(S,6);
125     pushA(S,4);
126     pushA(S,3);
127     pushA(S,1);
128     pushA(S,0);
129     // popA(S);
130     //popA(S);
131     printf("Minimum is: %d",getMinimum(S));

```

132 }
133