

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  typedef struct
4  {
5      int value;
6      int min;
7  } StackNode;
8
9
10 typedef struct
11 {
12     StackNode *array;
13     int capacity;
14     int top;
15 }MinStack;
16
17 MinStack* minStackCreate()
18 {
19     MinStack* stack=(MinStack*)malloc(sizeof(MinStack));
20     stack->capacity = 10;
21     stack->array = (StackNode*)malloc(stack->capacity * sizeof(StackNode));
22     stack->top = -1;
23     return stack;
24 }
25
26
```

```
27 void minStackPush(MinStack* obj, int val)
28 {
29     if (obj->top == obj->capacity - 1)
30     {
31         obj->capacity *= 2;
32         obj->array = (StackNode*)realloc(obj->array, obj->capacity * sizeof(StackNode));
33     }
34     StackNode newNode;
35     newNode.value = val;
36     newNode.min = (obj->top == -1) ? val : (val < obj->array[obj->top].min) ? val : obj->array[obj->top].min;
37     obj->array[++(obj->top)] = newNode;
38 }
39
40 void minStackPop(MinStack* obj)
41 {
42     if (obj->top != -1)
43     {
44         obj->top--;
45     }
46 }
47
48
49 int minStackTop(MinStack* obj)
50 {
51     if (obj->top != -1)
52     {
53         return obj->array[obj->top].value;
54     }
55 }
```

```
55     return -1;
56
57 }
58
59 int minStackGetMin(MinStack* obj)
60 {
61     if (obj->top != -1)
62     {
63         return obj->array[obj->top].min;
64     }
65     return -1;
66
67 }
68
69 void minStackFree(MinStack* obj)
70 {
71     free(obj->array);
72     free(obj);
73 }
```