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

```
void minStackPush(MinStack* obj, int val)
27
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;
        newNode.min = (obj->top == -1) ? val : (val < obj->array[obj->top].min) ? val : obj->array[obj->top].min;
36
37
        obj->array[++(obj->top)] = newNode;
38
39
40
    void minStackPop(MinStack* obj)
41
         if (obj->top != -1)
42
43
44
            obj->top--;
45
46
47
48
    int minStackTop(MinStack* obj)
49
50
51
        if (obj->top != -1)
52
        {
53
            return obj->array[obj->top].value;
54
```

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