```
c/> Code
C V & Auto
       typedef struct {
    4
    5
            int size;
    6
            int top;
    7
            int *s:
           int *minstack;
    8
    9
   10
        } MinStack;
   11
   12
       MinStack* minStackCreate() {
   13
           MinStack *st=(MinStack*) malloc(sizeof(MinStack));
   14
            if(st==NULL)
   15
   16
   17
                printf("memory alloction failed");
                exit(0);
   18
   19
   20
            st->size=5;
           st->top=-1;
   21
            st->s=(int*) malloc (st->size*sizeof(int));
   22
   23
            st->minstack = (int*) malloc (st->size * sizeof(int));
            if(st->s-NULL)
   24
   25
                printf("memory allocation failed");
   26
                free(st->s);
   27
   28
                free(st->minstack);
                exit(0);
   29
   30
   31
            return st;
   32
   33
       3
   34
        void minStackPush(MinStack* obj, int val) {
   35
   36
            if(obj->top==obj->size-1)
   37
                printf("stack is overflow");
   38
   39
   40
           else{
   41
   42
                obj->top++;
                obj->s[obj->top]=val;
   43
```

Saved to local

```
Auto
      else{
          obj->top++;
          obj->s[obj->top]=val;
          if (obj->top == 0 || val < obj->minstack[obj->top - 1]) {
               obj->minstack[obj->top] = val;
           } else {
               obj->minstack[obj->top] = obj->minstack[obj->top - 1];
2
3
   void minStackPop(MinStack* obj) {
4
       int value;
5
       if(obj->top==-1)
6
7/
           printf("underflow");
8
9
50
        1
        else
3
52
            value=obj->s[obj->top];
63
            obj->top--;
64
            printf("%d is popped\n", value);
65
66
67
68
69
    int minStackTop(MinStack* obj) {
70
        int value=-1;
71
        if(obj->top==-1)
```

72

73

74

75 76 77

78

79

{

else

1

printf("underflow\n");

exit(0);

```
Auto
 18
         else
 79
         {
             value=obj->s[obj->top];
 80
             return value;
 81
 82
 83
 84
 85
     }
 86
     int minStackGetMin(MinStack* obj) {
 87
         if(obj->top==-1)
 88
 89
         1
              printf("underflow\n");
 90
             exit(0);
 91
 92
 93
         else
 94
 95
              return obj->minstack[obj->top];
 96
 97
 98
 99
     3
100
101
     void minStackFree(MinStack* obj) {
         free(obj->s);
102
         free(obj->minstack);
103
         free(obj);
104
105
106
107
108
109
      * Your MinStack struct will be instantiated and called as such:
110
      * MinStack* obj = minStackCreate();
111
      * minStackPush(obj, val);
112
113
114
      * minStackPop(obj);
115
      * int param 3 = minStackTop(obj);
116
117
      * int param 4 = minStackGetMin(obj);
118
```

```
Testcase | >_ Test Result
Input
 ["MinStack", "push", "push", "getMin", "pop", "top", "getMin"]
 [[],[-2],[0],[-3],[],[],[],[],[]]
Stdout
                                                                                                        Ф
 -3 is popped
Output
  [null,null,null,-3,null,0,-2]
Expected
  [null,null,null,-3,null,0,-2]
                                              Contribute a testcase
```