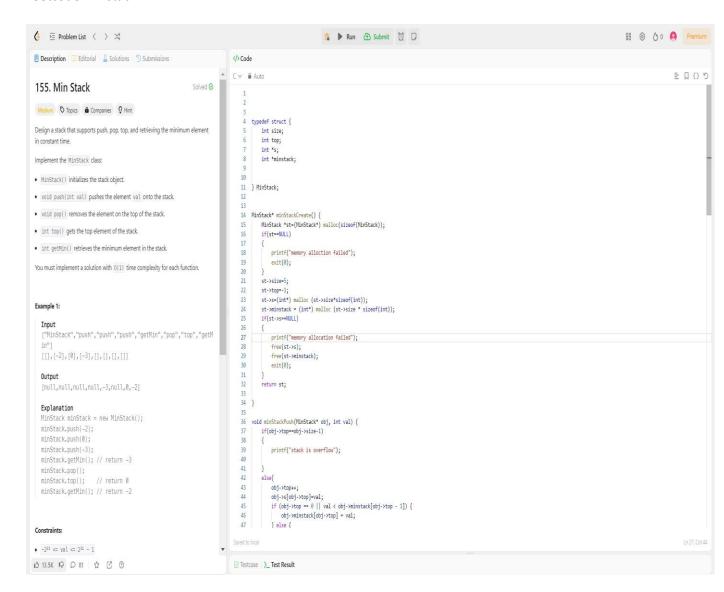
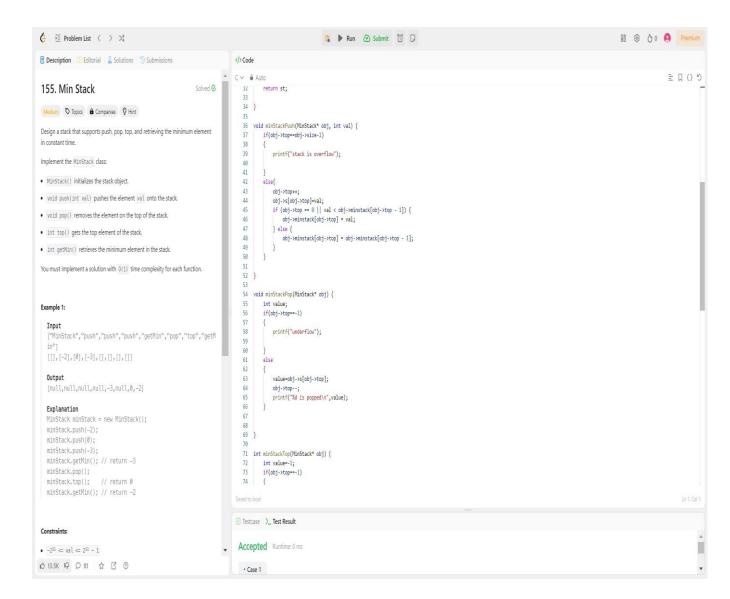
## Leetcode minstack





```
👉 🗏 Problem List ⟨ ⟩ 💢
                                                                                                                         88 @ O O Prem
⟨/> Code
                                                                                  C ∨ ≜ Auto
                                                                                                                                                                                                                                                                           2 () □ =
                                                                  Solved @
 155. Min Stack
                                                                                      69 }
70
                                                                                      71 int minStackTop(MinStack* obj) {
72 | int value*-1*
 Medium ♥ Topics ♠ Companies ♀ Hint
                                                                                             int value=-1;
if(obj->top==-1)
{
Design a stack that supports push, pop, top, and retrieving the minimum element
                                                                                                  printf("underflow\n");
exit(0);
Implement the MinStack class:

    MinStack() initializes the stack object.

    void push(int val) pushes the element val onto the stack.

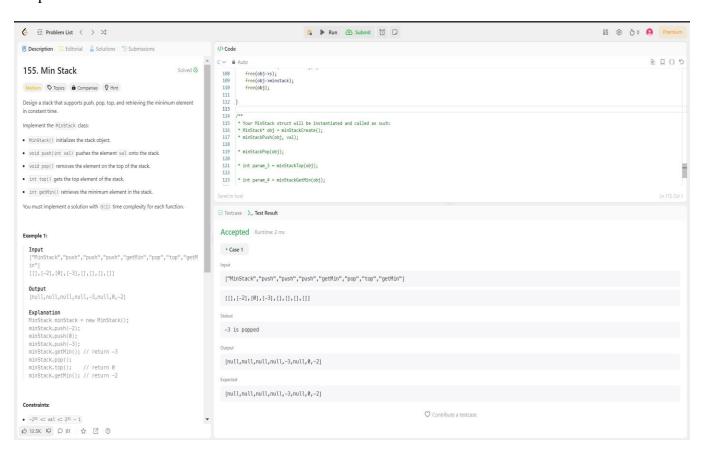
                                                                                                  value=obj->s[obj->top];

    void_pop() removes the element on the top of the stack.

    int top() gets the top element of the stack.

• int getMin() retrieves the minimum element in the stack.
 You must implement a solution with O(1) time complexity for each function.
                                                                                       90 int minStackGetMin(MinStack* obj) {
Example 1:
                                                                                               if(obj->top==-1)
    ["MinStack","push","push","push","getMin","pop","top","getM
                                                                                                  printf("underflow\n");
                                                                                     96
97
98
99
100
101
102
                                                                                                   exit(0):
   [[],[-2],[0],[-3],[],[],[],[],[]]
   Output
   [null,null,null,-3,null,0,-2]
                                                                                                  return obj->minstack[obj->top];
   Explanation
                                                                                     102
103
104
105 }
106
107 voi
108
109
110
   MinStack minStack = new MinStack();
minStack.push(-2);
   minStack.push(0);
                                                                                          void minStackFree(MinStack* obj) {
  free(obj->s);
  free(obj->minstack);
  free(obj);
   minStack.push(-3);
minStack.getMin(); // return -3
   minStack.pop();
minStack.top(); // return 0
minStack.getMin(); // return -2
                                                                                     111
112
                                                                                     112 }
113 |
114 /**
115 | * Your MinStack struct will be instantiated and called as such:
Constraints:
• -2^{31} \leftarrow \text{val} \leftarrow 2^{31} - 1
ið 13.5K I⊋ □ 81 ☆ ☑ ③
                                                                                 ☑ Testcase >_ Test Result
```

## Output



```
Code
```

```
typedef struct {
    int size;
    int top;
    int *s;
    int *minstack;
} MinStack;
MinStack* minStackCreate() {
    MinStack *st=(MinStack*) malloc(sizeof(MinStack));
    if(st==NULL)
    {
        printf("memory alloction failed");
        exit(0);
    }
    st->size=5;
    st->top=-1;
    st->s=(int*) malloc (st->size*sizeof(int));
    st->minstack = (int*) malloc (st->size * sizeof(int));
    if(st->s==NULL)
    {
        printf("memory allocation failed");
        free(st->s);
        free(st->minstack);
        exit(0);
    }
    return st;
}
void minStackPush(MinStack* obj, int val) {
    if(obj->top==obj->size-1)
    {
        printf("stack is overflow");
    }
    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];
        }
    }
}
```

```
void minStackPop(MinStack* obj) {
    int value;
    if(obj->top==-1)
        printf("underflow");
    }
    else
    {
        value=obj->s[obj->top];
        obj->top--;
        printf("%d is popped\n", value);
    }
}
int minStackTop(MinStack* obj) {
    int value=-1;
    if(obj->top==-1)
        printf("underflow\n");
        exit(0);
    }
    else
    {
        value=obj->s[obj->top];
        return value;
    }
}
int minStackGetMin(MinStack* obj) {
    if(obj->top==-1)
    {
        printf("underflow\n");
        exit(0);
    }
    else
    {
        return obj->minstack[obj->top];
    }
}
void minStackFree(MinStack* obj) {
    free(obj->s);
    free(obj->minstack);
```

```
free(obj);

/**

* Your MinStack struct will be instantiated and called as such:

* MinStack* obj = minStackCreate();

* minStackPush(obj, val);

* minStackPop(obj);

* int param_3 = minStackTop(obj);

* int param_4 = minStackGetMin(obj);

* minStackFree(obj);

*//
```