

ADA LAB-1

(Leet Code Questions)

Name : Kathasagaram Aishwarya

USN : 1BM22CS123

Section : 4C (B1)

Q) Implement a last-in-first-out (LIFO) stack using only two queues. The implemented stack should support all the functions of a normal stack (push, top, pop, and empty).

Link: <https://leetcode.com/problems/implement-stack-using-queues/description/>

CODE:

```
typedef struct {
    int q1[101];
    int q2[101];
    int f1, r1, f2, r2;
} MyStack;

MyStack* myStackCreate() {
    MyStack *nn = (MyStack *)malloc(sizeof(MyStack));
    nn->f1=nn->r1=nn->f2=nn->r2=-1;
    return nn;
}

void myStackPush(MyStack* obj, int x) {
    if(obj->f1 == -1) {
        obj->f1 = 0;
    }

    obj->q1[++obj->r1] = x;
}
```

```
int myStackPop(MyStack* obj) {
    int k;
    if(obj->f1!=-1) {
        for(int i=obj->f1;i<obj->r1;i++) {
            if(obj->f2==-1) {
                obj->f2++;
            }
            obj->q2[++obj->r2] = obj->q1[i];
        }
        k = obj->q1[obj->r1];
        obj->f1=obj->r1=-1;
    }
    else{
        for(int i=obj->f2;i<obj->r2;i++) {
            if(obj->f1==-1) {
                obj->f1++;
            }
            obj->q1[++obj->r1] = obj->q2[i];
        }
        k = obj->q2[obj->r2];
        obj->f2=obj->r2=-1;
    }

    return k;
}
```

```
int myStackTop(MyStack* obj) {
    if(obj->f1 != -1) {
        return obj->q1[obj->r1];
    }
    else if(obj->f2 != -1) {
        return obj->q2[obj->r2];
    }
}
```

```
    }  
    else{  
        return -1;  
    }  
}
```

```
bool myStackEmpty(MyStack* obj) {  
    return (obj->f1==-1 && obj->f2==-1);  
}
```

```
void myStackFree(MyStack* obj) {  
    free(obj);  
}
```

```
/**
```

```
* Your MyStack struct will be instantiated and called as such:
```

```
* MyStack* obj = myStackCreate();
```

```
* myStackPush(obj, x);
```

```
* int param_2 = myStackPop(obj);
```

```
* int param_3 = myStackTop(obj);
```

```
* bool param_4 = myStackEmpty(obj);
```

```
* myStackFree(obj);
```

```
*/
```

OUTPUT:

The screenshot displays a code submission interface with a dark theme. At the top, there's a navigation bar with icons for back, forward, and search, followed by a 'Problem List' button. To the right are icons for a bug report, a 'Run' button, and a share icon. Below this is a tabbed interface with 'Description', 'Editorial', 'Solutions', and 'Submissions' tabs. The 'Submissions' tab is active, showing a list of submissions. The first submission is by 'Aish_kathasagaram', submitted on May 06, 2024, at 19:08. It is marked as 'Accepted'. To the right of the submission name are buttons for 'Editorial' and 'Solution'. Below the submission details, there are two performance metrics: 'Runtime' and 'Memory'. The 'Runtime' is 0 ms, and the 'Memory' is 5.65 MB. Both metrics include a green leaf icon and the text 'Beats 100.00% of users with C' for runtime and 'Beats 69.48% of users with C' for memory. Below these metrics is a bar chart showing the distribution of runtime and memory usage. The chart has two bars: a white bar for runtime and a grey bar for memory. The runtime bar is at 0 ms, and the memory bar is at 5.65 MB. The chart also shows a distribution of other submissions, with some bars reaching up to 40% on the y-axis. At the bottom of the chart, there are labels for '1ms' and '2ms'. Below the chart, there's a 'Code' button and a 'C' language indicator. The bottom section of the interface is titled 'Testcase' and 'Test Result'. It shows 'Case 1' with a plus icon. The test case input is a string array: `["MyStack", "push", "push", "top", "pop", "empty"]`. The test case output is an array of arrays: `[[], [1], [2], [], [], []]`. At the bottom left, there's a 'Source' button with a code icon.

Problem List < > 🔍

Description | Editorial | Solutions | Submissions

← All Submissions 🔗

Accepted

Aish_kathasagaram submitted at May 06, 2024 19:08

Editorial Solution

⌚ Runtime

0 ms

🌿 Beats 100.00% of users with C

⚙️ Memory

5.65 MB

🌿 Beats 69.48% of users with C

40%

20%

0%

1ms

2ms

Code | C

☑️ Testcase | >_ Test Result

Case 1 +

`["MyStack", "push", "push", "top", "pop", "empty"]`

`[[], [1], [2], [], [], []]`

</> Source