

```
1  #include <stdbool.h>
2  #include <stdlib.h>
3
4  typedef struct {
5      int* data;
6      int front;
7      int rear;
8      int size;
9  } Queue;
10
11 typedef struct {
12     Queue* q1;
13     Queue* q2;
14 } MyStack;
15
16 Queue* createQueue(int size) {
17     Queue* queue = (Queue*)malloc(sizeof(Queue));
18     queue->data = (int*)malloc(size * sizeof(int));
19     queue->front = queue->rear = -1;
20     queue->size = size;
21     return queue;
22 }
23
24 void enqueue(Queue* queue, int value) {
25     if (queue->rear == -1) {
26         queue->front = queue->rear = 0;
27     } else {
28         queue->rear = (queue->rear + 1) % queue->size;
29     }
30     queue->data[queue->rear] = value;
31 }
32
33 int dequeue(Queue* queue) {
34     int value = queue->data[queue->front];
35     if (queue->front == queue->rear) {
36         queue->front = queue->rear = -1;
37     } else {
38         queue->front = (queue->front + 1) % queue->size;
```

```
38     queue->front = (queue->front + 1) % queue->size;
39 }
40 return value;
41 }
42
43 bool isEmpty(Queue* queue) {
44     return queue->front == -1;
45 }
46
47 MyStack* myStackCreate() {
48     MyStack* stack = (MyStack*)malloc(sizeof(MyStack));
49     stack->q1 = createQueue(1000); // Adjust the size as needed
50     stack->q2 = createQueue(1000);
51     return stack;
52 }
53
54 void myStackPush(MyStack* obj, int x) {
55     enqueue(obj->q1, x);
56 }
57
58 int myStackPop(MyStack* obj) {
59     if (isEmpty(obj->q1)) {
60         return -1; // Stack is empty
61     }
62
63     while (obj->q1->front != obj->q1->rear) {
64         enqueue(obj->q2, dequeue(obj->q1));
65     }
66
67     int poppedValue = dequeue(obj->q1);
68
69     // Swap q1 and q2
70     Queue* temp = obj->q1;
71     obj->q1 = obj->q2;
72     obj->q2 = temp;
73
74     return poppedValue;
75 }
```

```
74     return poppedValue;
75 }
76
77 int myStackTop(MyStack* obj) {
78     if (isEmpty(obj->q1)) {
79         return -1; // Stack is empty
80     }
81
82     while (obj->q1->front != obj->q1->rear) {
83         enqueue(obj->q2, dequeue(obj->q1));
84     }
85
86     int topValue = dequeue(obj->q1);
87     enqueue(obj->q2, topValue);
88
89     // Swap q1 and q2
90     Queue* temp = obj->q1;
91     obj->q1 = obj->q2;
92     obj->q2 = temp;
93
94     return topValue;
95 }
96
97 bool myStackEmpty(MyStack* obj) {
98     return isEmpty(obj->q1);
99 }
100
101 void myStackFree(MyStack* obj) {
102     free(obj->q1->data);
103     free(obj->q1);
104     free(obj->q2->data);
105     free(obj->q2);
106     free(obj);
107 }
```

Accepted Runtime: 3 ms

• Case 1

Input

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

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

Output

```
[null,null,null,2,2,false]
```

Expected

```
[null,null,null,2,2,false]
```