1.Implementing stacks using queues

Shashank Patel C J

1BM22CS255

```c
typedef struct {
    int* queue1;
    int* queue2;
    int f1, f2, r1, r2;

} MyStack;

MyStack* myStackCreate() {
    MyStack* st = (MyStack*)malloc(sizeof(MyStack));
    st->queue1 = (int*)calloc(10, sizeof(int));
    st->queue2 = (int*)calloc(10, sizeof(int));
    st->f1 = -1;
    st->f2 = -1;
    st->r1 = -1;
    st->r2 = -1;
    return st;
}

void myStackPush(MyStack* obj, int x) {
    if (obj->f1 == -1 && obj->r1 == -1) {
        obj->f1 = 0;
        obj->r1 = 0;
    }
```

```
    else {
        obj->r1++;
    }
    printf("%d\n", x);
    obj->queue1[obj->r1] = x;
}

int myStackPop(MyStack* obj) {
    if (obj->f1 == -1) {
        return -1;
    }
    int k1 = obj->f1;
    int l1 = obj->r1;
    int k2 = obj->f2;
    int l2 = obj->r2;
    int ch;
    while (k1 < l1) {
        if (k2 == -1) {
            k2 = 0;
            l2 = 0;
        } else {
            l2++;
        }

        obj->queue2[l2] = obj->queue1[k1];
```

```
        k1++;
    }
    ch = obj->queue1[k1];
    k1=-1;
    l1=-1;
    int* temp = obj->queue1;
    obj->queue1 = obj->queue2;
    obj->queue2 = temp;

    obj->f1 = k2;
    obj->f2 = k1;

    obj->r1 = l2;
    obj->r2 = l1;
    if(obj->r1<obj->f1){
        obj->r1=-1;
        obj->f1=-1;
    }

    return ch;
}

int myStackTop(MyStack* obj) {
    if (obj->f1 == -1) {
        return -1;
    }
```

```
    int k1 = obj->f1;
    int l1 = obj->r1;
    int k2 = obj->f2;
    int l2 = obj->r2;
    int ch;
    while (k1 <= l1) {
        if (k2 == -1) {
            k2 = 0;
            l2 = 0;
        } else {
            l2++;
        }
        ch = obj->queue1[k1];
        obj->queue2[l2] = obj->queue1[k1];
        k1++;
    }

    int* temp = obj->queue1;
    obj->queue1 = obj->queue2;
    obj->queue2 = temp;

    return ch;
}
```

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

void myStackFree(MyStack* obj) {
    free(obj->queue1);
    free(obj->queue2);
    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);
*/
```

Accepted Runtime: 3 ms

• Case 1

Input

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

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

Expected

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

Stdout

```
1
2
```

Output

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

Case 1

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

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