```c

struct queueNode{
    int data;
};

typedef struct {
    struct queueNode *arr[101];
    int front;
    int rear;
} MyStack;


MyStack* myStackCreate() {
    MyStack *q =(MyStack *)malloc(sizeof(MyStack));
    q->front=-1;
    q->rear=-1;
    return q;
}

void enqueue(MyStack* q, struct queueNode* item) {
    q->arr[++q->rear] = item;
    if (q->front == -1) {
        q->front = 0;
    }
}

struct queueNode* dequeue(MyStack* q) {

    struct queueNode* item = q->arr[q->front];
    if (q->front == q->rear) {
        q->front = q->rear = -1;
    } else {
        q->front++;
    }
    return item;
}

void myStackPush(MyStack* q, int x) {

  struct queueNode* node = (struct queueNode*) malloc(sizeof(struct queueNode));
```

```c
    node->data = x;
    enqueue(q, node);
    int size = q->rear - q->front + 1;
    while (size > 1) {
        struct queueNode* front = dequeue(q);
        enqueue(q, front);
        size--;
    }
}

int myStackPop(MyStack* q) {

    struct queueNode* front = dequeue(q);
    int item = front->data;
    free(front);
    return item;
}

int myStackTop(MyStack* q) {

    struct queueNode* front = q->arr[q->front];
    return front->data;
}

bool myStackEmpty(MyStack* q) {
  return q->front == -1;
}

void myStackFree(MyStack* q) {
    while (!myStackEmpty(q)) {
        struct queueNode* front = dequeue(q);
        free(front);
    }
    free(q);
}
```

Accepted

Sarvesh Rastogi submitted at Feb 26, 2024 12:03

Editorial | Solution

Runtime

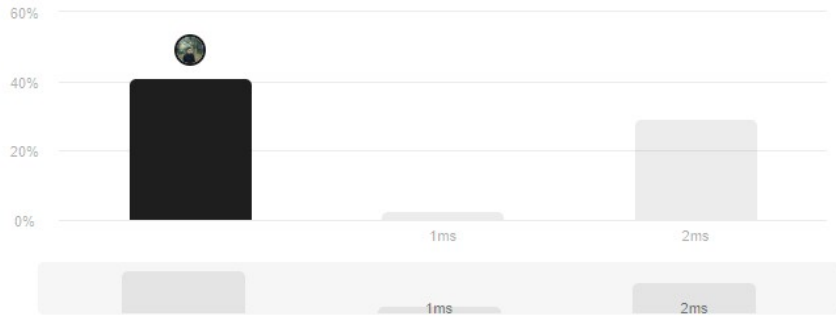
0 ms

Beats 100.00% of users with C

Memory

5.79 MB

Beats 71.69% of users with C



Code | C