

LC-1 LEETCODE:WAP to implement stack using queues.

CODE:

```
#include <stdbool.h>

#include <stdlib.h>

typedef struct {
    int* data;
    int front;
    int rear;
    int size;
} Queue;

typedef struct {
    Queue* q1;
    Queue* q2;
} MyStack;

Queue* createQueue(int size) {
    Queue* queue = (Queue*)malloc(sizeof(Queue));
    queue->data = (int*)malloc(size * sizeof(int));
    queue->front = queue->rear = -1;
    queue->size = size;
    return queue;
}

void enqueue(Queue* queue, int value) {
    if (queue->rear == -1) {
        queue->front = queue->rear = 0;
    } else {
        queue->rear = (queue->rear + 1) % queue->size;
```

```

}
queue->data[queue->rear] = value;
}

int dequeue(Queue* queue) {
int value = queue->data[queue->front];
if (queue->front == queue->rear) {
queue->front = queue->rear = -1;
} else {
queue->front = (queue->front + 1) % queue->size;
}
return value;
}

bool isEmpty(Queue* queue) {
return queue->front == -1;
}

MyStack* myStackCreate() {
MyStack* stack = (MyStack*)malloc(sizeof(MyStack));
stack->q1 = createQueue(1000); // Adjust the size as needed
stack->q2 = createQueue(1000);
return stack;
}

void myStackPush(MyStack* obj, int x) {
enqueue(obj->q1, x);
}

int myStackPop(MyStack* obj) {
if (isEmpty(obj->q1)) {

```

```

return -1; // Stack is empty
}
while (obj->q1->front != obj->q1->rear) {
    enqueue(obj->q2, dequeue(obj->q1));
}
int poppedValue = dequeue(obj->q1);
// Swap q1 and q2
Queue* temp = obj->q1;
obj->q1 = obj->q2;
obj->q2 = temp;
return poppedValue;
}

int myStackTop(MyStack* obj) {
    if (isEmpty(obj->q1)) {
        return -1; // Stack is empty
    }
    while (obj->q1->front != obj->q1->rear) {
        enqueue(obj->q2, dequeue(obj->q1));
    }
    int topValue = dequeue(obj->q1);
    enqueue(obj->q2, topValue);
    // Swap q1 and q2
    Queue* temp = obj->q1;
    obj->q1 = obj->q2;
    obj->q2 = temp;
    return topValue;
}

```

```

}

bool myStackEmpty(MyStack* obj) {
return isEmpty(obj->q1);
}

void myStackFree(MyStack* obj) {
free(obj->q1->data);
free(obj->q1);
free(obj->q2->data);
free(obj->q2);
free(obj);
}

```

OUTPUT:

The screenshot displays a code editor interface for a C++ problem. The solution is marked as "Accepted" and was submitted by "Shreevarnam" on Jan 08, 2024, at 23:14. The performance metrics show a runtime of 3 ms (beating 27.08% of users) and a memory usage of 6.94 MB (beating 8.92% of users). A bar chart compares the user's performance with other users. The code defines a `MyStack` struct with two queues, `q1` and `q2`, and implements `myStackEmpty` and `myStackFree` functions. A sidebar on the right shows the user's profile and navigation options.

Accepted
Shreevarnam submitted at Jan 08, 2024 23:14

Runtime
3 ms
Beats 27.08% of users with C

Memory
6.94 MB
Beats 8.92% of users with C

Code C

```

#include <stdbool.h>
#include <stdlib.h>

typedef struct {
    int* data;
    int front;
    int rear;
    int size;
} Queue;

typedef struct {
    Queue* q1;
    Queue* q2;
} MyStack;

Queue* createQueue(int size) {
    Queue* queue = (Queue*)malloc(sizeof(Queue));
    queue->data = (int*)malloc(size * sizeof(int));
    queue->front = queue->rear = -1;
    queue->size = size;
    return queue;
}

void enqueue(Queue* queue, int value) {
    if (queue->rear == -1) {

```

Shreevarnam
Access all features with our Premium subscription!

- My Lists
- Notebook
- Submissions
- Progress
- Points
- Session
- Try New Features
- Orders
- My Playgrounds
- Revert to old version
- Appearance
- Sign Out

Saved to local

Testcase Test Result

LC-2 LEETCODE:Maximum twin sum of a linked list. In a linked list of size n , where n is even the i th node (0-indexed) of the linked list is known as twin if the $(n-1-i)$ th node, if $0 \leq i \leq (n/2)-1$

CODE:

```
struct ListNode* reverse(struct ListNode* head)
{
    struct ListNode* p = NULL, *q = NULL, *r = head;
    while(r != NULL){
        p = q;
        q = r;
        r = r->next;
        q->next = p;
    }
    head = q;
    return head;
}

int pairSum(struct ListNode* head) {
    if(head == NULL){
        return -1;
    }
    //if only 2 nodes
    if(head->next->next == NULL){
        int sum = head->val + head->next->val;
        return sum;
    }
    struct ListNode *temp = head, *s = head, *f = head->next;
```

```
// find middle
while(f!=NULL){
f = f->next;
if(f!=NULL){
f = f->next;
s = s->next;
}
}
struct ListNode* second = reverse(s->next);
s->next = second;
struct ListNode* first = head;
int ans = INT_MIN;
while(second != NULL){
int data = first->val + second->val;
ans = fmax(ans,data);
first = first->next;
second = second->next;
}
return ans;
}
```

OUTPUT:

Accepted
Shreevarnam submitted at Jan 29, 2024 22:45

Runtime
132 ms
Beats 31.52% of users with C

Memory
46.68 MB
Beats 94.94% of users with C

Code

```

1  /**
2   * Definition for singly-linked list.
3   * struct ListNode {
4   *     int val;
5   *     struct ListNode *next;
6   * };
7   */
8   struct ListNode* reverse(struct ListNode* head) {
9   {
10      struct ListNode*p = NULL, *q = NULL,
11      while(r!= NULL){
12          p = q;
13          q = r;
14          r = r->next;
15          q->next = p;
16      }
17      head = q;
18      return head;
19  }
20  }
21
22  int pairSum(struct ListNode* head) {
23      if(head == NULL){
24          return -1;
25      }
26
27      //if only 2 nodes
28      if(head->next->next ==NULL){

```

Shreevarnam
Access all features with our Premium subscription!

My Lists Notebook Submissions Progress Points Session

Try New Features Orders My Playgrounds Revert to old version Appearance Sign Out

LC-3 LEETCODE:Merged two binary trees.

CODE:

```

struct TreeNode* mergeTrees(struct TreeNode* root1, struct TreeNode*
root2) {
if(root1 == NULL)
{
return root2;
}
if(root2 == NULL)
{
return root1;
}
root1->val += root2->val;
root1->left = mergeTrees(root1->left,root2->left);

```

```

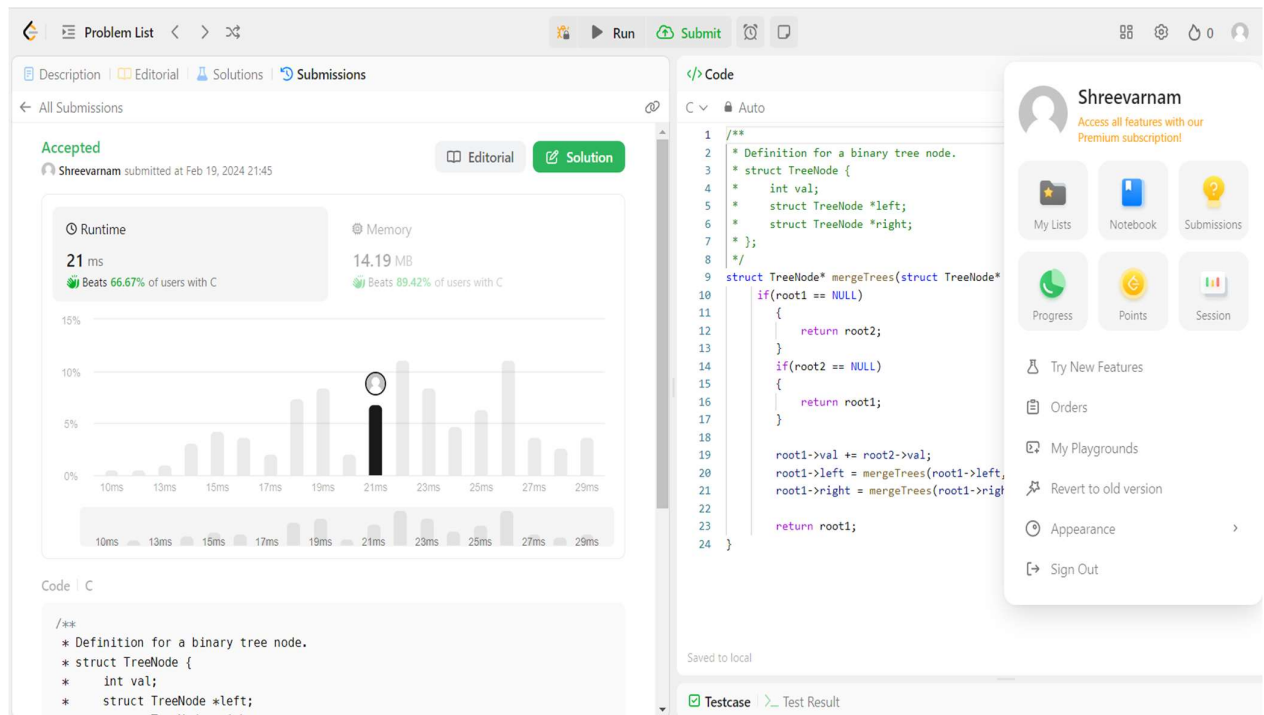
root1->right = mergeTrees(root1->right,root2->right);

return root1;

}

```

OUTPUT:



HR-1 HACKER RANK:Merge two sorted linked lists.

CODE:

```

SinglyLinkedListNode* mergeLists(SinglyLinkedListNode* head1,
SinglyLinkedListNode*
head2)
{
SinglyLinkedListNode *head3 = NULL, *t1 = head1, *t2 = head2,
*t3 = NULL;
while (t1 != NULL && t2 != NULL) {
SinglyLinkedListNode* newNode =

```



```

malloc(sizeof(SinglyLinkedListNode));
if (t1->data < t2->data) {
newNode->data = t1->data;
t1 = t1->next;
} else {
newNode->data = t2->data;
t2 = t2->next;
}
newNode->next = NULL;
if (head3 == NULL) {
head3 = newNode;
t3 = head3;
} else {
t3->next = newNode;
t3 = newNode;
}
}

// If one of the lists is not fully processed, append the remaining
elements to
the merged list.
if (t1 != NULL) {
if (head3 == NULL) {
head3 = t1;
} else {
t3->next = t1;
}
}

```

```

}

if (t2 != NULL) {
if (head3 == NULL) {
head3 = t2;
} else {
t3->next = t2;
}
}

return head3;
}

```

OUTPUT:

The screenshot displays the HackerRank interface for the 'Merge two sorted linked lists' problem. The top navigation bar includes the HackerRank logo and links to 'Prepare', 'Certify', 'Compete', and 'Apply'. A search bar and user profile icon are also present. The problem title 'Merge two sorted linked lists' is prominently displayed, along with a progress indicator showing '25 more points to get your first star' and the user's rank '3772756' and points '5/30'.

Below the problem title, there are tabs for 'Problem', 'Submissions', 'Leaderboard', 'Discussions', and 'Editorial'. The 'Problem' tab is active, showing the problem description: 'You made this submission 17 days ago. Score: 5.00 Status: Accepted'. It also mentions that 'People who solved Merge two sorted linked lists attempted this next:' and provides a 'Get Node Value' challenge link.

The 'Submitted Code' section shows the user's C++ code in a code editor. The code is as follows:

```

Language: C++
74 SinglyLinkedListNode* mergelists(SinglyLinkedListNode* head1,
75 SinglyLinkedListNode* head2) {
76 SinglyLinkedListNode *head3 = NULL, *t1 = head1, *t2 = head2, *t3 = NULL;
77 while (t1 != NULL || t2 != NULL) {

```

On the right side, a user profile menu is open, showing 'Hackos: 111' and a list of navigation options: Profile, Dark Mode (BETA), Leaderboard, Settings, Bookmarks, Network, Submissions, Administration, and Logout. There is also a 'NEED HELP?' section with links to 'View discussions', 'View editorial', and 'View top submissions'.