

29 Jan 2024

Sanjeet Prajwal Pandit (IBM22CS241)

(1) Leet code challenge

The screenshot shows a LeetCode submission interface. On the left, the submission status is 'Accepted' with a green checkmark. The user 'Sanje...' submitted the code on Jan 29, 2024 at 20:27. The submission is compared against other users' solutions, showing a runtime of 3 ms (Beats 57.39% of users with C) and a memory usage of 6.20 MB (Beats 94.39% of users with C). The right side of the screen displays the C code for reversing a singly-linked list. The code defines a 'ListNode' struct with an integer 'val' and a pointer to the next 'ListNode'. The 'reverseList' function takes the head of the list and returns the new head after reversing the list. The algorithm uses a 'prev' pointer to build the reversed list by iterating through the original list and re-linking the nodes.

Accepted

Sanje... submitted at Jan 29, 2024 20:27

Runtime

3 ms

Beats 57.39% of users with C

Memory

6.20 MB

Beats 94.39% of users with C

40%

Code

```
1 /**
2  * Definition for singly-linked list.
3  * struct ListNode {
4  *     int val;
5  *     struct ListNode *next;
6  * };
7  */
8 struct ListNode* reverseList(struct ListNode* head) {
9     struct ListNode* prev = NULL;
10    struct ListNode* current = head;
11    struct ListNode* nextNode = NULL;
12
13    while (current != NULL) {
14        nextNode = current->next;
15        current->next = prev;
16        prev = current;
17        current = nextNode;
18    }
19    return prev;
20 }
```

(2)

(1) Stack using linked list.

Code:

```
#include <stdio.h>

struct node{
    int data;
    struct node* next;
};

void push(struct node**head,int val){
    struct node* newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data = val;
    newnode->next =*head;
    *head = newnode;
}

void pop(struct node**head){
    if (*head == NULL) {
        printf("Stack is empty.\n");
        return;
    }

    struct node* temp = *head;
    *head =(*head)->next;
    printf("%d is popped.",temp->data);
    free(temp);
}

void display(struct Node* head) {
    struct node* temp = head;
    if (temp == NULL) {
        printf("Stack is empty.\n");
        return;
    }
    printf("Elements of Stack are:\n");
    while (temp != NULL) {
        printf("%d\t",temp->data);
        temp = temp->next;
    }
}

int main(){
    struct node* head=NULL;
    int ch,val;
    while(ch!=4){
        printf("\nMenu : 1:Push 2:Pop 3:Display 4:Exit\n");
        scanf("%d",&ch);
        switch(ch){
```

```
case 1:
    printf("Enter the value : ");
    scanf("%d",&val);
    push(&head,val);
    break;
case 2:
    pop(&head);
    break;
case 3:
    display(head);
    break;
case 4:
    return 0;
default:
    printf("Invalid Choice\n");
    break;
}
}
}
```

Output:

```
Menu : 1:Push  2:Pop  3:Display  4:Exit  
1
```

```
Enter the value : 10
```

```
Menu : 1:Push  2:Pop  3:Display  4:Exit  
1
```

```
Enter the value : 20
```

```
Menu : 1:Push  2:Pop  3:Display  4:Exit  
1
```

```
Enter the value : 30
```

```
Menu : 1:Push  2:Pop  3:Display  4:Exit  
3
```

```
Elements of Stack are:
```

```
30      20      10
```

```
Menu : 1:Push  2:Pop  3:Display  4:Exit  
2
```

```
30 is popped.
```

```
Menu : 1:Push  2:Pop  3:Display  4:Exit  
2
```

```
20 is popped.
```

```
Menu : 1:Push  2:Pop  3:Display  4:Exit  
2
```

```
10 is popped.
```

```
Menu : 1:Push  2:Pop  3:Display  4:Exit  
2
```

```
Stack is empty.
```

```
Menu : 1:Push  2:Pop  3:Display  4:Exit  
3
```

```
Stack is empty.
```

```
Menu : 1:Push  2:Pop  3:Display  4:Exit  
4
```

```
Press any key to continue . . . |
```

(2) Queue using Linked list.

Code:

```
#include <stdio.h>

struct node {
    int data;
    struct node* next;
};

void insert(struct node** head, int val) {
    struct node* newnode = (struct node*)malloc(sizeof(struct node));
    struct node* temp = *head;
    newnode->data = val;
    newnode->next = NULL;

    if (*head == NULL) {
        *head = newnode;
        return;
    }

    while (temp->next != NULL) {
        temp = temp->next;
    }

    temp->next = newnode;
}

void delete1(struct node** head) {
    if (*head == NULL) {
        printf("Queue is empty.\n");
        return;
    }

    struct node* temp = *head;
    *head = (*head)->next;
    printf("%d is deleted.",temp->data);
    free(temp);
}

void display(struct Node* head) {
    struct node* temp = head;
    if (temp == NULL) {
        printf("Queue is empty.\n");
        return;
    }
    printf("Elements of Queue are:\n");
    while (temp != NULL) {
        printf("%d\t",temp->data);
        temp = temp->next;
    }
}
```

```
int main(){
    struct node* head=NULL;
    int ch,val;
    while(ch!=4){
        printf("\nMenu : 1:Insert 2:Delete 3:Display 4:Exit\n");
        scanf("%d",&ch);
        switch(ch){
            case 1:
                printf("Enter the value : ");
                scanf("%d",&val);
                insert(&head,val);
                break;
            case 2:
                delete1(&head);
                break;
            case 3:
                display(head);
                break;
            case 4:
                return 0;
            default:
                printf("Invalid Choice\n");
                break;
        }
    }
}
```

Output:

```
Menu : 1:Insert  2:Delete 3:Display  4:Exit  
1
```

```
Enter the value : 10
```

```
Menu : 1:Insert  2:Delete 3:Display  4:Exit  
1
```

```
Enter the value : 20
```

```
Menu : 1:Insert  2:Delete 3:Display  4:Exit  
1
```

```
Enter the value : 30
```

```
Menu : 1:Insert  2:Delete 3:Display  4:Exit  
3
```

```
Elements of Queue are:
```

```
10      20      30
```

```
Menu : 1:Insert  2:Delete 3:Display  4:Exit  
2
```

```
10 is deleted.
```

```
Menu : 1:Insert  2:Delete 3:Display  4:Exit  
2
```

```
20 is deleted.
```

```
Menu : 1:Insert  2:Delete 3:Display  4:Exit  
2
```

```
30 is deleted.
```

```
Menu : 1:Insert  2:Delete 3:Display  4:Exit  
2
```

```
Queue is empty.
```

```
Menu : 1:Insert  2:Delete 3:Display  4:Exit  
4
```

```
Process returned 0 (0x0)    execution time : 27.154 s  
Press any key to continue.
```