18.2.2019

Q. Write a program to construct Binary Search Tree
(a) Traverse the tree using inorder, postorder, preorder.
(b) Display the elements in the tree.

```
#include <stdio.h>
#include <stdlib.h>

struct BST {
        int data;
        struct BST *left, *right;
};
struct BST *root = NULL, *temp;

void create(){
    temp = (struct BST*) malloc (sizeof (struct BST));
    printf("Enter data: ");
    scanf("%d", &temp->data);
    temp->left = temp->right = NULL;
    return temp;
}
void insert (struct BST *root, struct BST *temp)
{
    if (temp->data < root->data)
      { if (root->left != NULL)
            insert (root->left, temp);
        else
            root->left = temp;
        if (temp->data > root->data)
        {
          if (root->right != NULL)
              insert (root->right, temp)
          else
              root->right = temp;
        }
}
```

```
void inorder (struct BST * root)
{ if (root != NULL)
    { inorder (root -> left);
      printf (" %d", root -> data);
      inorder (root -> right);
    }
}

void postorder (struct BST * root)
{ if (root != NULL)
    {
      postorder (root -> left);
      postorder (root -> right);
      printf ("%d", root -> data);
    }
}

void preorder (struct BST * root)
  { if (root != NULL)
      printf ("%d\t", root -> data);
      preorder (root -> left);
      preorder (root -> right);
    }
}

void main ()
{ int choice; char ch;
  printf (" Enter operation \n 1. create \n 2. display using inorder \n
      3. display using postorder \n 4. display using preorder \n
      5. -1 to end \n ");
  while (1)
  {
    printf (" enter operation ");
    scanf (" %d ", &choice);
    if (choice == -1)
```

```
    { exit(1); }
else
{ switch(choice)
  {
    case 1: do
        { temp = create();
        if(root == NULL)
            root = temp;
        else
            insert(root, temp);
        printf("Do you want to enter more
                (Y/n)?");
        getchar();
        scanf("%c", &ch);
        } while(ch=='y'||ch=='Y');

    case 2: printf("elements of tree are:");
            Inorder(root);
    case 3: printf("elements of tree are:");
            Postorder(root);
    case 4: printf("elements of tree are:");
            Preorder(root);
    default: printf("Invalid operation");
    }
  }
}
}
```

// Output :-

1. Enter data into BST
2. Display the elements

Enter choice : 1
Enter data : 5
Enter choice : 1
Enter data : 6

Enter choice: 1
Enter data: 9
Enter choice: 1
Enter data: 4
Enter choice: 1
Enter data: 2
Enter choice: 2
elements using In-order Traversal: 24569
elements using Pre-order Traversal: 54269
elements using Post-order Traversal: 24965

Leetcode demonstration :-

1. Delete the Middle Node of a Linked List

```
struct ListNode* deleteMiddle(struct ListNode* head){
    if (head == NULL)
        return head;
    if (head->next == NULL){
        return NULL;
    struct ListNode *curr = head;
    int size = 0;
    while (curr != NULL){
        size++;
        curr = curr->next;
    }
    curr = head;
    for (int i=0; i < size/2-1; i++){
        curr = curr->next;
    }
    struct ListNode *temp = curr->next;
    curr->next = temp->next;
    free(temp);
    return head;
}
```

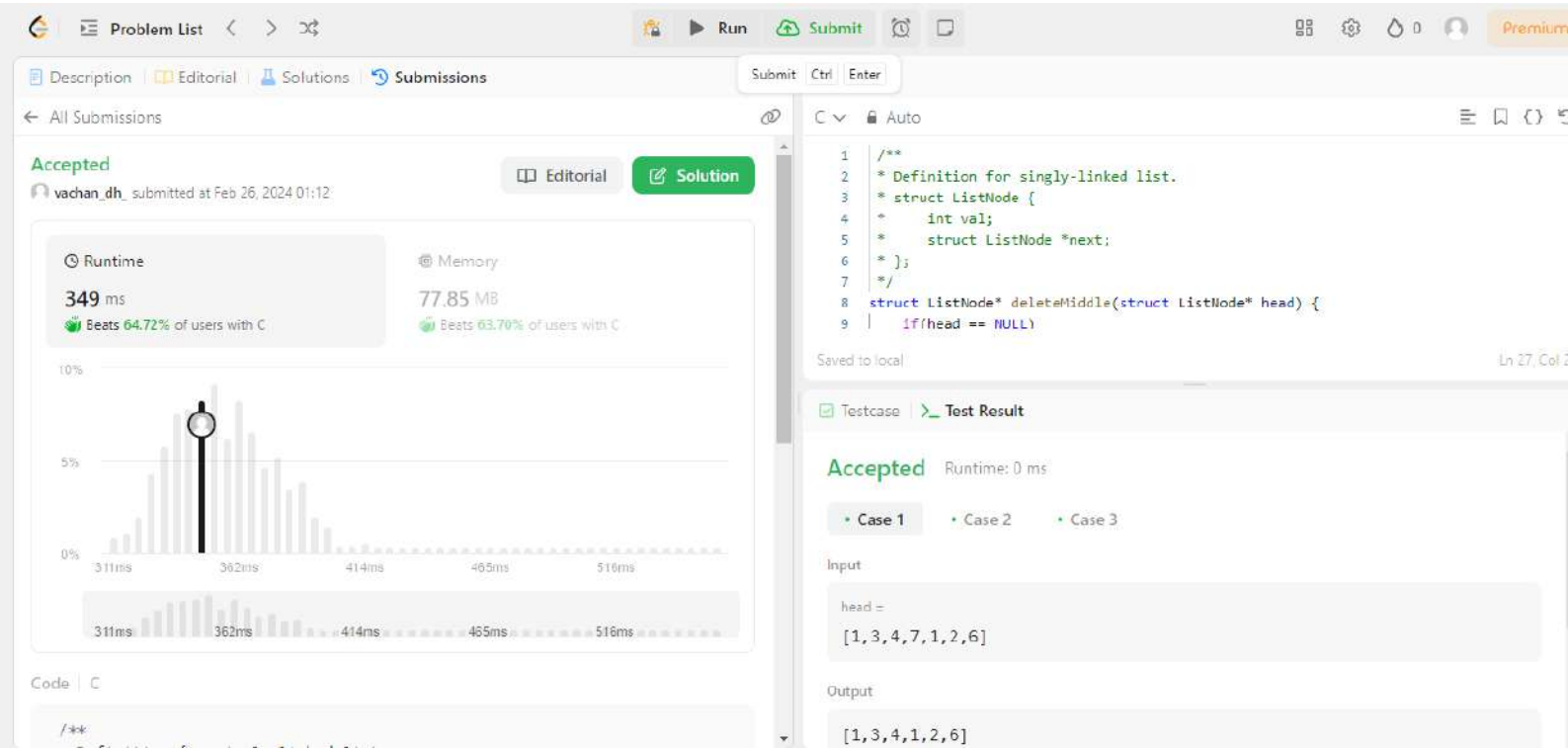
2. Odd even linked list :-

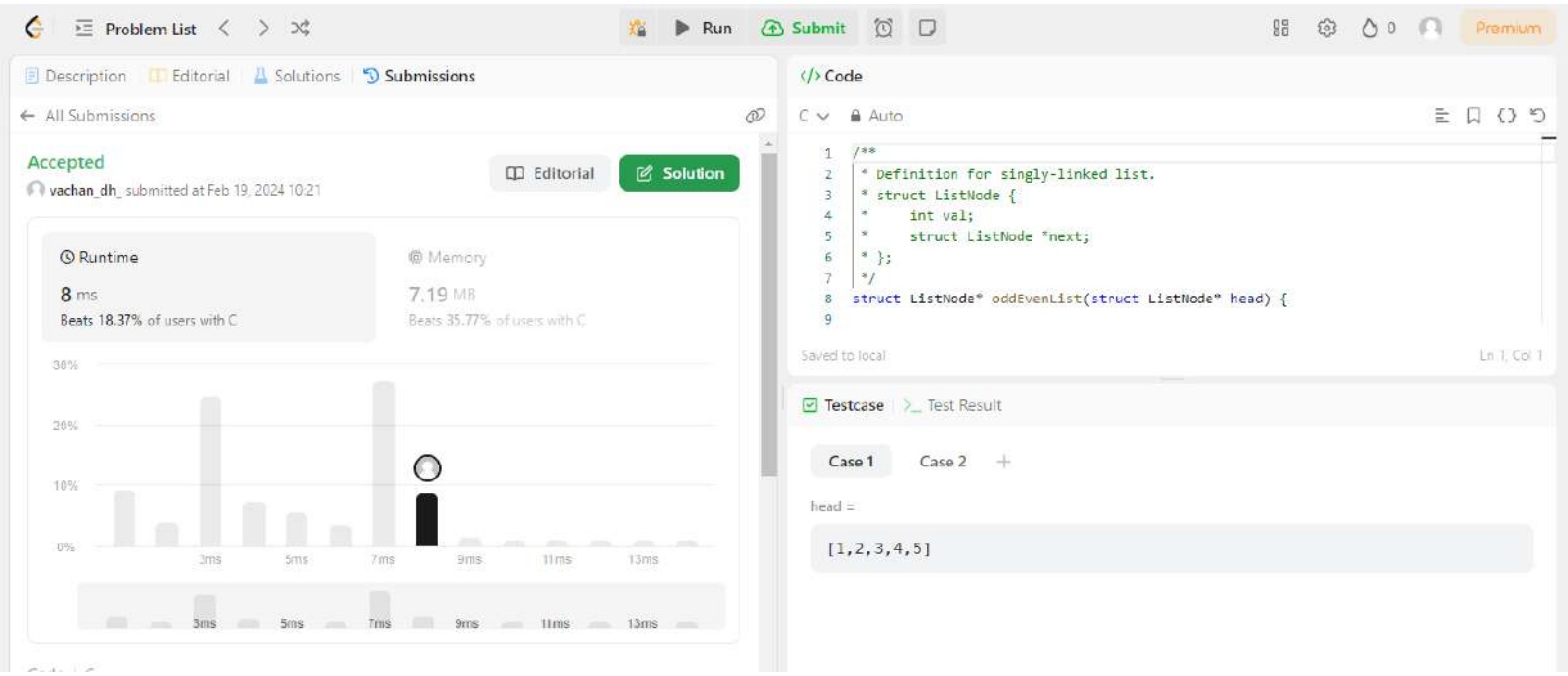
```
struct ListNode {
    int val;
    struct ListNode *next;
};

struct ListNode * oddEvenList (struct ListNode * head){
    if (head == NULL){
        return head;
    }
    struct ListNode * head1;
    struct ListNode * head2;
    struct ListNode * Tail1;
    struct ListNode * Tail2;
    head1 = head2 = Tail1 = Tail2 = NULL;
    for(int i=0 ; head!= NULL ; i++){
        struct ListNode * n = (struct ListNode *) malloc (sizeof(struct ListNode));
        n -> val = head -> val;
        n -> next = NULL;
        if(i%2==0){
            if(head1==NULL){
                head1 = Tail1 = n; }
            else{
                Tail1 -> next = n;
                Tail1 = n;
            }
        }
        else{
            if(head2==NULL){
                head2 = Tail2 = n;
            }else{
                Tail2 -> next = n;
                Tail2 = n;
            }
        }
        head = head -> next;
    }

    Tail1 -> next = head2;
    return head1;

}
```

```
    }
}
head = head -> next;
}

tail1 -> next = head2;
  return head1;

}
```

C:\Users\bmsce\Desktop\1bm

```
1.Enter data into BST
2.display the elements
Enter choice: 1
Enter data: 5
Enter choice: 1
Enter data: 6
Enter choice: 1
Enter data: 9
Enter choice: 1
Enter data: 4
Enter choice: 1
Enter data: 2
Enter choice: 2
elements using In-order Traversal: 2 4 5 6 9
elements using Pre-order Traversal: 5 4 2 6 9
elements using Post-order Traversal: 2 4 9 6 5

Process returned 0 (0x0)   execution time : 41.047 s
Press any key to continue.
```


Problem List
Run
Submit
Premium
Description
Editorial
Solutions
Submissions
Submit Ctrl Enter
All Submissions
Accepted
vachan_dh_ submitted at Feb 26, 2024 01:12
Editorial
Solution
Runtime
349 ms
Beats 64.72% of users with C
Memory
77.85 MB
Beats 63.70% of users with C
10%
5%
0%
311ms
362ms
414ms
465ms
516ms
Code | C
/**
C
Auto
1 /**
2 * Definition for singly-linked list.
3 * struct ListNode {
4 * int val;
5 * struct ListNode *next;
6 * };
7 */
8 struct ListNode* deleteMiddle(struct ListNode* head) {
9 if(head == NULL)
Saved to local
Ln 27, Col
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
head =
[1,3,4,7,1,2,6]
Output
[1,3,4,1,2,6]


Problem List
Run
Submit
Premium
Description
Editorial
Solutions
Submissions
All Submissions
Accepted
vachan_dh_ submitted at Feb 19, 2024 10:21
Editorial
Solution
Runtime
8 ms
Beats 18.37% of users with C
Memory
7.19 MB
Beats 35.77% of users with C
30%
20%
10%
0%
3ms
5ms
7ms
9ms
11ms
13ms
Code
C
Auto
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;
* struct ListNode *next;
* };
*/
struct ListNode* oddEvenList(struct ListNode* head) {
Saved to local
Ln 1, Col 1
Testcase
Test Result
Case 1
Case 2
head =
[1,2,3,4,5]