Prog-12.

9.02.2024

1. Delete the Middle Node of a Linked List.

⇒

```
Struct ListNode * deleteMiddle (struct ListNode *head)
{
struct ListNode *ptr = head;
struct ListNode *freeptr = NULL;
int count = 0;
int n = 0;
if (head = NULL)
{
return head;
}
else if (head -> next == NULL)
{
free (head);
return NULL;
}
else
{
while (ptr != NULL)
  {
  count ++;
ptr = ptr -> next;
}
ptr = head;
while (n != count/2)
{
  preptr = ptr;
  ptr = ptr -> next
  n++;
}
```

19-02-2024.

```
preptr ->next = ptr->next;
free (ptr);
}
} return (head);
```

2. Odd even linked list

⇒

```
Struct ListNode* oddevenList (struct ListNode* head)
{
if (head == NULL || head -> next == NULL)
{
return head;
}
Struct ListNode * oddNode = head;
Struct ListNode * evenNode = oddNode -> next;
Struct ListNode * headeven = evenNode;
while (evenNode != NULL && evenNode -> next != NULL)
{
oddNode -> next = evenNode -> next;
oddNode = oddNode -> next;
evenNode -> next = oddNode -> next;
evenNode = evenNode -> next;
}
oddNode -> next = headeven;
return head;
}
```

Description Editorial Solutions **Submissions**

← All Submissions

Accepted

user0435PL submitted at Feb 26, 2024 09:33

Editorial Solution

Runtime

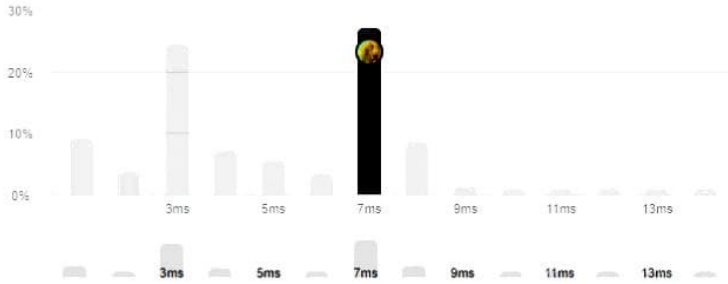
7 ms

Beats **45.53%** of users with C

Memory

6.53 MB

Beats 95.61% of users with C



Code C

```c
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
#include <stdio.h>
```

View more

More challenges

- **725. Split Linked List in Parts**

</> Code

C Auto

```c
}

odd->next = evenHead;
return head;
}
struct ListtNode* newNode(int val)
{
struct ListtNode* node = (struct ListtNode*)malloc(sizeof(struct ListtNode));
node->val = val;
node->next = NULL;
return node;
}

```

Saved to local

Ln 35, Col 1

Testcase Test Result

Case 1 Case 2 +

head =

```
[1,2,3,4,5]
```

</> Source

Description Editorial Solutions **Submissions**

All Submissions

Accepted

user0435PL submitted at Feb 26, 2024 10:31

Editorial Solution

Runtime

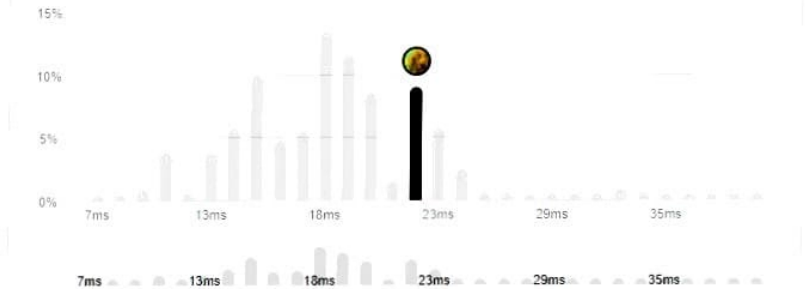
22 ms

Beats **28.62%** of users with C

Memory

13.78 MB

Beats 56.27% of users with C



Code | C

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     struct TreeNode *left;
 *     struct TreeNode *right;
 * };
 */
```

View more

More challenges

- **776. Split BST**

Write your notes here

</> Code

C Auto

```
48  free(root);
49  return temp;
50  }
51
52
53  struct TreeNode* temp = minValueNode(root->right);
54
55
56  root->val = temp->val;
57
58
59  root->right = deleteNode(root->right, temp->val);
60  }
61  return root;
62  }
63
```

Saved to local

Ln 63, Col

Testcase | **Test Result**

Accepted Runtime: 5 ms

- Case 1
- Case 2
- **Case 3**

Input

root =

[]

key =

0

Output

[]

Expected

[]

Prog-13

19-02-2024

write a program to construct, traverse (inorder, postorder, preorder) & display.

=>
```
typedef struct BST {
int data;
struct BST *left;
struct BST *right;
} node;
node * create ()
{
node *temp;
printf ("enter the data :");
temp = (node *) malloc (size of (node));
scanf ("%d", &temp->data);
temp-> left = temp -> right = NULL;
return temp;
}
void insert (node * root , node * temp)
{
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 -> left = temp;
}
```

19: 19/02/2024

```
if (temp -> data > root -> data)
{
if (root -> right != NULL)
  insert (root -> right, temp)
else
  root -> right = temp;
}
}
void inorder (node *root)
{
 if (root != NULL)
 {
 inorder (root -> left)
 printf("%d", root -> data);
 inorder (root -> right);
  }
 }
void postorder (node* root)
{
 if (root != NULL)
 {
 postorder (root -> left);
 printf ("%d", root -> data);
 postorder (root -> right);
  }
 }
void preorder (node* root)
{
 if (root != NULL)
  {
```

10-02-2024

```
printf("%d", root->data);
preorder(root->lyt);
preorder(root->right);
}
}
void main()
{
root = NULL;
root = insert(root,10);
root = insert(root, 20);
root = insert(root, 30);
root = insert(root, 40);
root = insert(root, 50);
root = insert(root, 60);
root = insert(root, 70);
printf("insertion succesful \n");
inorder(root);
printf("\n");
preorder(root);
printf("\n");
postorder(root);
printf("\n");
}
```

p. Insertion Succesfull

10	20	30	40	50	60	70
10	50	20	60	70	30	40
50	70	60	40	30	20	10

```
1. Preorder
2. Inorder
3. Postorder
4. Exit
Choice: 1
8 3 1 6 4 7 10 14 13
Enter choice: 2
1 3 4 6 7 8 10 13 14
Enter choice: 3
1 4 7 6 3 13 14 10 8
Enter choice: 4

Process returned 0 (0x0)   execution time : 14.294 s
Press any key to continue.
```