

C 4_1_rotate_linked_list.c > displaylist(node *)

```
1  /* Given a linked list of n nodes and an integer k, write a function to rotate the linked list counter clockwise by k nodes. */
2
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  struct node{
7      int data;
8      struct node *next;
9  };
10
11 struct node *createnode(int value){
12     struct node *newnode=(struct node *)malloc(sizeof(struct node));
13     newnode->data=value;
14     newnode->next=NULL;
15     return newnode;
16 }
17
18 struct node *createlinkedlist(int n){
19     struct node *head=NULL,*temp=NULL,*newnode=NULL;
20     int value;
21     for(int i=1;i<=n;i++){
22         printf("Enter data for node %d : ",i);
23         scanf("%d",&value);
24         newnode=createnode(value);
25         if(head==NULL){
26             head=newnode;
27         }
28         else{
29             temp->next=newnode;
30         }
31         temp=newnode;
32     }
33     return head;
34 }
35
36 void displaylist(struct node *head){
37     struct node *temp=head;
38     while(temp!=NULL){
39         printf("%d || %u, ",temp->data,temp->next);
40         temp=temp->next;
41     }
42 }
43
```

```

44 void rotate(struct node **head,int k){
45     int length=1;
46     struct node *current=*head;
47     while(current->next!=NULL){
48         length++;
49         current=current->next;
50     }
51     struct node *end=current;
52     struct node *split=*head;
53     for(int i=1;i<k;i++){
54         split=split->next;
55     }
56     struct node *newhead=split->next;
57     split->next=NULL;
58     end->next=*head;
59     *head=newhead;
60 }
61
62 int main(){
63     int n;
64     printf("Enter length of linked list : ");
65     scanf("%d",&n);
66     struct node *head=createlinkedlist(n);
67     int k;
68     printf("Enter k : ");
69     scanf("%d",&k);
70     printf("Linked list before rotating :--\n");
71     displaylist(head);
72     rotate(&head,k);
73     printf("\nLinked list after rotating counterclockwise by %d :--\n",k);
74     displaylist(head);
75     return 0;
76 }

```

TERMINAL

COMMENTS

Enter length of linked list : 5

Enter data for node 1 : 1

Enter data for node 2 : 2

Enter data for node 3 : 3

Enter data for node 4 : 4

Enter data for node 5 : 5


Enter k : 3

Linked list before rotating :--

1 || 7248464, 2 || 7248528, 3 || 7249328, 4 || 7249392, 5 || 0,

Linked list after rotating counterclockwise by 3 :--

4 || 7249392, 5 || 7237104, 1 || 7248464, 2 || 7248528, 3 || 0,

C 4_2_remove_duplicate.c >  displaylist(node *)

```
1  /* Given an unsorted linked list of n nodes, remove duplicates from the list. */
2
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  struct node{
7      int data;
8      struct node *next;
9  };
10
11 struct node *createnode(int value){
12     struct node *newnode=(struct node *)malloc(sizeof(struct node));
13     newnode->data=value;
14     newnode->next=NULL;
15     return newnode;
16 }
17
18 struct node *createlinkedlist(int n){
19     struct node *head=NULL,*temp=NULL,*newnode=NULL;
20     int value;
21     for(int i=1;i<=n;i++){
22         printf("Enter data for node %d : ",i);
23         scanf("%d",&value);
24         newnode=createnode(value);
25         if(head==NULL){
26             head=newnode;
27         }
28         else{
29             temp->next=newnode;
30         }
31         temp=newnode;
32     }
33     return head;
34 }
35
36 void displaylist(struct node *head){
37     struct node *temp=head;
38     while(temp!=NULL){
39         printf("%d || %u, ",temp->data,temp->next);
40         temp=temp->next;
41     }
42 }
43
```

C 4_2_remove_duplicate.c > removeduplicate(node **)

```
44 void removeduplicate(struct node **head){
45     struct node* current=*head;
46     while(current!=NULL){
47         struct node* runner=current->next;
48         struct node* prevrunner=current;
49         while(runner!=NULL){
50             if(runner->data==current->data){
51                 prevrunner->next=runner->next;
52                 free(runner);
53                 runner=prevrunner->next;
54             }
55             else{
56                 prevrunner=runner;
57                 runner=runner->next;
58             }
59         }
60         current=current->next;
61     }
62 }

63
64 int main(){
65     int n;
66     printf("Enter length of linked list : ");
67     scanf("%d",&n);
68     struct node *head=createlinkedlist(n);
69     printf("Linked list initially :--\n");
70     displaylist(head);
71     removeduplicate (&head);
72     printf("\nLinked list after removing duplicate elements :--\n");
73     displaylist(head);
74     return 0;
75 }
```

TERMINAL COMMENTS

Enter length of linked list : 6

Enter data for node 1 : 1

Enter data for node 2 : 2

Enter data for node 3 : 1

Enter data for node 4 : 2

Enter data for node 5 : 3

Enter data for node 6 : 4

Linked list initially :--

1 || 8315728, 2 || 8315792, 1 || 8315856, 2 || 8297888, 3 || 8297952, 4 || 0,

Linked list after removing duplicate elements :--

1 || 8315728, 2 || 8297888, 3 || 8297952, 4 || 0,

C 4_3_detect_loop.c > detectloop(node *)

```
1  /* Given a singly linked list of n nodes, detect if it contains a loop or not. */
2
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  struct node{
7      int data;
8      struct node *next;
9  };
10
11 struct node *createnode(int value){
12     struct node *newnode=(struct node *)malloc(sizeof(struct node));
13     newnode->data=value;
14     newnode->next=NULL;
15     return newnode;
16 }
17
18 int detectloop(struct node *head){
19     struct node *temp=head->next;
20     while(temp!=NULL){
21         if(temp==head){
22             return 1;
23         }
24         temp=temp->next;
25     }
26     return 0;
27 }
28
```

```

29 int main(){
30     // test
31     struct node *head=createnode(1);
32     struct node *second=createnode(2);
33     struct node *third=createnode(3);
34     struct node *end=createnode(4);
35     head->next=second;
36     second->next=third;
37     third->next=end;
38     end->next=head; // looped linked list
39     if(detectloop(head)){
40         printf("Given linked list is looped\n");
41     }
42     else{
43         printf("Given linked list is NOT looped\n");
44     }
45
46     end->next=NULL; // Straight linked list
47     if(detectloop(head)){
48         printf("Given linked list is looped\n");
49     }
50     else{
51         printf("Given linked list is NOT looped (straight)\n");
52     }
53     return 0;
54 }

```

TERMINAL COMMENTS

```

Given linked list is looped
Given linked list is NOT looped (straight)

```

C 4_4_insertion_dlinked_list.c > main()

```
1  /* Write a C/C++ program to implement doubly linked list with the following function :--
2
3  (i) insertAtFirst(&head, new_data): This function should insert the new data/element at the beginning of the linked list.
4
5  (ii) insertAtEnd(&head, new_data): This function should insert the new data/element at the end of the linked list
6
7  (iii) insertAtMiddle(&head, new_data): This function should insert the new data/element at the middle of the linked list
8
9  (iv) InsertAfterNode(&head, given_node, new_data): This function should insert the new data/element after the given node in the linked list.
10
11 Example: Suppose, you want to insert 60 after node 40 in the given linked list :--
12 10 <- -> 20 <- -> 30 <- -> 40 <- -> 50
13 the updated linked list will be :--
14 10 <- -> 20 <- -> 30 <- -> 40 <- -> 60 <- -> 50
15
16 (v) display(&head): This function should display the content of the linked list
17
18 Note:--
19 1. If the linked list has 4 elements, let's say 10, 20, 30, and 40, the linked list
20 should be displayed in the following format 10 <- -> 20 <- -> 30 <- -> 40
21 2. After each operation, you should display the content of the linked list. */
22
23 #include <stdio.h>
24 #include <stdlib.h>
25
26 struct node{
27     int data;
28     struct node *next,*prev;
29 };
30
31 struct node *createnode(int value){
32     struct node *newnode=(struct node *)malloc(sizeof(struct node));
33     newnode->data=value;
34     newnode->next=NULL;
35     newnode->prev=NULL;
36     return newnode;
37 }
38
```

```

38
39 int getlen(struct node *head){
40     int n=0;
41     struct node *temp=head;
42     while(temp!=NULL){
43         n++;
44         temp=temp->next;
45     }
46     return n;
47 }
48
49 void display(struct node *head){
50     struct node *temp=head;
51     while(temp->next!=NULL){
52         printf("%d <- -> ",temp->data);
53         temp=temp->next;
54     }
55     printf("%d",temp->data);
56 }
57
58 void insertAtFirst(struct node **head,int new_data){
59     struct node *newnode=createnode(new_data);
60     newnode->next=*head;
61     (*head)->prev=newnode;
62     *head=newnode;
63 }
64
65 void insertAtEnd(struct node **head,int new_data){
66     struct node *newnode=createnode(new_data);
67     struct node *end=*head;
68     while(end->next!=NULL){
69         end=end->next;
70     }
71     end->next=newnode;
72     newnode->prev=end;
73     end=newnode;
74 }
75

```


75

```
76 void insertAtMiddle(struct node **head,int new_data){
77     struct node *newnode=createnode(new_data);
78     struct node *mid=*head;
79     int n=getlen(*head);
80     for(int i=1;i<n/2;i++){
81         mid=mid->next;
82     }
83     struct node *midplus=mid->next;
84     mid->next=newnode;
85     newnode->prev=mid;
86     newnode->next=midplus;
87     midplus->prev=newnode;
88 }
```

89

```
90 void insertAfterNode(struct node **head,struct node **given,int new_data){
91     struct node *newnode=createnode(new_data);
92     struct node *givenplus=(*given)->next;
93     (*given)->next=newnode;
94     newnode->prev=*given;
95     newnode->next=givenplus;
96     givenplus->prev=newnode;
97 }
```

98

```

99  int main(){
100     // creating a doubly linked list for testing
101     struct node *head=createnode(1);
102     struct node *second=createnode(2);
103     struct node *third=createnode(3);
104     struct node *end=createnode(4);
105     head->prev=NULL;
106     head->next=second;
107     second->prev=head;
108     second->next=third;
109     third->prev=second;
110     third->next=end;
111     end->prev=third;
112     end->next=NULL;
113     printf("\nLinked list before any insertion :--\n");
114     display(head);
115     insertAtFirst(&head,0); // inserted 0 at beginning
116     printf("\nLinked list after inserting 0 at beginning :--\n");
117     display(head);
118     insertAtEnd(&head,5); // inserted 5 at end
119     printf("\nLinked list after inserting 5 at end :--\n");
120     display(head);
121     insertAtMiddle(&head,99); // inserted -1 at middle
122     printf("\nLinked list after inserting 99 at middle :--\n");
123     display(head);
124     insertAfterNode(&head,&second,69); // inserted 69 after node containing value=2
125     printf("\nLinked list after inserting 69 after node containing value=2 :--\n");
126     display(head);
127     return 0;
128 }
129
130

```

TERMINAL COMMENTS

Linked list before any insertion :--

1 <- -> 2 <- -> 3 <- -> 4

Linked list after inserting 0 at beginning :--

0 <- -> 1 <- -> 2 <- -> 3 <- -> 4

Linked list after inserting 5 at end :--

0 <- -> 1 <- -> 2 <- -> 3 <- -> 4 <- -> 5

Linked list after inserting 99 at middle :--

0 <- -> 1 <- -> 2 <- -> 99 <- -> 3 <- -> 4 <- -> 5

Linked list after inserting 69 after node containing value=2 :--

0 <- -> 1 <- -> 2 <- -> 69 <- -> 99 <- -> 3 <- -> 4 <- -> 5