

```

Linked_List.c
Saved

1 #include<stdio.h>
2 struct node
3 {
4     int info;
5     struct node *link;
6 };
7 typedef struct node *NODE;
8 NODE getnode()
9 {
10     NODE x;
11     x=(NODE)malloc(sizeof(struct node));
12     if(x==NULL)
13     {
14         printf("Memory full\n");
15         exit(0);
16     }
17     return x;
18 }
19 void freenode(NODE x)
20 {
21     free(x);
22 }
23 NODE insert_front(NODE first,int item)
24 {
25     NODE temp;
26     temp=getnode();
27     temp->info=item;
28     temp->link=NULL;
29     if(first==NULL)
30         return temp;
31     temp->link=first;
32     first=temp;
33     return first;
34 }
35 NODE delete_front(NODE first)
36 {
37     NODE temp;
38     if(first==NULL)
39     {
40         printf("list is empty cannot delete\n");
41         return first;
42     }
43     temp=first;
44     temp=temp->link;
45     printf("Item deleted at front-end is=%d\n",first->info);
46     free(first);
47     return temp;
48 }
49 NODE insert_rear(NODE first,int item)
50 {
51     NODE temp,cur;
52     temp=getnode();
53     temp->info=item;
54     temp->link=NULL;
55     if(first==NULL)
56         return temp;
57     cur=first;
58     while(cur->link!=NULL)
59         cur=cur->link;
60     cur->link=temp;
61     return first;
62 }
63 NODE delete_rear(NODE first)
64 {
65     NODE cur,prev;
66     if(first==NULL)
67     {
68         printf("List is Empty. Cannot delete!\n");
69         return first;
70     }
71     if(first->link==NULL)
72     {
73         printf("Item deleted is %d\n",first->info);
74         free(first);
75         return NULL;
76     }
77     prev=NULL;
78     cur=first;
79     while(cur->link!=NULL)
80     {
81         prev=cur;
82         cur=cur->link;
83     }
84     printf("Item deleted at rear-end is %d",cur->info);
85     free(cur);
86     prev->link=NULL;
87     return first;
88 }
89 void display(NODE first)
90 {
91     NODE temp;
92     if(first==NULL)
93         printf("List empty. Cannot display items!\n");
94     for(temp=first;temp!=NULL;temp=temp->link)
95     {
96         printf("%d\n",temp->info);
97     }
98 }
99 void main()
100 {
101     int item,choice,pos;
102     NODE first=NULL;
103     for(;;)
104     {
105         printf("\n1.Insert_front\n2.Delete_front\n3.Insert_rear\n4.Delete_rear\n5.display_list\n6.Exit\n");
106         printf("Enter Choice:");
107         scanf("%d",&choice);
108         switch(choice)
109         {
110             case 1:
111                 printf("Enter the item at front-end: ");
112                 scanf("%d",&item);
113                 first=insert_front(first,item);
114                 break;
115             case 2:
116                 first=delete_front(first);
117                 break;
118             case 3:
119                 printf("Enter the item at rear-end: ");
120                 scanf("%d",&item);
121                 first=insert_rear(first,item);
122                 break;
123             case 4:
124                 first=delete_rear(first);
125                 break;
126             case 5:
127                 display(first);
128                 break;
129             case 6:
130                 exit(0);
131                 break;
132             default:
133                 printf("Enter a Valid Choice!!");
134         }
135     }
136 }
137
138

```

```

x Terminal
1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:1
Enter the item at front-end: 10

1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:1
Enter the item at front-end: 9

1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:3
Enter the item at rear-end: 20

1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:3
Enter the item at rear-end: 21

1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:5
9
10
20
21

1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:2
item deleted at front-end is=9

1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:2
item deleted at front-end is=10

1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:4
Item deleted at rear-end is 21
1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:4
Item deleted is 20

1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:4
List is Empty. Cannot delete!

1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:5
List empty. Cannot display items!

1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:7
Enter a Valid Choice!!
1.Insert_front
2.Delete_front
3.Insert_rear
4.Delete_rear
5.display_list
6.Exit
Enter Choice:6

Process finished.

```

WAP to implement Singly Linked List with following operations
 a) Create a linked List. b) Insertion of a node at first position, at any position and at end of list. c) Display the contents of l-lists d) delete the first element, specified element (and last element in the list).

code:

```
#include <stdio.h>
struct node
{
    int info;
    struct node *link;
};
typedef struct node *NODE;
NODE getnode ()
{
    NODE x;
    x = (NODE) malloc (sizeof (struct node));
    if (x == NULL)
    {
        printf ("Mem full \n");
        exit (0);
    }
    return x;
}
void freenode (NODE x)
{
    free (x);
}
NODE insert_near (NODE first, int item)
{
    NODE temp, cur;
```

temp = getnode();

temp → info = item;

temp → link = NULL;

if (first == NULL)

return temp;

cur = first;

while (cur → link != NULL)

cur = cur → link;

cur → link = temp;

return first;

NODE delete_rear (NODE first)

{
NODE cur, prev;

if (first == NULL)

{ printf ("List is Empty Cannot Delete \n");

return first;

}
if (first → link == NULL)

{ printf ("Item deleted is %d \n", first → info);

free (first);

return NULL;

}

prev = NULL;

cur = first;

while (cur → link != NULL)

{

prev = cur;

cur = cur → link;

}

printf ("Item Deleted at rear-end is %d", cur → info);

free (cur);

prev → link = NULL;

IBM19CS090

RAHIL

return first;

}
NODE insert_pos (int item, int pos, NODE first)
{

NODE temp, cur, prev;

int count;

temp = getnode ();

temp -> info = item;

temp -> link = NULL;

if (first == NULL && pos == 1)

{
 return temp;

}
if (first == NULL)

{
 printf ("Invalid ~~Exp~~ Position\n");
 return first;

}

if (pos == 1)

{
 temp -> link = first;
 first = temp;

 return temp;
}

count = 1;

prev = NULL;

cur = first;

while (cur != NULL && count != pos)

{
 prev = cur;
 cur = cur -> link;
 count ++;

}

```

if (count == pos)
{
    prev → link = temp;
    temp → link = cur;
    return first;
}
printf("Invalid position\n");
return first;
}
NODE delete_pos (int pos, NODE first)
{
    NODE cur;
    NODE prev;
    int count, flag = 0;
    if (first == NULL || pos < 0)
    {
        printf("Invalid position\n");
        return NULL;
    }
    if (pos == 1)
    {
        cur = first;
        first = first → link;
        free node (cur);
        return first;
    }
    prev = NULL;
    cur = first;
    count = 1;
    while (cur != NULL)
    {
        if (count == pos) { flag = 1; break; }
        count ++;
        prev = cur;
    }

```



```

        cur = cur -> link;
    }
    if (flag == 0)
    {
        printf("Invalid position \n");
        return first;
    }
    printf("Item deleted at given position is %d \n", cur -> info);
    prev -> link = cur -> link;
    freeNode(cur);
    return first;
}

void display (NODE first)
{
    NODE temp;
    if (first == NULL)
        print("list empty cannot display items \n");
    for (temp = first; temp != NULL; temp = temp -> link)
    {
        printf("%d \n", temp -> info);
    }
}

void main()
{

```