```c
/*WAP to Implement Singly Linked List with following operations
a) a) Create a linked list. b) Deletion of first element, specified element and last element in
the list. c) Display the contents of the linked list.*/
#include <stdio.h>
#include <stdlib.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("Memory full\n");
        exit(0);
    }
    return x;
}
void freenode(NODE x)
{
    free(x);
}
NODE insert_rear(int item,NODE first)
{
    NODE temp,cur;
    temp=getnode();
    temp->info=item;
    temp->link=NULL;
    if (first==NULL)
    {
        return temp;
```

```
}
NODE insert_rear(int item,NODE first)
{
    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;
```

line: 68 / 167 col: 16 sel: 0 INS TAB mode: CRLF encoding: UTF-8 filetype: C scope: delete_rear

```
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("iten deleted at rear-end is %d",cur->info);
free(cur);
prev->link=NULL;
return first;
}
NODE delete_front(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
temp=first;
temp=temp->link;
printf("item deleted at front-end is=%d\n",first->info);
free(first);
return temp;
}
NODE delete_pos(int pos,NODE first)
{
    NODE prev,cur;
    int count;
    if (first==NULL || pos<=0)
```

```
}
NODE delete_pos(int pos,NODE first)
{
    NODE prev,cur;
    int count;
    if (first==NULL || pos<=0)
    {
        printf("Invalid position\n");
        return NULL;
    }
    if (pos==1)
    {
        cur=first;
        first=first->link;
        freenode(cur);
        return first;
    }
    prev=NULL;
    cur=first;
    count=1;
    while (cur!=NULL)
    {
        if (count==pos)
        {
            break;
        }
        prev=cur;
        cur=cur->link;count++;
    }
    if (count!=pos)
    {
        printf("Invalid position\n");
        return first;
    }
    prev->link=cur->link;
```

```
    if (count!=pos)
    {
        printf("Invalid position\n");
        return first;
    }
    prev->link=cur->link;
    freenode(cur);
    return first;
}
void display(NODE first)
{
    NODE temp;
    if (first==NULL)
    {
        printf("Linked is empty cannot display items\n");
    }
    printf("The contents of the linked list are:\n");
    for (temp=first;temp!=NULL;temp=temp->link)
    {
        printf("%d\n",temp->info);
    }
}

int main()
{
int item,choice,pos;
NODE first=NULL;
for(;;)
{
printf("\n 1:Insert_rear\n 2:Delete_front\n 3:Delete_rear\n4:Delete at specified position 5:Display_list\n6:Exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the item at rear-end\n");
```

```
}

int main()
{
int item,choice,pos;
NODE first=NULL;
for(;;)
{
printf("\n 1:Insert_rear\n 2:Delete_front\n 3:Delete_rear\n4:Delete at specified position 5:Display_list\n6:Exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the item at rear-end\n");
scanf("%d",&item);
first=insert_rear(item,first);
break;
case 2:first=delete_front(first);
break;
case 3:first=delete_rear(first);
break;
case 4:printf("Enter the position:\n");
                    scanf("%d",&pos);
                    first=delete_pos(pos,first);
                    break;
break;
case 5:display(first);
break;
default:exit(0);
break;
}
}
}
```

```
 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
1
enter the item at rear-end
10

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
1
enter the item at rear-end
20

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
1
enter the item at rear-end
30

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
1
enter the item at rear-end
40
```

```
enter the choice
1
enter the item at rear-end
40

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
1
enter the item at rear-end
50

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
5
The contents of the linked list are:
10
20
30
40
50

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
2
item deleted at front-end is=10

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
```

```
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
2
item deleted at front-end is=10

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
5
The contents of the linked list are:
20
30
40
50

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
3
iten deleted at rear-end is 50
 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
5
The contents of the linked list are:
20
30
40

 1:Insert_rear
 2:Delete_front
```

```
20
30
40

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
4
Enter the position:
2

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
5
The contents of the linked list are:
20
40

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
2
item deleted at front-end is=20

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
3
```

```
 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
2
item deleted at front-end is=20

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
3
item deleted is 40

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
2
list is empty cannot delete

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
6


------------------
(program exited with code: 0)

Press any key to continue . . .
```