Singly linked list (insert & delete)

```
#include <stdio.h>
#include <malloc.h>
struct node
{
    int info;
    struct node *next;
};
typedef struct node * NODE;

NODE getnode()
{
    NODE temp;
    temp = (NODE) malloc (sizeof (struct node));
    if (temp == NULL)
    {
        Printf("memory full\n");
        return NULL;
    }
    return temp;
}
void freenode (NODE temp)
{
    free (temp);
}
NODE insert front (NODE first)
{
    int num;
    printf("Enter number\n");
    scanf ("%d", &num);
    NODE temp;
    temp = getnode();
```

```
temp->info = num;
temp->next = NULL;
if (first==NULL)
{
    return temp;
}
temp->next = first
first = temp;
return first;
}
NODE insertrear (NODE first)
{
int num;
printf ("Enter the number \n");
scanf ("%d", &num);
NODE curr, temp;
temp = getnode();
temp->info = num;
temp->next = NULL;
if (first==NULL)
{
    return temp;
}
curr = first;
while (curr->next != NULL)
{
    curr = curr->next;
}
curr->next = temp;
return first;
}
NODE insertpos (NODE first, int pos)
{
```

```
int num;
printf("Enter num\n");
scanf("%d", &num);
NODE temp, curr, prev;
int count = 1;
temp = getnode();
temp->info = num
temp->next = NULL
if (pos == count)
        return temp;
curr = first;
prev = NULL
while (curr != NULL)
{
    if (count == pos)
        break;
    prev = curr;
    curr = curr->next;
    count++;
}
if (curr == NULL)
{
    pf("Entered pos > length\n");
    return first;
}
temp->next = prev->next;
prev->next = temp;
return first;
}
void display(NODE first)
{
    if (first == NULL)
```

```
{   pf("List is empty\n");
    return;
}
NODE cuss = first
while (cuss != NULL)
{
    printf("%d", cuss -> info);
    cuss = cuss -> next;
}
pf("\n");

NODE deletefront(NODE first)
{
    NODE cuss;
    if (first == NULL)
    {
        printf("List is empty\n");
        return first;
    }
    if (first -> next == NULL)
    {
        printf("Del element = %d", first -> info);
        freenode(first);
        return NULL;
    }
    cuss = first;
    cuss = cuss -> next;
    printf("Delelement = %d\n", first -> info);
    freenode(first);
    return cuss;
}
```

Camlin | Page
Date / /

```
NODE deletefrom(NODE first)
{
    NODE curr;
    if (first==NULL)
    {
        Pf("\nList empty\n");
        return first;
    }
    if (first->next==NULL)
    {
        Pf("\nDeleted element=%d\n", first->info);
        free node(first);
        return NULL;
    }
    curr = first;
    curr=curr->next;
    Pf("\nDel element=%d\n", first->info);
    freenode(first);
    return curr;
}
node deletefear(NODE first)
{
    NODE curr, prev=NULL;
    if (first==NULL)
    {
        Pf("\nEmpty list\n");
        return first;
    }
    if(first->next=NULL)
    {
        Pf("Del element=%d\n", first->info);
        freenode(first);
        return NULL;
    }
```

```
curr = first;
while (curr->next != NULL)
{
    prev = NULL;
    Pf ("Deleted ele = %d\n", curr->info);
    freenode (curr);
    return first;
}
NODE delete pos (NODE first, int pos)
{
    NODE curr, prev;
    if (first == NULL)
    {
        Pf ("Empty list\n");
        return first;
    }
    int count = 1;
    if (pos == count)
    {
        Pf ("Del element = %d\n", first->info);
        curr = first;
        curr = curr->next;
        freenode (first);
        return curr;
    }
    curr = first;
    prev = NULL;
    while (curr != NULL)
    {
        if (count == pos)
            break;
        prev = curr;
        curr = curr->next;
```

```
        count++;
    }
    if (cosr==NULL)
    {
        pf("Ent pos > length \n");
        return first;
    }
    prev->next = cosr->next;
    pf("Del element = %d \n", cosr->info);
    freenode(cosr);
    return(first);
}

int main()
{
    int ch;
    int pos;
    NODE first=NULL;
    while(1)
    {
        pf("1->insertpos 2->insertrear \n 3 insertpos
            \n 4. deletefront \n 5. deleterear
            \n 6. deletepos \n 7- disp \n 8. exit \n");
        scanf("%d", &ch);
        switch(ch)
        {
            case 1: first = insertfront(first)
                    break;
            case 2: first = insertrear(first)
                    break;
            case 3: pf("Enter pos to insert \n");
                    scanf("%d", &pos);
                    first = insertpos(first, pos);
```

```
        break;
case 4: first = deletefront(first);
        break;
case 5: first = deleterear(first);
        break;
case 6: pf("Enter pos");
        scanf("%d", &pos);
        first = delete pos(first, pos);
        break;
case 7: display(first);
        break;
case 8: return 0;
        break;
}
}
```