```c
#include<stdio.h>
#include<malloc.h>
#include<stdlib.h>
typedef struct node
{
        int data;
        struct node *next;
}node;
typedef struct freq
{
        int no;
        int count;
}func;
node* createnode(int d)
{
        node *nd;
        nd=(node*)malloc(sizeof(node));
        nd->data=d;
        nd->next=NULL;
        return nd;
}
node* createlist()
{
        int data;char ans;
        node *first,*last,*nd;
        first=NULL;
        while(1)
        {
                printf("enter data:-");
                scanf("%d",&data);
                nd=     createnode(data);
                if(first==NULL)
                        first=nd;
                else
                        last->next=nd;
                last=nd;
                printf("do you want to continue(Y\N)?");
                fflush(stdin);
                scanf("%c",&ans);
                if(ans!='y' && ans!='Y')
                        break;
        }
        return first;
}
void insertatbeg(node **first)
{
        node *nd;
        int d;
        printf("enter data");
        scanf("%d",&d);
        nd=createnode(d);
        nd->next=*first;
        *first=nd;
}
void insertatlast(node *first)
```

```
{
    node *nd,*t;
    int d;
    printf("enter data");
    scanf("%d",&d);
    nd=createnode(d);
    t=first;
    while(t->next!=NULL)
    {
        t=t->next;
    }
    t->next=nd;
    nd->next=NULL;
}
void insertanywhere(node **first)
{
    node *nd,*prev,*t;
    prev=NULL;
    t=*first;
    int d,c=0,pos;
    printf("enter data");
    scanf("%d",&d);
    printf("enter position");
    scanf("%d",&pos);
    nd=createnode(d);
    while(t!=NULL)
    {
        c++;
        if(c==pos)
            break;
        prev=t;
        t=t->next;
    }
    nd->next=t;
    if(prev!=NULL)
        prev->next=nd;
    else
        *first=nd;
}
void delfirst(node **first)
{
    node *t;
    t=*first;
    *first=t->next;
    free(t);
}
void frequency(node *first)
{
    int i=0,j,n=0;
    func *a;
    node *t=first;
    while(t!=NULL)
    {
        t=t->next;
        n++;
    }
    a=(func*)malloc(n*sizeof(func));
    t=first;
    while(t!=NULL)
    {
        for(j=0;j<i;j++)
            if(a[j].no==t->data)
            {
                a[j].count++;
                break;
            }
        if(j==i)
        {
            a[i].no=t->data;
            a[i].count=1;
            i++;
            }
        t=t->next;
    }
    for(j=0;j<i;j++)
        printf("Freq(%d)=%d\n",a[j].no,a[j].count);
}
void dellast(node *first)
{
        node *t,*prev;
        t=first;
        prev=NULL;
        while(t->next!=NULL)
        {
            prev=t;
            t=t->next;
        }
        prev->next=NULL;
        free(t);
}
void delanywhere(node **first)
{
    node *prev,*t;
    int pos,c=0;
    prev=NULL;
    t=*first;
    printf("enter position");
    scanf("%d",&pos);
    while(t!=NULL)
    {
        c++;
        if(c==pos)
            break;
        prev=t;
        t=t->next;
    }
    if(prev!=NULL)
    {
        prev->next=t->next;
        free(t);
```

```
                    }
                    else
                    {
                              *first=t->next;
                              free(t);
                    }
}
void printlist(node *first)
{
          node *t=first;
          while(t!=NULL)
          {
                    printf("%d\t",t->data);
                    t=t->next;
           }
}
int main()
{
          int c; node *start;
          start=NULL;
          while(1)
          {
                    printf("1.create\n2.print\n3.Insert at first\n4.Insert at last\n5.Insert
anywhere\n6.Delete from first\n7.Delete from last\n8.Delete from any position\n9.show
frequency\n10.exit\n");
                    printf("enter choice:-");
                    scanf("%d",&c);
                    switch(c)
                    {
                              case 1: start=createlist();
                                                              break;
                              case 2:  printlist(start);
                                                              break;
                              case 3: insertatbeg(&start);
                                                    break;
                              case 4: insertatlast(start);
                                                    break;
                              case 5: insertanywhere(&start);
                                                    break;
                              case 6: delfirst(&start);
                                                    break;
                              case 7: dellast(start);
                                                    break;
                              case 8: delanywhere(&start);
                                                    break;
                              case 9: frequency(start);
                                                    break;

                              case 10: exit(0);
                    }
          }
}
```