Code:

```
#include <stdio.h>
#include<stdlib.h>

struct node{
int info;
struct node *previous;
struct node *next;
};

//Display all the elements
void display(struct node *first)
{
   struct node *p;
   p=first;
   while(p!=NULL){
      printf("%d\t",p->info);
      p=p->next;
   }
}

//insert element in the empty linked
struct node* insertemp(int x,struct node *start){
   struct node *temp;
```

```
    temp=(struct node *)malloc(sizeof(struct node));
    temp->info=x;
    temp->previous=NULL;
    temp->next=NULL;
    start=temp;
    return start;
}
//Insert element at the end
struct node* insertend(int x,struct node *start){
    struct node *temp,*p;
    temp=(struct node *)malloc(sizeof(struct node));
    temp->info=x;
    p=start;
    while(p->next!=NULL){
        p=p->next;
    p->next=temp;
    temp->previous=p;
    temp->next=NULL;
    return start;
    }
}
//Insert element in the front
struct node* insertfront(int x,struct node *start){
    struct node *temp;
    temp=(struct node *)malloc(sizeof(struct node));
    temp->info=x;
    temp->previous=NULL;
    temp->next=start;
    start=temp;
    return start;
}
//Insert element at choice

struct node *nposition(int x,int y,struct node *start){
    int i;
    struct node *temp,*p;
    temp=(struct node *)malloc(sizeof(struct node));
    temp->info=x;
    if(y==1){
        temp->previous=NULL;
        temp->next=start;
        start->previous=temp;
        start=temp;
        return start;
    }
    p=start;
    for(i=1;i<y-1 && p!=NULL;i++){
        p=p->next;
    }
    if(p==NULL){
        printf("There are less than %d element",y);
    }
    else{
        temp->previous=p;
        temp->next=p->next;
        if(p->next!=NULL){
            p->next->previous=temp;
        }
        p->next=temp;
    }

      return start;
}
//Delete element in the front
struct node* deletefront(struct node *start){
    struct node *temp;
    temp=(struct node *)malloc(sizeof(struct node));
    temp=start;
    start=start->next;
    start->previous=NULL;
    free(temp);
    return start;
```

```
}

//Delete any element in the series
struct node* deleteany(int x,struct node *start){
   struct node *temp,*p;
   temp=(struct node *)malloc(sizeof(struct node));
   p=start;
   while(p->next!=NULL){
      if(p->next->info==x){
         temp=p->next;
         p->next=temp->next;
         temp->next->previous=temp->previous;
         free(temp);
      }
      p=p->next;
   }
   return start;
}

struct node* deleteend(int x,struct node *start){
   struct node *temp,*p;
   temp=(struct node *)malloc(sizeof(struct node));
   p=start;
   while(p->next!=NULL){
        if(p->next->info==x){
            temp=p->next;
            p->next=temp->next;
            free(temp);
            return start;
        }
        p=p->next;
    }
}

int main()
{
    struct node *first;
    first = (struct node *)malloc(sizeof(struct node));
    //Insert element in empty list
    first=insertemp(2,first);
    printf("\nthe elements are\n");
    display(first);
    //Insert element at front of the list
    first=insertfront(4,first);
    printf("\nThe elements are\n");
    display(first);
    //Insert elements at end of the list
    first=insertend(80,first);
    printf("\nThe elements are\n");
    display(first);
    first=nposition(10,2,first);//Insert
    printf("\nThe elements are\n");
    display(first);
    first=deletefront(first);//Delete first element in the list
    printf("\nThe elements are\n");
    display(first);
    first=insertfront(22,first);
    printf("\nThe elements are\n");
    display(first);
    first=insertfront(1,first);
    printf("\nThe elements are\n");
    display(first);
    first=deleteany(22,first); //Delete any element from the list
    printf("\nThe elements are\n");
    display(first);
    //Delete the element fromm the end
    first=deleteend(80,first);
    printf("\nThe elements are\n");
    display(first);
    return 0;
}
```

Output:

```
the elements are
2
The elements are
4       2
The elements are
4       2       80
The elements are
4       10      2       80
The elements are
10      2       80
The elements are
22      10      2       80
The elements are
1       22      10      2       80
The elements are
1       10      2       80
The elements are
1       10      2
```