Implementation of singly list

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int e;
struct node*next;
};
typedef struct node Node;
int isempty(Node * list);
void display(Node*list);
Node *findnext(Node * list,int x);
Node *findprev(Node * list,int x);
void insbeg(Node * list,int x);
void insmid(Node * list,int x);
void insend(Node * list,int x);
void delbeg(Node *list);
void delmid(Node *list);
void delend(Node *list);
int main()
{
int ch,x;
Node *list;
list=malloc(sizeof(Node));
list->next=NULL;
while(1){
printf("1.insert element at begining\n2.insert element at middle\n3.insert element at end\n4.delete
element at begining\n5.delete element at middle\n6.delete element at end\n7.display
elements\n8.Find an element\n9.exit\nEnter your choice: ");
scanf("%d",&ch);
switch(ch)
  {
case 1: printf("Enter element to be inserted:\n");
        scanf("%d",&x);
        insbeg(list,x);
        display(list);
        break;
case 2: printf("Enter element to be inserted:\n");
        scanf("%d",&x);
        insmid(list,x);
        display(list);
        break;
case 3: printf("Enter element to be inserted:\n");
        scanf("%d",&x);
        insend(list,x);
        display(list);
        break;
case 4: delbeg(list);
        display(list);
        break;
```

```
case 5: delmid(list);
        display(list);
        break;
case 6: delend(list);
        display(list);
        break;
case 7: display(list);
        break;
case 8: printf("Enter the element to be found: ");
        scanf("%d",&ch);
        if(findnext(list,ch)!=NULL)
                        {
                        printf("The element is found!!\n");
                        }
                        else
                        {
                        printf("The element is not found!!\n");
                        }
        break;
case 9: exit(0);
default:printf("Invalid input");
        break;
  }
 }
}
int isempty(Node * list)
{
if(list->next==NULL)
   return 1;
else
           return 0;
}
void display(Node*list)
{
           Node * position=list;
           if(!isempty(list)){
           printf("The elements are:\n");
           while(position->next!=NULL)
           {
                        position=position->next;
                        printf("%d ",position->e);
           }
           printf("\n");
           }
 else
 {
           printf("the list is empty!!!\n");
 }
}
Node *findnext(Node * list,int x)
  {
           if(isempty(list))
```

```
        {
        printf("The list is empty!!\n");
        return NULL;
        }
        else{
        Node * position=list->next;
        while(position!=NULL)
        {
        if(position->e==x)
        {break;}
        position=position->next;
        }
        if(position!=NULL)
        {return position;}
        else
        {  printf("The element is not found in the list.\n");
           return NULL;
        }
  }
 }
 Node *findprev(Node * list,int x)
  {
        if(isempty(list))
        {
        printf("The list is empty!!\n");
        return NULL;
        }
        else{
        Node * position=list;
        while(position!=NULL)
        {
        if(position->next->e==x)
        {break;}
        position=position->next;
        }
        if(position!=NULL)
        {return position;}
        else
        {  printf("The element is not found in the list.\n");
           return NULL;
  }
 }
}
void insbeg(Node * list,int x)
{
 Node * newnode;
 newnode=malloc(sizeof(Node));
 newnode->e=x;
 newnode->next=list->next;
 list->next=newnode;
}
void insmid(Node * list,int x)
```

```
{
 int p;
 Node * newnode;
 Node * position;
 position=list;
 newnode=malloc(sizeof(Node));
 newnode->e=x;
 printf("next to which element the element should be inserted:");
 scanf("%d",&p);
 position=findnext(list,p);
 if(position!=NULL)
 {
 newnode->next=position->next;
 position->next=newnode;
 }
}
void insend(Node * list,int x)
{
 Node * newnode;
 Node * position;
 position=list;
 newnode=malloc(sizeof(Node));
 newnode->e=x;
 while(position->next!=NULL)
            {
              position=position->next;
            }
 newnode->next=position->next;
 position->next=newnode;
}
void delbeg(Node *list)
{
   if(!isempty(list))
    list->next=list->next->next;
}
void delmid(Node *list)
{
            if(!isempty(list))
            {
            Node*position;
            int pos;
            position=list;
            printf("what element should be deleted:");
            scanf("%d",&pos);
            position=findprev(list,pos);
            position->next=position->next->next;
            }
}
void delend(Node *list)
{
            if(!isempty(list))
            {
```

```
            Node*position;
            position=list;
            while(position->next->next!=NULL)
            {position=position->next;}
             position->next=position->next->next;
  }
}
```

Output:

```
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 1
Enter element to be inserted:
1
The elements are:
1
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 2
Enter element to be inserted:
2
next to which element the element should be inserted:1
The elements are:
1 2
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 3
Enter element to be inserted:
3
The elements are:
```

1 2 3
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 4
The elements are:
2 3
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 5
what element should be deleted:2
The elements are:
3
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 3
Enter element to be inserted:
1
The elements are:
3 1
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 2
Enter element to be inserted:
1

next to which element the element should be inserted:1
The elements are:
3 1 1
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 7
The elements are:
3 1 1
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 8
Enter the element to be found: 1
The element is found!!
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 9

Implementation of Double list

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
        int e;
        struct node*next;
        struct node*prev;
};
typedef struct node Node;
int isempty(Node *list);
Node *find(Node * list,int x);
void display(Node*list);
void insbeg(Node * list,int x);
void insmid(Node * list,int x,int p);
void insend(Node * list,int x);
void delbeg(Node *list);
void delmid(Node *list,int p);
void delend(Node *list);
int main()
{
int ch,x,p;
Node *list;
list=malloc(sizeof(Node));
list->next=NULL;
while(1){
printf("1.insert element at begining\n2.insert element at middle\n3.insert element at end\n4.delete
element at begining\n5.delete element at middle\n6.delete element at end\n7.display
elements\n8.Find an element\n9.exit\nEnter your choice: ");
scanf("%d",&ch);
switch(ch)
 {
case 1: printf("Enter element to be inserted: ");
      scanf("%d",&x);
      insbeg(list,x);
      display(list);
      break;
case 2: printf("Enter element to be inserted: ");
      scanf("%d",&x);
                printf("Enter element next to which the element to be inserted:\n");
                scanf("%d",&p);
      insmid(list,x,p);
      display(list);
      break;
case 3: printf("Enter element to be inserted: ");
      scanf("%d",&x);
      insend(list,x);
      display(list);
      break;
```

```
case 4: delbeg(list);
        display(list);
        break;
case 5: printf("Enter element to be deleted: ");
                        scanf("%d",&p);
        delmid(list,p);
        display(list);
        break;
case 6: delend(list);
        display(list);
        break;
case 7: display(list);
        break;
case 8: printf("Enter the element to be found: ");
        scanf("%d",&ch);
        if(find(list,ch)!=NULL)
                        {
                        printf("The element is found!!\n");
                        }
                        else
                        {
                        printf("The element is not found!!\n");
                        }
        break;
case 9: exit(0);
default:printf("Invalid input\n");
        break;
  }
 }
}
int isempty(Node *list)
{
            if(list->next==NULL)
    return 1;
    else
            return 0;
}
Node *find(Node * list,int x)
   {
            if(isempty(list))
            {
            printf("The list is empty!!\n");
            return NULL;
            }
            else{
            Node * position=list->next;
            while(position!=NULL)
            {
            if(position->e==x)
            {break;}
            position=position->next;
            }
```

```
        if(position!=NULL)
        {return position;}
        else
        {  printf("The element is not found in the list.\n");
           return NULL;
        }
   }
 }
void display(Node*list)
{
        Node * position=list;
        if(!isempty(list)){
        printf("The elements are:\n");
        while(position->next!=NULL)
        {
                position=position->next;
                printf("%d ",position->e);
        }
        printf("\n");
        }
 else
 {
        printf("the list is empty!!!\n");
 }
}
void insbeg(Node * list,int x)
{
 Node * newnode;
 newnode=malloc(sizeof(Node));
 newnode->e=x;
 newnode->next=list->next;
 if(!isempty(list))
 {newnode->next->prev=newnode;}
 list->next=newnode;
 newnode->prev=list;
}
void insmid(Node * list,int x,int p)
{
 Node * newnode;
 Node * position;
 position=list;
 newnode=malloc(sizeof(Node));
 newnode->e=x;
 position=find(list,p);
 if(position!=NULL)
 {
 newnode->next=position->next;
 position->next=newnode;
 newnode->prev=position;
 newnode->next->prev=newnode;
 }
}
```

```
void insend(Node * list,int x)
{
 Node * newnode;
 Node * position;
 position=list;
 newnode=malloc(sizeof(Node));
 newnode->e=x;
 while(position->next!=NULL)
           {
             position=position->next;
           }
 newnode->next=NULL;
 position->next=newnode;
 newnode->prev=position;
}
void delbeg(Node *list)
{
if(!isempty(list))
{
list->next=list->next->next;
if(list->next!=NULL)
list->next->prev=list;
 }
}
void delmid(Node *list,int p)
{
           if(!isempty(list))
           {
           Node*position;
           position=list;
           position=find(list,p);
           if(position!=NULL)
           {
           position->prev->next=position->next;
           position->next->prev=position->prev;
           }
  }
}
void delend(Node *list)
{
           if(!isempty(list))
           {
           Node*position;
           position=list;
           while(position->next->next!=NULL)
           {position=position->next;}
            position->next=NULL;
  }
}
```

Output:

```
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 1
Enter element to be inserted: 7
The elements are:
7
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 3
Enter element to be inserted: 9
The elements are:
7 9
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 2
Enter element to be inserted: 8
Enter element next to which the element to be inserted:
7
The elements are:
7 8 9
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
```

Enter your choice: 4
The elements are:
8 9
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 5
Enter element to be deleted: 7
The element is not found in the list.
The elements are:
8 9
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 6
The elements are:
8
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 1
Enter element to be inserted: 4
The elements are:
4 8

Implementation of circular list:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int e;
struct node*next;
};
typedef struct node Node;
Node * end;
int isempty(Node * list);
void display(Node*list);
Node *findnext(Node * list,int x);
Node *findprev(Node * list,int x);
void insbeg(Node * list,int x);
void insmid(Node * list,int x);
void insend(Node * list,int x);
void delbeg(Node *list);
void delmid(Node *list);
void delend(Node *list);
int main()
{
int ch,x;
Node *list;
list=malloc(sizeof(Node));
list->next=NULL;
end=list;
while(1){
printf("1.insert element at begining\n2.insert element at middle\n3.insert element at end\n4.delete
element at begining\n5.delete element at middle\n6.delete element at end\n7.display
elements\n8.Find an element\n9.exit\nEnter your choice: ");
scanf("%d",&ch);
switch(ch)
  {
case 1: printf("Enter element to be inserted:\n");
        scanf("%d",&x);
        insbeg(list,x);
        display(list);
        break;
case 2: printf("Enter element to be inserted:\n");
        scanf("%d",&x);
        insmid(list,x);
        display(list);
        break;
case 3: printf("Enter element to be inserted:\n");
        scanf("%d",&x);
        insend(list,x);
        display(list);
        break;
case 4: delbeg(list);
        display(list);
```

```
        break;
case 5: delmid(list);
        display(list);
        break;
case 6: delend(list);
        display(list);
        break;
case 7: display(list);
        break;
case 8: printf("Enter the element to be found: ");
        scanf("%d",&ch);
        if(findnext(list,ch)!=NULL)
                        {
                        printf("The element is found!!\n");
                        }
                        else
                        {
                        printf("The element is not found!!\n");
                        }
        break;
case 9: exit(0);
default:printf("Invalid input");
        break;
  }
 }
}
int isempty(Node * list)
{
if(list->next==NULL)
   return 1;
else
           return 0;
}
void display(Node*list)
{
           Node * position=list;
           if(!isempty(list)){
           printf("The elements are:\n");
           while(position->next!=list)
           {
                        position=position->next;
                        printf("%d ",position->e);
           }
           printf("\n");
           }
 else
 {
           printf("the list is empty!!!\n");
 }
}
Node *findnext(Node * list,int x)
  {
```

```
            if(isempty(list))
            {
            printf("The list is empty!!\n");
            return NULL;
            }
            else{
            Node * position=list->next;
            while(position!=list)
            {
            if(position->e==x)
            {break;}
            position=position->next;
            }
   return position;
  }
 }
 Node *findprev(Node * list,int x)
  {
            if(isempty(list))
            {
            printf("The list is empty!!\n");
            return NULL;
            }
            else{
            Node * position=list;
            while(position->next!=list)
            {
            if(position->next->e==x)
            {break;}
            position=position->next;
            }
   return position;
 }
}
void insbeg(Node * list,int x)
{
 Node * newnode;
 newnode=malloc(sizeof(Node));
 newnode->e=x;
 if(isempty(list))
 { newnode->next=list;}
 else
 {newnode->next=list->next;}
 list->next=newnode;
}
void insmid(Node * list,int x)
{
 int p;
 Node * newnode;
 Node * position;
 position=list;
 newnode=malloc(sizeof(Node));
```

```
 newnode->e=x;
 printf("next to which element the element should be inserted:");
 scanf("%d",&p);
 position=findnext(list,p);
 newnode->next=position->next;
 position->next=newnode;
}
void insend(Node * list,int x)
{
 Node * newnode;
 Node * position;
 position=list;
 newnode=malloc(sizeof(Node));
 newnode->e=x;
 while(position->next!=list)
            {
              position=position->next;
            }
 newnode->next=position->next;
 position->next=newnode;
}
void delbeg(Node *list)
{
   if(!isempty(list))
    list->next=list->next->next;
}
void delmid(Node *list)
{
            if(!isempty(list))
            {
            Node*position;
            int pos;
            position=list;
            printf("what element should be deleted:");
            scanf("%d",&pos);
            position=findprev(list,pos);
            position->next=position->next->next;
            }
}
void delend(Node *list)
{
            if(!isempty(list))
            {
            Node*position;
            position=list;
            while(position->next->next!=list)
            {position=position->next;}
             position->next=list;
  }
}
```

Output:

1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 1
Enter element to be inserted:
5
The elements are:
5
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 3
Enter element to be inserted:
2
The elements are:
5 2
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 2
Enter element to be inserted:
4
next to which element the element should be inserted:5
The elements are:
5 4 2
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit

```
Enter your choice: 4
The elements are:
4 2
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 6
The elements are:
4
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 1
Enter element to be inserted:
3
The elements are:
3 4
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 5
what element should be deleted:3
The elements are:
4
1.insert element at begining
2.insert element at middle
3.insert element at end
4.delete element at begining
5.delete element at middle
6.delete element at end
7.display elements
8.Find an element
9.exit
Enter your choice: 9
```