Double Linked List

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node *lptr,*rptr;
};
struct node* #include <stdio.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node *lptr,*rptr;
};
struct node* createNode(int x)
{
    struct node *temp;
    temp=(struct node*)malloc(sizeof(struct node));
    temp->data=x;
    return temp;
}
void insert(struct node *header, int x)
{
    struct node *temp,*p;
    temp=createNode(x);
    p=header->rptr;
    temp->rptr=header->rptr;
    header->rptr=temp;
    temp->lptr=header;
    if(p!=NULL)
     p->lptr=temp;
}
void display(struct node *header)
{
    struct node *ptr,*end;
    ptr=header->rptr;

    while(ptr!=NULL)
    {
        printf("%d ",ptr->data);
        end=ptr;
        ptr=ptr->rptr;
     }printf("\n");
    while(end!=header)
    {
        printf("%d ",end->data);
        end=end->lptr;
     }printf("\n");
     printf("\n");
}
```

```
int search(struct node *header,int key)
{
    struct node *ptr;
    ptr=header->rptr;
    while(ptr!=NULL)
    {
        if(ptr->data==key)
          return ptr->data;
        else
         ptr=ptr->rptr;
    }
   return -1;
}

void insertAfter(struct node *header,int key,int x)
{
    struct node *ptr,*temp,*r;
    ptr=header->rptr;
    temp=createNode(x);
    while(ptr!=NULL)
    {
        if(ptr->data==key)
         {
             r=ptr->rptr;
             ptr->rptr=temp;
             //printf("\n %d",ptr->rptr->data);
             temp->lptr=ptr;
             //printf("\n %d",temp->lptr->data);
             temp->rptr=r;
             //printf("\n %d",temp->rptr->data);
             r->lptr=temp;
             //printf("\n %d",r->lptr->data);
             //break;
         }
        else
         ptr=ptr->rptr;
    }
}
void delete(struct node *header, int num)
{
    struct node *ptr,*l,*r;
    ptr=header->rptr;
    while(ptr!=NULL)
    {
        if(ptr->data==num)
        {
           l=ptr->lptr;
           r=ptr->rptr;
           l->rptr=r;
           r->lptr=l;
           free(ptr);
        }
        else
        {
            ptr=ptr->rptr;
        }
    }
}
int main()
{
    struct node *header;
    header=(struct node*)malloc(sizeof(struct node));
    header->lptr=header->rptr=NULL;
    insert(header,10);
    insert(header,20);
    insert(header,30);
    insert(header,40);
    insert(header,50);
    display(header);
    delete(header,50);
    display(header);

    return 0;
}(int x)
{
    struct node *temp;
    temp=(struct node*)malloc(sizeof(struct node));
    temp->data=x;
    return temp;
}
void insert(struct node *header, int x)
{
    struct node *temp,*p;
    temp=createNode(x);
    p=header->rptr;
    temp->rptr=header->rptr;
    header->rptr=temp;
    temp->lptr=header;
    if(p!=NULL)
     p->lptr=temp;
}
void display(struct node *header)
{
    struct node *ptr,*end;
    ptr=header->rptr;

    while(ptr!=NULL)
    {
        printf("%d ",ptr->data);
        end=ptr;
        ptr=ptr->rptr;
     }printf("\n");
    while(end!=header)
    {
        printf("%d ",end->data);
        end=end->lptr;
     }printf("\n");
     printf("\n");
}

int search(struct node *header,int key)
{
    struct node *ptr;
    ptr=header->rptr;
    while(ptr!=NULL)
    {
        if(ptr->data==key)
          return ptr->data;
        else
         ptr=ptr->rptr;
    }
   return -1;
}

void insertAfter(struct node *header,int key,int x)
{
    struct node *ptr,*temp,*r;
    ptr=header->rptr;
    temp=createNode(x);
    while(ptr!=NULL)
    {
        if(ptr->data==key)
         {
             r=ptr->rptr;
             ptr->rptr=temp;
             //printf("\n %d",ptr->rptr->data);
             temp->lptr=ptr;
             //printf("\n %d",temp->lptr->data);
             temp->rptr=r;
             //printf("\n %d",temp->rptr->data);
             r->lptr=temp;
             //printf("\n %d",r->lptr->data);
             //break;
         }
        else
         ptr=ptr->rptr;
    }
}
void delete(struct node *header, int num)
{
    struct node *ptr,*l,*r;
    ptr=header->rptr;
    while(ptr!=NULL)
    {
        if(ptr->data==num)
        {
           l=ptr->lptr;
           r=ptr->rptr;
           l->rptr=r;
           r->lptr=l;
           free(ptr);
        }
        else
        {
            ptr=ptr->rptr;
        }
    }
}
int main()
{
    struct node *header;
    header=(struct node*)malloc(sizeof(struct node));
    header->lptr=header->rptr=NULL;
    insert(header,10);
    insert(header,20);
    insert(header,30);
    insert(header,40);
    insert(header,50);
    display(header);
    delete(header,50);
    display(header);

    return 0;
}
```

```
Output:

50 40 30 20 10-after insert
10 20 30 40 50-print rev

40 30 20 10 - after delete
10 20 30 40 - printf rev
```