```
#include <iostream>
#include <cstdlib>
#include<string.h>
using namespace std;



class node{

      public:
        int data;
        node *next;

        node(){
            next=NULL;
            data=0;
        }

        node(int i, node *in) {
            data = i;
              next = in;
        }

           node(int i){
            data=i;
            next=NULL;
        }
};


class list
{
    private:
        node *head, *tail;

      public:
        list(){
            head=NULL;
            tail=NULL;
      }

      list(const list &rhs){   //copy constructor
            this->head=NULL;
          this->tail=NULL;
          node *q=rhs.head;
          while(q!=NULL){
              add_node_tail(q->data);
              q=q->next;
          }
      }

      list& operator=(const list& rhs){    //assignment operator
          if(this!=&rhs){
                      node *ptr;
                      /*instead of writing this line and while loop
                      we can simply use destructor as this->~list();*/
                while(this->head!=NULL){
                    ptr=this->head->next;
                    delete this->head;
                    this->head=ptr;
```

```
        }

        this->head=NULL;
      this->tail=NULL;
      node *q=rhs.head;
      while(q!=NULL){
          add_node_tail(q->data);
          q=q->next;
      }
     }
               return *this;


 }

void add_node_head(int n)
 {
     if(head == NULL)  {
        head=tail=new node(n);
     }
     else{
        head=new node(n,head);
     }
 }



   void add_node_tail(int n){
 //add to tail
        if(tail!=NULL)
     {
         tail->next = new node(n);
         tail = tail->next;
        }
     else{
         head=tail=new node(n);
        }
 }


 void display()
 {
     node *temp=new node;
     temp=head;
     while(temp!=NULL)  {
         cout<<temp->data<<endl;
         temp=temp->next;
     }
 }

 void delete_head() {           //delete first node
     if (head!=NULL){
         int delnode = head->data;
         node *tmp = head;
         if (head == tail){
             head = tail = NULL;
         }
         else{
             head = head->next;
         }
```

```
            delete tmp;
        }
        else{
            cout<<"list is empty";
        }
    }


    void delete_tail() {    //delete last node
        int delnode = tail->data;
        if (head == tail) {
            delete head;
            head=tail=NULL;
        }
        else{
            node *tmp;
            for(tmp = head; tmp->next != tail; tmp = tmp->next);
            delete tail;
            tail = tmp;
            tail->next = NULL;
        }
    }

    ~list(){               //destructor
        node *ptr;
        while(head!=NULL){
            ptr=head->next;
            delete head;
            head=ptr;
        }
    }

      void reverse(){
        node *current = head;
        node *prev = NULL, *next = NULL;
        while (current != NULL)  {
            next = current->next;
            current->next = prev;
            prev = current;
            current = next;
        }
        head = prev;
    }

    bool searching(int a){
        node *tmp;
        for(tmp=head;tmp!=NULL && tmp->data!=a;tmp=tmp->next);
        return tmp!=NULL;
    }

};

int main()
{
    list a;
    a.add_node_head(1);
    //adding nodes to a
      a.add_node_head(2);
    a.add_node_tail(4);
    a.add_node_tail(5);
```

```
    a.display();
    //a.delete_tail();
    //a.display();
    //a.delete_head();
    //a.display();
      a.reverse();
      //a.display();

    list b(a);  //case of copy constructor
      b.display();
      list c;
      c.add_node_head(6);
      c.add_node_head(7);
      c=a;        //case of operator assignment
      c.display();

      cout<<c.searching(9);
      return 0;

}
```