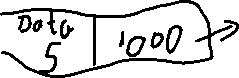
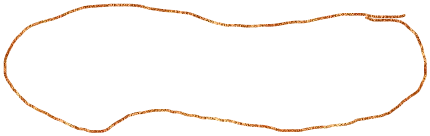
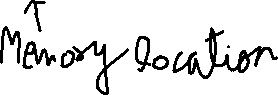
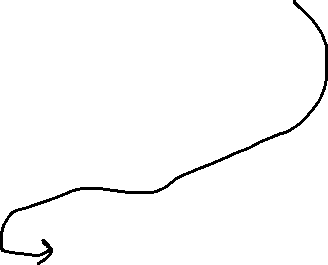
**Linked list**

* Formation of linked list

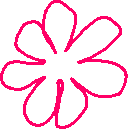


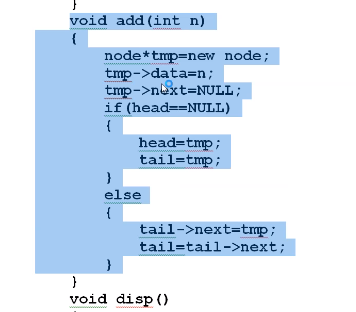












#include <iostream>

using namespace std;

struct node

{

    int data;

    node \*next;

};

class Link

{

    node \*head, \*tail;

    public:

    Link()

    {

        head=NULL;

        tail=NULL;

    }

    void add(int n)

    {

        node \*temp= new node;

        temp->data= n;

        temp->next= NULL;

        if(head==NULL)

        {

            head=temp;

            tail=temp;

        }

        else

        {

            tail->next= temp;

            tail=tail->next;

        }

    }

    void disp()

    {

        while (head!=NULL)

        {

            cout<<head->data<< "\t"<<head->next<< endl;

            head=head->next;

        }

    }

};

int main()

{

    cout<<"Linked List :"<<endl;

    Link lobj;

    lobj.add(5);

    lobj.add(10);

    lobj.add(15);

    lobj.add(20);

    lobj.add(25);

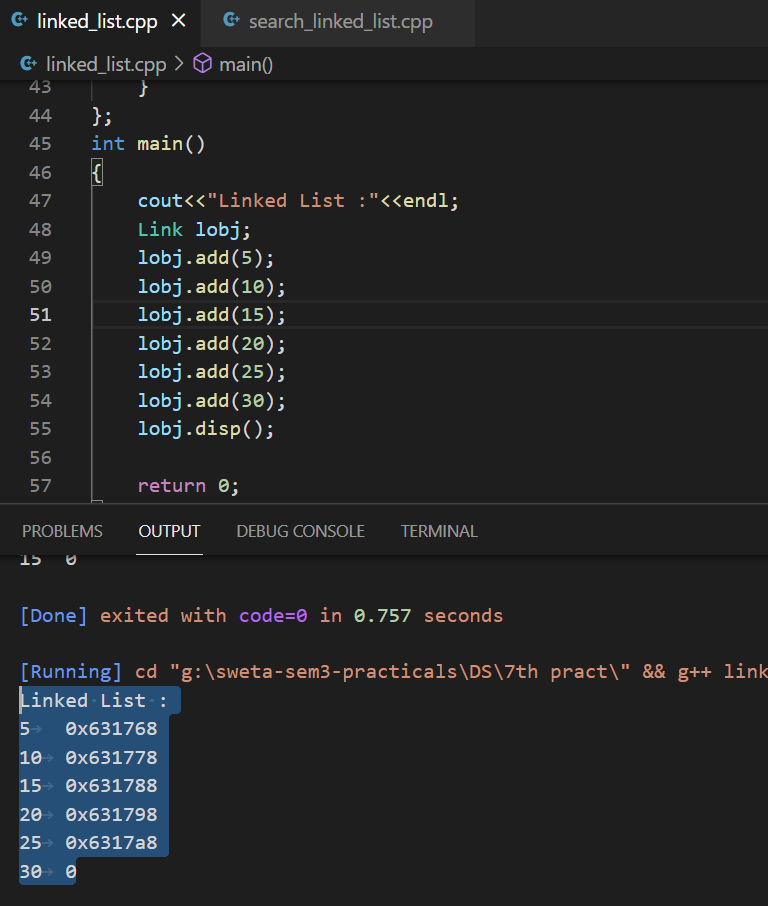
    lobj.add(30);

    lobj.disp();

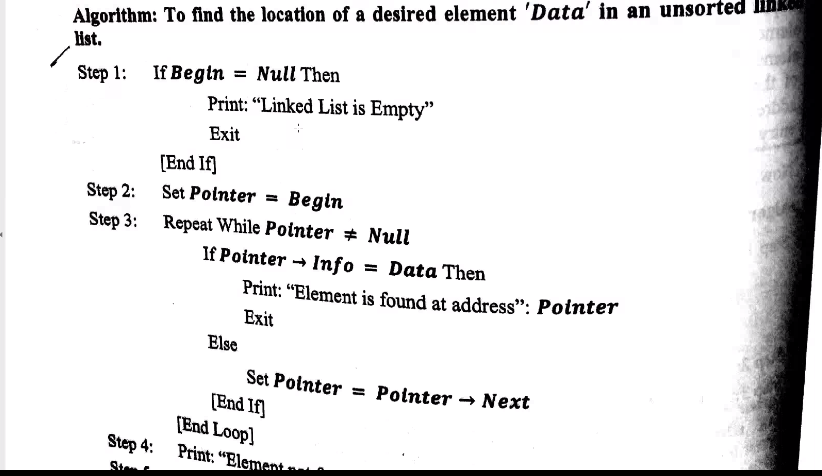
    return 0;

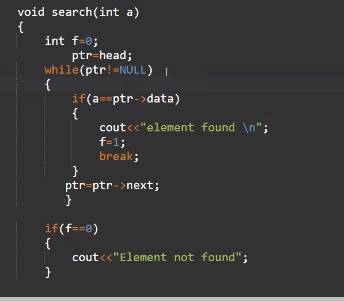
}

Output



2)





Code

#include <iostream>

using namespace std;

struct node

{

    int data;

    node \*next;

};

class Link

{

    node \*head, \*tail , \*ptr;

    public:

    Link()

    {

        head=NULL;

        tail=NULL;

        ptr=NULL;

    }

    void add(int n)

    {

        node \*temp= new node;

        temp->data= n;

        temp->next= NULL;

        if(head==NULL)

        {

            head=temp;

            tail=temp;

            ptr=temp;

        }

        else

        {

            tail->next= temp;

            tail=tail->next;

        }

    }

    void search(int a)

    {

        int f=0;

        while(ptr!=NULL)

        {

            if(a==ptr->data)

            {

                cout<<"Element found at \t"<<ptr->next<<endl;

                f=1;

                break;

            }

            ptr=ptr->next;

        }

        if(f==0)

        {

            cout<<"Element not found";

        }

    }

    void disp()

    {

        while (head!=NULL)

        {

            cout<<head->data<< "\t"<<head->next<< endl;

            head=head->next;

        }

    }

};

int main()

{

    cout<<"Linked List :"<<endl;

    Link lobj;

    lobj.add(5);

    lobj.add(10);

    lobj.add(15);

    lobj.add(20);

    lobj.add(25);

    lobj.add(30);

    lobj.disp();

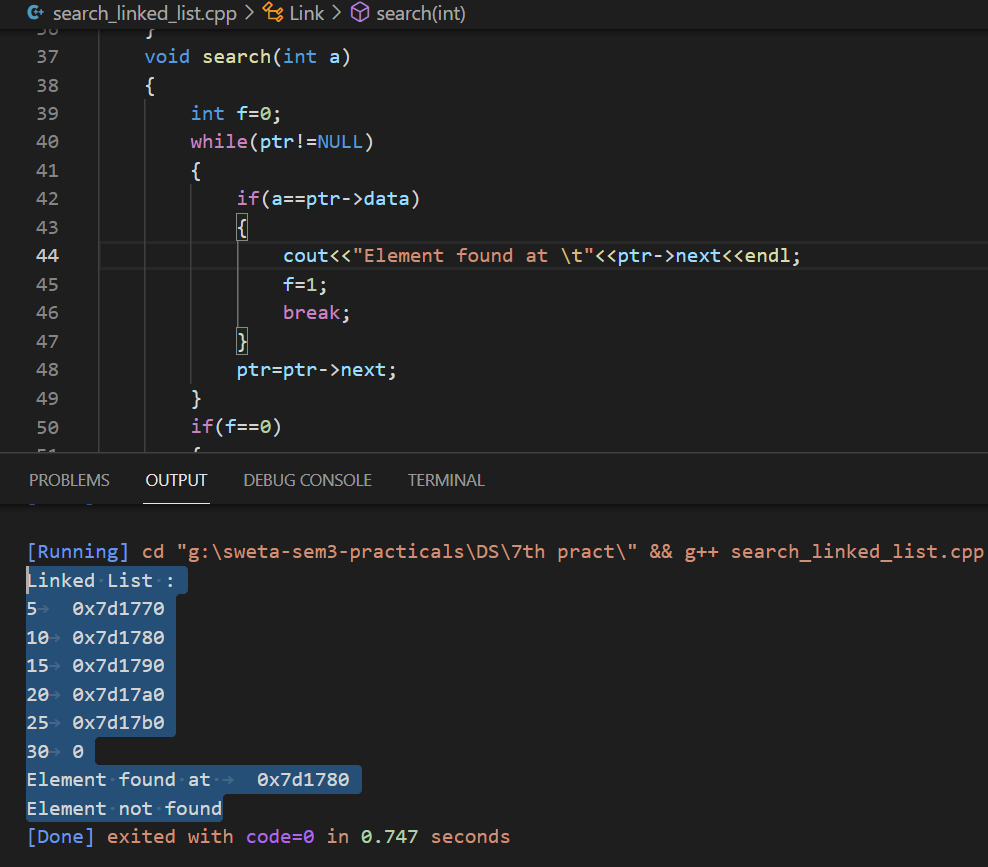
    lobj.search(10);

    lobj.search(35);

    return 0;

}

Output



1. Create a class Node which has two attributes: data and next. Next is a pointer to the next node in the list.
2. Create another class which has two attributes: head and tail.
3. addNode() will add a new node to the list:
   1. Create a new node.
   2. It first checks, whether the head is equal to null which means the list is empty.
   3. If the list is empty, both head and tail will point to the newly added node.
   4. If the list is not empty, the new node will be added to end of the list such that tail's next will point to the newly added node. This new node will become the new tail of the list.
4. reverse() will reverse the order of the nodes present in the list.
   1. This method checks whether node next to current is null which implies that, current is pointing to tail, then it will print the data of the tail node.
   2. Recursively call reverse() by considering node next to current node and prints out all the nodes in reverse order starting from the tail.
5. display() will display the nodes present in the list:
   1. Define a node current which will initially point to the head of the list.
   2. Traverse through the list till current points to null.
   3. Display each node by making current to point to node next to it in each iteration.