Exercise 1:Write a c++ program to sort the elements of unordered link list?

template <class T>

void unorderednextlist<T>::insert(T num) {

    node<T> \*q,\*t;

    if( head == NULL ) {// insert into empty list

        head = new node<T>;

        head->data = num;

        head->next = NULL;

    }

    else { //append

        q = head;

        while( q->next != NULL ){

            q = q->next;

            t = new node<T>;

            t->data = num;

            q->next= t;

            t->next = NULL;

        }

    }

}

template <class T>

void unorderednextlist<T>::add\_as\_first(T num) {// insert in the beginning

    node <T>\*q;

    q = new node<T>;

    q->data = num;

    q->next = head;

    head = q;

}

template <class T>

void unorderednextlist<T>::addafter( T c, T num) { // insert in the middle

    node <T>\*q,\*t;

    int i;

    for(i=1,q=head;i<c;i++) {

        q = q->next;

        if( q == NULL ) {

            cout<<"\nThere are less than "<<c<<" elements.";

            return ;

        }

    }

    t = new node<T>;

    t->data = num;

    t->next = q->next;

    q->next = t;

}

Exercise 2:Write a c++ program to print the elements of ordered link list in reverse order?

**void** printReverse(Node\* head)

{

    stack<**int**> st;

    Node \*curr = head;

**while**(curr!=NULL)

    {

      st.push(curr->data);

      curr = curr->next;

    }

**while**(st.empty()==**false**)

    {

      cout << st.top()<<" -> ";

      st.pop();

    }

}

Or

**void** reverse()

    {

        // Initialize current, previous and next pointers

        Node\* current = head;

        Node \*prev = NULL, \*next = NULL;

**while** (current != NULL) {

            // Store next

            next = current->next;

            // Reverse current node's pointer

            current->next = prev;

            // Move pointers one position ahead.

            prev = current;

            current = next;

        }

        head = prev;

    }