Insertion Sort:

```
#include <bits/stdc++.>
using namespace std;

class Node {
    public:
        int val;
        Node* next;
        Node(int x)
        {
            val = x;
             next = NULL;
        }
};

class LinkedlistIS {

public:
     Node* head;
     Node* sorted;

     void push(int val)
     {
          /* allocate node */
          Node* newnode = new Node(val);
          /* link the old list off the new node */
          newnode->next = head;
          /* move the head to point to the new node */
          head = newnode;
     }

     // function to sort a singly linked list using insertion
     // sort
     void insertionSort(Node* headref)
     {
          // Initialize sorted linked list
          sorted = NULL;
          Node* current = headref;
          // Traverse the given linked list and insert every
          // node to sorted
          while (current != NULL) {
               // Store next for next iteration
               Node* next = current->next;
               // insert current in sorted linked list
```

```
                sortedInsert(current);
                // Update current
                current = next;
            }
            // Update head_ref to point to sorted linked list
            head = sorted;
        }

        /*
        * function to insert a new_node in a list. Note that
        * this function expects a pointer to head_ref as this
        * can modify the head of the input linked list
        * (similar to push())
        */
        void sortedInsert(Node* newnode)
        {
            /* Special case for the head end */
            if (sorted == NULL || sorted->val >= newnode->val) {
                newnode->next = sorted;
                sorted = newnode;
            }
            else {
                Node* current = sorted;
                /* Locate the node before the point of insertion
                */
                while (current->next != NULL
                    && current->next->val < newnode->val) {
                    current = current->next;
                }
                newnode->next = current->next;
                current->next = newnode;
            }
        }
        /* Function to print linked list */
        void printlist(Node* head)
        {
            while (head != NULL) {
                cout << head->val << " ";
                head = head->next;
            }
        }
};
// Driver program to test above functions
int main()
{
```

```
    LinkedlistIS list;
    list.head = NULL;
    list.push(5);
    list.push(20);
    list.push(4);
    list.push(3);
    list.push(30);
    cout << "Linked List before sorting" << endl;
    list.printlist(list.head);
    cout << endl;
    list.insertionSort(list.head);
    cout << "Linked List After sorting" << endl;
    list.printlist(list.head);
}
```

Bubble Sort in singly linked list:

```
bubbleSort()
{
node *ptr,*ptr1;
ptr1=NULL;
int swap;
if(header==NULL)
cout<<"list is empty"<<endl;
else
{

do {
ptr=header;
swap=0;
while(ptr->link!=ptr1)
{
if(ptr->data>ptr->link->data)
{
int temp;
temp=ptr->data;
ptr->data=ptr->link->data;
ptr->link->data=temp;
swap=1;
}
ptr=ptr->link;
}
ptr1=ptr;
}while(swap);
```

```
}
}
```

//Selection sort

```
 Void selectionLin
                    {
                       node* key;
                       key = start;


                       while(key != NULL)
                       {
                            temp = key->next;
                            while(temp != NULL)
                            {
                                if(key->data >
                    temp->data)
                                {
                                    swapNode(key,
                    temp);
                                }
                                temp = temp->next;
                            }
                            key = key->next;
                       }
                    }


                    void swapNode(node *x, node *y)
                    {
                       int temp = x->data;
```

```
    x->data = y->data;
    y->data = temp;
}
```