1) Create an empty sorted (or result) list

2) Traverse the given list, do following for every node.

......a) Insert current node in sorted way in sorted or result list.

3) Change head of given linked list to head of sorted (or result) list.

/\* C program for insertion sort on a linked list \*/

#include<stdio.h>

#include<stdlib.h>

/\* Link list node \*/

struct Node

{

    int data;

    struct Node\* next;

};

// Function to insert a given node in a sorted linked list

void sortedInsert(struct Node\*\*, struct Node\*);

// function to sort a singly linked list using insertion sort

void insertionSort(struct Node \*\*head\_ref)

{

    // Initialize sorted linked list

    struct Node \*sorted = NULL;

    // Traverse the given linked list and insert every

    // node to sorted

    struct Node \*current = \*head\_ref;

    while (current != NULL)

    {

        // Store next for next iteration

        struct Node \*next = current->next;

        // insert current in sorted linked list

        sortedInsert(&sorted, current);

        // Update current

        current = next;

    }

    // Update head\_ref to point to sorted linked list

    \*head\_ref = 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(struct Node\*\* head\_ref, struct Node\* new\_node)

{

    struct Node\* current;

    /\* Special case for the head end \*/

    if (\*head\_ref == NULL || (\*head\_ref)->data >= new\_node->data)

    {

        new\_node->next = \*head\_ref;

        \*head\_ref = new\_node;

    }

    else

    {

        /\* Locate the node before the point of insertion \*/

        current = \*head\_ref;

        while (current->next!=NULL &&

               current->next->data < new\_node->data)

        {

            current = current->next;

        }

        new\_node->next = current->next;

        current->next = new\_node;

    }

}

/\* BELOW FUNCTIONS ARE JUST UTILITY TO TEST sortedInsert \*/

/\* Function to print linked list \*/

void printList(struct Node \*head)

{

    struct Node \*temp = head;

    while(temp != NULL)

    {

        printf("%d  ", temp->data);

        temp = temp->next;

    }

}

/\* A utility function to insert a node at the beginning of linked list \*/

void push(struct Node\*\* head\_ref, int new\_data)

{

    /\* allocate node \*/

    struct Node\* new\_node = new Node;

    /\* put in the data  \*/

    new\_node->data  = new\_data;

    /\* link the old list off the new node \*/

    new\_node->next = (\*head\_ref);

    /\* move the head to point to the new node \*/

    (\*head\_ref)    = new\_node;

}

// Driver program to test above functions

int main()

{

    struct Node \*a = NULL;

    push(&a, 5);

    push(&a, 20);

    push(&a, 4);

    push(&a, 3);

    push(&a, 30);

    printf("Linked List before sorting \n");

    printList(a);

    insertionSort(&a);

    printf("\nLinked List after sorting \n");

    printList(a);

    return 0;

}