Insertion Sort

Eg: Insert 5 into a list

The following cases arise. The node **underlined** is 1st node with value > 5.

Case 1: List is empty

Ø (Sol: 5 → Ø)

Case 2: The new node might be the new head (5 < head)

14 → 17 → Ø (Sol: 5 → 14 → 17 → Ø)

Case 3: Correct internal location is reached (*ie,* 5 is not the tail)

3 → **7** → Ø (Sol: 3 → 5 → 7 → Ø)

3 → 4 → **7** → Ø (Sol: 3 → 4 → 5 → 7 → Ø)

3 → 4 → **7** → 8 → Ø (Sol: 3 → 4 → 5 → 7 → 8 → Ø)

Case 4: Cannot insert before any node, hence *must* be the new tail.

1 → 2 → **3** → Ø (Sol: 1 → 2 → 3 → 5 → Ø)

```
struct node
      int data;
      node *next;

void isort(node *&head, int i)
      node *xp; // pointer to previous node
      node *xc; // pointer to current node
      node *xi; // pointer to new node, ie., the node to be inserted

      xc=head;
      xi=new node;
      xi->data=i;
      xi->next=NULL;

      // keep moving to the next node till any of the 3 conditions fail
      while
            1. current node is not NULL ( xc != NULL )
            2. next node is not NULL ( xc->next != NULL )
            3. i <= data in current NULL ( i <= xc->data)
                  move to the next node ( xp = xc; xc = xc->next;)
      end while

      // insert the new node (xi) identifying the appropriate case
      Case 1: the list is empty ( head == NULL )
            xi becomes the head
      Case 2: xi->data is smaller than head->data
            xi becomes the new head
      Case 3: the current data is larger than xi->data
            xi comes between previous node (xp) and current node (xc)
      Case 4: none of above (ie., new node could not be inserted before any node)
            xi comes immediately after xc and is the new tail
```