Implementing Singly Linked List in C++ Using struct

Example 1: Declarations to create a node and playing with pointers.

```
#include <iostream>
using namespace std;

struct Node
{

   int data;
   Node *next;

};

int main()
{
   Node *head=NULL;
   Node  *t;
   Node  *n;

   n = new Node;
   n->data = 10;
   n->next = NULL;

   head=n;
   t=n;

   n = new Node;
   n->data = 20;
   n->next = NULL;

   cout<<head->data<<endl;
   cout<<head<<endl;
   cout<<t->data<<endl;
   cout<<t<<endl;

   cout<<n->data<<endl;
   cout<<n<<endl;

   head->next=n;

   cout<<n<<endl;

   t=t->next;
   cout<<t<<endl;
```

}

Example 2: Adding or Appending a Node to the List

```
#include <iostream>
using namespace std;
//Declaration to create a node
struct Node
        {
                int data;
                Node *next;
        };
// define the head pointer and make it points to null
Node *head = NULL;

// Adding or Appending a Node to the List
void appendNode(int value)
{
   Node *n, *curr;
   // Allocate a new node & store data
   n = new Node();
   n->data = value;
   n->next = NULL;
   // If there are no nodes in the list
   // make n the first node
   if (head==NULL)
                head = n;
```

```
        else    // Otherwise, insert newNode at end
        {
           // Initialize curr to head of list
           curr = head;
           // Find the last node in the list
           while (curr->next!= NULL)
           {
              curr = curr->next;
           }

           // Insert newNode as the last node
           curr->next = n;
         }

}
int main()
{
   appendNode(2);
   appendNode(14);
   appendNode(26);
   return 0;
}
```

Example 3: Adding Nodes to the List and then printing them from the beginning until the end.

```
#include <iostream>
using namespace std;
//Declaration to create a node
struct Node
	{
		int data;
		Node *next;
	};
// define the head pointer and make it points to null
Node *head = NULL;
// Adding or Appending a Node to the List
void appendNode(int value)
{
   Node *n, *curr;
   // Allocate a new node & store data
   n = new Node();
   n->data = value;
   n->next = NULL;
   // If there are no nodes in the list
   // make n the first node
   if (head==NULL)
		head = n;
```

```
        else    // Otherwise, insert newNode at end
        {
           // Initialize curr to head of list
           curr = head;
           // Find the last node in the list
           while (curr->next!= NULL)
           {
              curr = curr->next;
           }

           // Insert newNode as the last node
           curr->next = n;
         }
}

// displaying the list from the beginning
void displayList(void)
{
   Node *curr;
   curr = head;
   while (curr!=NULL)
      {
         cout << curr->data << endl;
         curr = curr->next;
```

```
        }

}

// main function
int main()
{
    appendNode(2);
    appendNode(14);
    appendNode(26);
    displayList();
    return 0;
}
```

Example 4: Insertion at the end of the list.

```
#include <iostream>
using namespace std;
//Declaration to create a node
struct Node
        {
                int data;
                Node *next;
        };
// define the head pointer and make it points to null
Node *head = NULL;
// Adding or Appending a Node to the List
void appendNode(int value)
{
   Node *n, *curr;
   // Allocate a new node & store data
   n = new Node();
   n->data = value;
   n->next = NULL;
   // If there are no nodes in the list
   // make n the first node
   if (head==NULL)
                head = n;
        else    // Otherwise, insert newNode at end
```

```
        {
            // Initialize curr to head of list
            curr = head;
            // Find the last node in the list
            while (curr->next!= NULL)
            {
                curr = curr->next;
            }

            // Insert newNode as the last node
            curr->next = n;
        }
}

// displaying the list from the beginning
void displayList(void)
{
    Node *curr;
    curr = head;
    while (curr!=NULL)
      {
         cout << curr->data << endl;
         curr = curr->next;
      }
```

```
}

// main function
int main()
{
    appendNode(2);
    appendNode(14);
    appendNode(26);
    displayList();
    return 0;
}
```

Example 5: Insertion at the top of the list.

```
#include <iostream>
using namespace std;
//Declaration to create a node
struct Node
        {
                int data;
                Node *next;
        };
// define the head pointer and make it points to null
Node *head = NULL;
// Adding or Appending a Node to the List
void InsertNode(int value)
{
   Node *n, *curr;
   // Allocate a new node & store data
   n = new Node();
   n->data = value;
   n->next = NULL;
   // If there are no nodes in the list
   // make n the first node
   if (head==NULL)
                head = n;
        else    // Otherwise, insert newNode at end
```

```
        {
           // Initialize curr to head of list
           curr = head;
                 // Insert n as the first node
           head = n;
           // connect the first node to the rest of the nodes
           head->next = curr;
         }
}

// displaying the list from the beginning
void displayList(void)
{
   Node *curr;
   curr = head;
   while (curr!=NULL)
      {
         cout << curr->data << endl;
         curr = curr->next;
      }

}

// main function
```

```
int main()
{
    InsertNode(2);
    InsertNode(14);
    InsertNode(26);
    displayList();
    return 0;
}
```

Example 6: Insertion at the middle of the list

```
#include <iostream>
using namespace std;
//Declaration to create a node
struct Node
        {
                int data;
                Node *next;
        };
// define the head pointer and make it points to null
Node *head = NULL;
// Adding or Appending a Node to the List
void InsertNode(int value)
{
   Node *n, *curr, *previous=NULL;
   // Allocate a new node & store data
   n = new Node();
   n->data = value;
   n->next = NULL;
   // If there are no nodes in the list
   // make n the first node
   if (head==NULL)
                head = n;
```

```
        else    // Otherwise, insert n at end
        {
           curr = head;
           // Skip all nodes whose value member is less than value.
           while (curr != NULL && curr->data < value)
           {
              previous = curr;
              curr = curr->next;
           }

// If n is to be the 1st in the list, insert it before all other nodes
       if (previous == NULL)
           {
              head = n;
              n->next = curr;
           }
           else
           {

              previous->next = n;
              n->next = curr;
           }

           }
```

```
}

// displaying the list from the beginning
void displayList(void)
{
   Node *curr;
   curr = head;
   while (curr!=NULL)
      {
         cout << curr->data << endl;
         curr = curr->next;
      }

}

// main function
int main()
{
   InsertNode(2);
   InsertNode(14);
   InsertNode(26);
```

```
    InsertNode(30);
    InsertNode(18);
    InsertNode(1);
    displayList();
    return 0;
}
```

Example 7: Delete a node with a specific value from the list

```
#include <iostream>
using namespace std;
//Declaration to create a node
struct Node
        {
                int data;
                Node *next;
        };
// define the head pointer and make it points to null
Node *head = NULL;
// Adding or Appending a Node to the List
void InsertNode(int value)
{
   Node *n, *curr, *previous=NULL;
   // Allocate a new node & store data
   n = new Node();
   n->data = value;
   n->next = NULL;
   // If there are no nodes in the list
   // make n the first node
   if (head==NULL)
                head = n;
        else    // Otherwise, insert n at end
```

```
        {
            curr = head;
            // Skip all nodes whose value member is less than value.
            while (curr != NULL && curr->data < value)
            {
                previous = curr;
                curr = curr->next;
            }

// If n is to be the 1st in the list, insert it before all other nodes
        if (previous == NULL)
            {
                head = n;
                n->next = curr;
            }
            else
            {

                previous->next = n;
                n->next = curr;
            }

            }
```

```
}

// displaying the list from the beginning
void displayList(void)
{
   Node *curr;
   curr = head;
   while (curr!=NULL)
      {
         cout << curr->data << endl;
         curr = curr->next;
      }

}

void DeleteNode(int value)
{
   Node *curr, *previous = NULL;
   // If the list is empty, do nothing.
   if (!head)
         return;
   // Determine if the first node is the one.
```

```
if (head->data == value)
{
   curr = head->next;
   delete head;
   head = curr;
}


      else
      {
         // Initialize nodePtr to head of list
         curr = head;
         // Skip all nodes whose value member is
         // not equal to num.
         while (curr != NULL && curr->data != value)
         {
            previous = curr;
            curr = curr->next;
         }
         // Link the previous node to the node after
         // nodePtr, then delete nodePtr.
         previous->next = curr->next;
         delete curr;
      }
```

```
}



// main function
int main()
{
    InsertNode(2);
    InsertNode(14);
    InsertNode(26);
    InsertNode(30);
    InsertNode(18);
    InsertNode(1);
    cout<<"the list after insertion is:\n";
    displayList();

    DeleteNode(18);
    cout<<"the list after deleting 18 is:\n";
    displayList();

    return 0;
}
```

Example 8: Delete the first node from the list

```
#include <iostream>
using namespace std;
//Declaration to create a node
struct Node
        {
                int data;
                Node *next;
        };
// define the head pointer and make it points to null
Node *head = NULL;
// Adding or Appending a Node to the List
void InsertNode(int value)
{
   Node *n, *curr, *previous=NULL;
   // Allocate a new node & store data
   n = new Node();
   n->data = value;
   n->next = NULL;
   // If there are no nodes in the list
   // make n the first node
   if (head==NULL)
                head = n;
        else    // Otherwise, insert n at end
```

```
        {
            curr = head;
            // Skip all nodes whose value member is less than value.
            while (curr != NULL && curr->data < value)
            {
                previous = curr;
                curr = curr->next;
            }

// If n is to be the 1st in the list, insert it before all other nodes
        if (previous == NULL)
            {
                head = n;
                n->next = curr;
            }
            else
            {

                previous->next = n;
                n->next = curr;
            }

            }
```

```
}

// displaying the list from the beginning
void displayList(void)
{
   Node *curr;
   curr = head;
   while (curr!=NULL)
      {
         cout << curr->data << endl;
         curr = curr->next;
      }

}

void DeleteFirstNode()
{
   Node *curr;
   // If the list is empty, do nothing.
   if (!head)
         return;
   else
```

```
        {
            curr = head->next;
            delete head;
            head = curr;
        }
}
// main function
int main()
{
    InsertNode(2);
    InsertNode(14);
    InsertNode(26);
    InsertNode(30);
    InsertNode(18);
    InsertNode(1);
    cout<<"the list after insertion is:\n";
    displayList();
    DeleteFirstNode();
    cout<<"the list after deleting the first node is:\n";
    displayList();

    return 0;
}
```

Example 9: Delete the last node from the list

```
#include <iostream>
using namespace std;
//Declaration to create a node
struct Node
        {
                int data;
                Node *next;
        };
// define the head pointer and make it points to null
Node *head = NULL;
// Adding or Appending a Node to the List
void InsertNode(int value)
{
   Node *n, *curr, *previous=NULL;
   // Allocate a new node & store data
   n = new Node();
   n->data = value;
   n->next = NULL;
   // If there are no nodes in the list
   // make n the first node
   if (head==NULL)
                head = n;
        else    // Otherwise, insert n at end
```

```
        {
            curr = head;
            // Skip all nodes whose value member is less than value.
            while (curr != NULL && curr->data < value)
            {
                previous = curr;
                curr = curr->next;
            }

// If n is to be the 1st in the list, insert it before all other nodes
        if (previous == NULL)
            {
                head = n;
                n->next = curr;
            }
            else
            {

                previous->next = n;
                n->next = curr;
            }

            }
```

```
}

// displaying the list from the beginning
void displayList(void)
{
   Node *curr;
   curr = head;
   while (curr!=NULL)
      {
         cout << curr->data << endl;
         curr = curr->next;
      }

}

void DeleteLastNode()
{
   Node *curr, *previous;
   // If the list is empty, do nothing.
   if (!head)
         return;
   else
```

```
        {
            curr = head;
           while (curr->next != NULL)
               {
                   previous = curr;
                   curr = curr->next;
               }

        }
        delete curr;
        previous->next = NULL;
}

// main function
int main()
{
    InsertNode(2);
    InsertNode(14);
    InsertNode(26);
    InsertNode(30);
    InsertNode(18);
    InsertNode(1);
    cout<<"the list after insertion is:\n";
    displayList();
```

```
    DeleteLastNode();
    cout<<"the list after deleting the last node is:\n";
    displayList();

    return 0;
}
```