U10P32002 Spring 2021

ABID ALI, 2019380141

Abiduu354@gmail.com, (04/0/2021)

**◎ Problem Description**

In this assignment, we are going to write a C++ program files(dse\_assign01.cpp”and text file with code also included), which includes implementations of the methods of merge two sorted linked lists. These program needs to use software tool for implementation of the list of assignment.We used CodeBlocks (IDE). Using this program, a user can marge two short list together. The user can then view all actions and processes by this program. The following screen shot depicts the program during execution.

**◎ Goals**

This assignment is designed to help understanding and mastering the use of C++ header files, basic C++ syntax, and stream-based I/O. By successfully completing this assignment we could master the following outcomes:

* **Understanding the of basic C++ syntax.**
* **By using basic C++ we learn implementation of control structures.**
* **Learn to use different function and implement in program which is provided by teacher.**
* **Learning the use of a singly linked list with dummy node implementation.**

**Linked List** is a linear collection of data elements whose order is not given by their physical placement in memory. Instead, each element [points](https://en.wikipedia.org/wiki/Pointer_(computer_programming)) to the next. It is a [data structure](https://en.wikipedia.org/wiki/Data_structure) consisting of a collection of [nodes](https://en.wikipedia.org/wiki/Node_(computer_science)) which together represent a [sequence](https://en.wikipedia.org/wiki/Sequence).

The concept of Linked List those which has high importance are discussed below:

**Link-**We can consider Link where we can store data.InLinked List we can create a sequence of data structures, which are connected together via links. In Linked List we can see sequence of links which contains items. Each link contains a connection to another link.

**Node-** Each element in a linked list is stored in the form of a node. Node: A node is a collection of two sub-elements or parts. A data part that stores the element and a next part that stores the link to the next node.

**◎ Requirement Analysis**

Specify the input (e.g. form, range), the output (e.g. form), the function or purpose of your code project, and sample data (including correct/incorrect inputs and corresponding outputs).

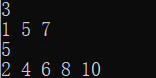
**1.Input:**

* First line – the length of the first sorted linked list, which is an integer ;
* Second line -- the  numbers belonging to the first sorted linked list which are separated by spaces;
* Third line -- the length of the second sorted linked list, which is an integer ;
* Forth line -- the  numbers belonging to the second sorted linked list which are separated by spaces.

**2.Output:**

The merged sorted linked list. Print each number on a single line.

Sample input:



Sample output:



**◎ General Design**

**1. Data structure**

The data structure for singly linked list.

**2. Basic operations needed to complete:**

* **Traversal**: To traverse all the nodes one after another.
* **Insertion**: To add a node at the given position.
* **Deletion**: To delete a node.
* **Merging:**To merge two linked lists into one.

We will see the various implementation of these operations on a singly linked list.

#### **Linked List node Insertion**

There can be three cases that will occur when we are inserting a node in a linked list.

* Insertion at the beginning
* Insertion at the end. (Append)
* Insertion after a given node

#### **Linked List node Deletion**

To delete a node from a linked list, we need to do these steps

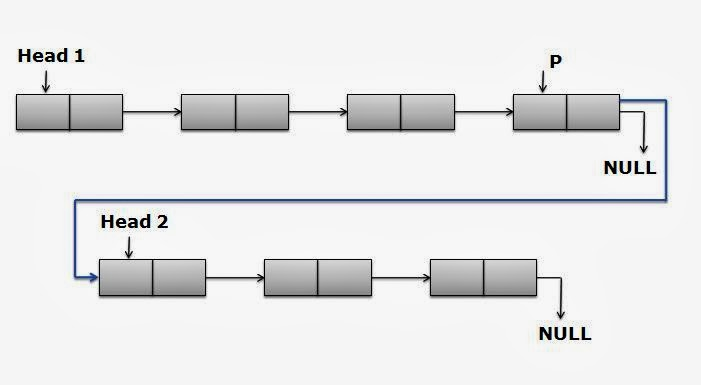
* Find the previous node of the node to be deleted.
* Change the next pointer of the previous node
* Free the memory of the deleted node.

In the deletion, there is a special case in which the first node is deleted. In this, we need to update the head of the linked list.

#### **Linked List Merging**

To merge a linked list, we need to do these steps

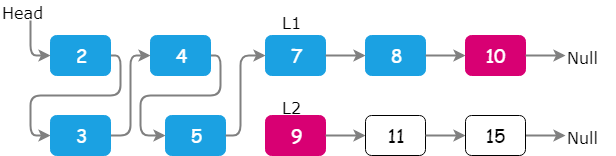
* We will have a sorted elements.
* We need to connect the sorted element in order.



**Picture: Basic operations of singly linked list**

**3.Hierarchy of the program:**

The structure chart of the program is displayed in Figure

****

**Figure:** Structure chart of the program

**◎ Implementation**

The linked list is made up of different kind of nodes which are connected in sequence. Every node can be divided into two parts: one of them is storing data and remaining part is connected to a different node. In this program, we will use structures and pointers. Each structure represents a node having some data and the address of the next node and creates the link between two nodes.

In the merge.cpp(C++) two list classes are there.One is Private and Public.We didn’t use Private in this problem because it automatically acts private class when the public class is defined later.I have learned about Private and Public in Object Oriented Programming then I wanted to implement that idea in this code.

Public class can be used by everyone based upon their need.During creating a class and object we use this Public term a lot.

Public class contains three functions:

build () and print\_list() are no return type whereas merge\_list(list, list) is a return type.

**Insertion: an element at the beginning of the list:**

#include <iostream>

using namespace std;

typedef struct Node

{

public:

int info;

struct Node \*link;

}Node,\*PNode;

typedef struct LinkList

{

PNode head;

PNode tail;

int size;

}LinkList, \*PLinkList;

struct Node \*create(struct Node \*start);

struct Node \*insert\_list(struct Node \*start, int data);

struct Node \*insert(struct Node \*start, int data);

void print\_list(struct Node \*start);

**Deletion − Deletes an element at the beginning of the list:**

void merge\_list(struct Node \*p1, struct Node \*p2)

{

struct Node \*start3;

start3 = NULL;

while (p1 != NULL && p2 != NULL)

{

if (p1->info < p2->info)

{

start3 = insert(start3, p1->info);

p1 = p1->link;

}

else if (p2->info < p1->info)

{

start3 = insert(start3, p2->info);

p2 = p2->link;

}

else if (p1->info == p2->info)

{

start3 = insert(start3, p1->info);

p1 = p1->link;

p2 = p2->link;

}

}

while (p1 != NULL)

{

start3 = insert(start3, p1->info);

p1 = p1->link;

}

while (p2 != NULL)

{

start3 = insert(start3, p2->info);

p2 = p2->link;

}

print\_list(start3);

}

**Search − Searches an element using the given key:**

struct Node\*create(struct Node \*start)

{

int i, n, data;

cin>>n;

start = NULL;

for (i = 1; i <= n; i++)

{

cin>>data;

start = insert\_list(start, data);

}

return start;

}

struct Node \*

insert\_list(struct Node \*start, int data)

{

struct Node \*p, \*tmp;

tmp = (struct Node \*)malloc(sizeof(struct Node));

tmp->info = data;

if (start == NULL || data < start->info)

{

tmp->link = start;

start = tmp;

return start;

}

else

{

p = start;

while (p->link != NULL && p->link->info < data)

p = p->link;

tmp->link = p->link;

p->link = tmp;

}

return start;

}

**Delete − Deletes an element using the given key:**

struct Node\*insert(struct Node \*start, int data)

{

struct Node \*p, \*tmp;

tmp = (struct Node \*)malloc(sizeof(struct Node));

tmp->info = data;

if (start == NULL)

{

tmp->link = start;

start = tmp;

return start;

}

else

{

p = start;

while (p->link != NULL)

p = p->link;

tmp->link = p->link;

p->link = tmp;

}

return start;

}

**Final modifications:**

void print\_list(struct Node \*start)

{

struct Node \*p;

if (start == NULL)

{

cout<<"List is empty\n";

return;

}

p = start;

while (p != NULL)

{

printf("%d ", p->info);

p = p->link;

}

cout<<"\n";

}

**Display − Displays the complete list:**

int main()

{

struct Node \*start1 = NULL, \*start2 = NULL;

start1 = create(start1);

start2 = create(start2);

merge\_list(start1, start2);

return 0;

}

**◎ Test Description and Results**

**Test 1:**

**Input:**

**3**

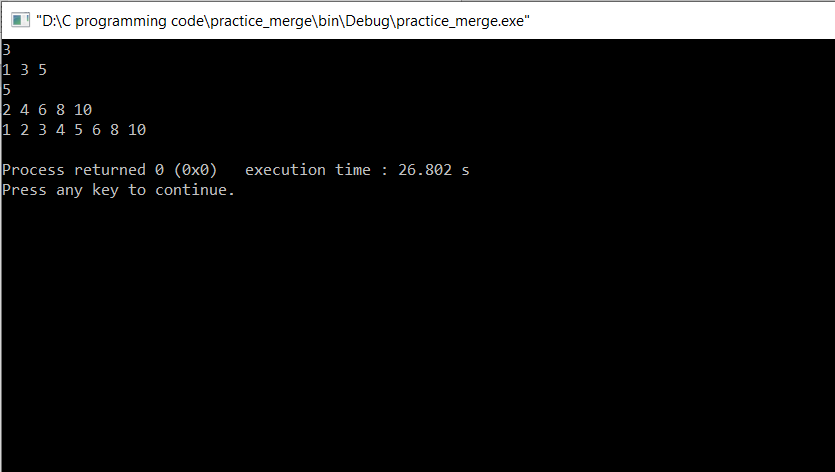
**1 5 7**

**5**

**2 4 6 8 10**

**Output:**

**1 2 4 5 6 7 8 10**

****

**Figure:Test 1**

**Test 2:**

**Input:**

**4**

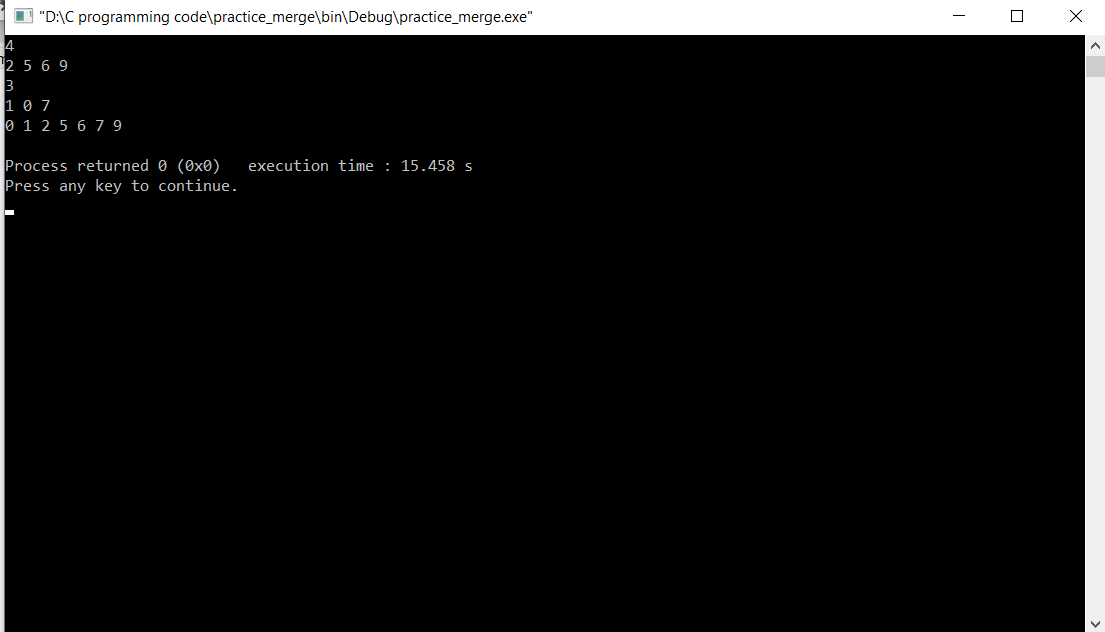
**2 5 6 9**

**3**

**0 1 7**

**Output:**

**0 1 2 5 6 7 9**

****

**Figure:Test 2**

**Bug:**At first,two linked list were not giving our desired result.Then,after using loop then ,I was able to link 2 sorted linked list and merge them together.One problem was I had to input values vertically then I started researching again and understood the problem and solve it.

**◎ Epilogue**

While doing this assignment I had three big bugs which had influence on program’s working and many troubles with compilation programs, because of syntax and different kinds of functions and using all those functions together.

At first,I draw the picture and watch some video about the topic and then I started to Implement the design in code.The hardest part was merged the link list perfectly.At the beginning I was having problem to lin the Node and different function together.After few trial and error ,I was able to make a working problem.Then,I go through my code code again to remove any unused code or function.At last,I was able to solve it correctly.

The most interesting thing in that assignment for me was to implement all those new idea together and create a program. I got more acquainted with different set of functions and Node.

**◎ Attachments**

dse-assignt01.cpp

codeAssignment\_1.txt

**◎ Acknowledgement**

I complete this assignment by myself by using online videos and different books discussing about Algorithm and Data structure. It was very useful and helpful for me to increase knowledge for solving complex problem.

**◎ Remarks and Grade (by the instructor)**

Instructor Signature:

Grading Date: