CET214 – Data Structures & Algorithms

Experiment # 09

Experiment Title

Linked List - I

Assessment of CLO(s): IV

Performed on 06-12-2024

Student Name			
Roll No.		Group	
Semester		Session	

Total (Max)	Criteria 1 (2.5)	Criteria 2 (2.5)	Criteria 3 (2.5)	Criteria 4 (2.5)	Total (10)
Marks Obtained					
Remarks (if any)					

Experiment evaluated by

Instructor's Name	Engr. Muhammad Asad Husain		
Date		Signature	

Department of Engineering Technology
(UIT University)

Course Code: CET214 Course Title: Data Structures & Algorithms Course Credits: 3+1 Session: Fall 2024

Rubric for assessment criteria to perform experiment number 9.

Level / Criteria	UNSATISFACTORY 1	COMPETENT 2	PROFICIENT 3	DISTINGUISHED 4
Capability of writing algorithm/ Procedure	None of the steps are implemented of an algorithm.	Few steps are implemented correctly of an algorithm.	Most of the steps are implemented correctly of an algorithm.	All the steps are implemented correctly of an algorithm.
Capability of writing Program	Programs not completed.	Completeness of code, consistent variable naming and unformatted.	Completeness of code, inconsistent variable naming and well formatted.	Completeness of code, consistent variable naming and well formatted.
Completion of target in Lab	25% target has been completed	50% target has been completed	75% target has been completed	100% target has been completed
Output	None of the outputs are correct.	Few outputs have been found correctly.	Some of the outputs are correct and well formatted.	Most of the outputs are correct and well formatted.

Practical Objective(s):

i. Creating a Linked List using structures and pointers.
ii. Traversing a Linked List.

Theory

Linked List

Linked list is a convenient way to store an unbounded array, that is create an array where one doesn't know in advance how large the array will be. The disadvantage of the linked list is that data can only be accessed sequentially and not in random order. To read the millionth element of a linked list, you must read the 999,999 elements that precede it. An array allocates memory for all its elements lumped together as one block of memory. In contrast, a linked list allocates space for each element separately in its own block of memory. A linked list element is called a “node". The list gets its overall structure by using pointers to connect all its nodes together like the links in a chain.

A linked list usually consists of a root node, allocated on the stack (an ordinary C++ variable) and one or more nodes allocated on the heap (by calling new). Each node is a structure object.

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

The structure node typically has two fields:
- the data field
- the next (or link) field

The data filed holds the information stored in the node. The next field saves address of the next node in the linked list. The end of the linked list is indicated by setting the next field to NULL, which means that there is no node next to this node. Nodes are allocated on the heap memory using new operator in C++.

Figure 1 shows how a linked list looks like. The beginning of the linked list is stored in a "start" or "root" pointer which points to the first node. The first node contains a pointer to the second node. The second node contains a pointer to the third node, and so on. The last node in the list has its next field set to NULL to mark the end of the list. Any node in the list can be accessed by starting at the start and following the next pointers. Operations towards the front of the list are fast while operations which access node farther down the list take more time.

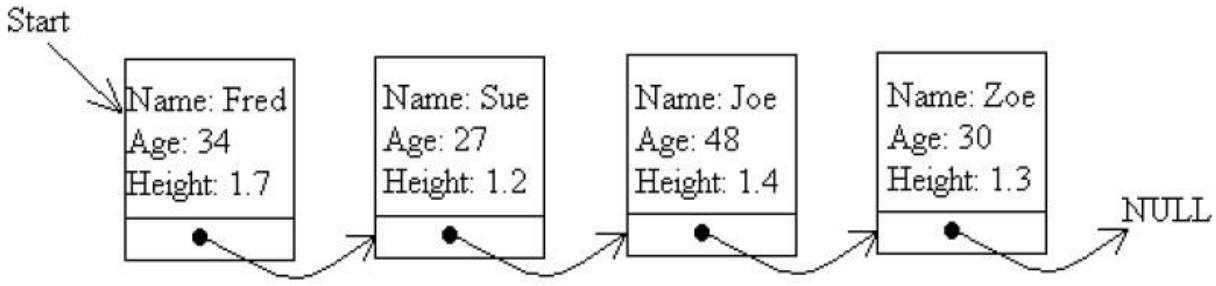

Figure 1: Linked List Representation

Program 1:

```
#include <iostream>
#include <string>
using namespace std;
struct node
{
     int data;
     struct node *next;
};
int main()
{
      struct node *root_node;

      root_node->data=50;
      root_node->next=NULL;
      cout<<"Data at Root Node="<<root_node->data<<endl;
      cout<<"Address of Root Node="<<root_node<<endl;
      cout<<"Next field of Root Node points to address="<<root_node->next;
      cout<<endl;

      char choice;
      int data;
      cout<<"Create another node?(Y/N)";
      cin>>choice;

      while(choice=='Y')
      {
          cout<<"Enter data to be saved in this node:";
          cin>>data;
          struct node *new_node, *s;
          s = root_node;
          new_node=new struct node;

      while (s->next != NULL)
      {
          s = s->next;
      }
```

```
        new_node->next = NULL; s->next = new_node;
        new_node->data=n;

        cout<<"Data at New Node:"<<new_node->data<<endl;
        cout<<"Address of New Node="<<new_node<<endl;
        cout<<"Next field of New Node is pointing to address="<<new_node-
        >next<<endl;
        cout<<"Next field of previous Node is pointing to address="<<s-
        >next<<endl;
        cout<<endl;
        cout<<"Create another node?(Y/N)";
        cin>>choice;
        }

        //Traversing the Linked List
        struct node *counter=root_node;
        if(counter!=NULL)
        {
             while(counter->next!=NULL)
             {
                  cout<<counter->data<<" is at "<<counter<<endl;
                  counter=counter->next;
             }
         }

        cout<<counter->data<<" is at "<<counter<<endl;
        return 0;
}
```

Do It Yourself:

1. (a) Add these fields in the structure node created in Program 1.
   - Vehicle type (e.g. car, van, bus, truck etc.)
   - Vehicle company
   - Vehicle color

   (b) Print all the vehicles of a given type.