Lab 02 Singly Linked List Implementation

Objectives:

- Learn to create a singly linked list.
- Learn to insert nodes at the end/start of a singly linked list.
- Learn to implement other simple linked list operations such as searching the list for key, find the length of the linked list etc.

Pre-Lab

Reading Task 1:

Working with Linked Lists in C:

A linked list, in simple terms, is a linear collection of data elements. These data elements are called nodes. Linked list is a data structure which in turn can be used to implement other data Learning Objective A linked list is a collection of data elements called nodes in which the linear representation is given by links from one node to the next node. In this chapter, we are going to discuss different types of linked lists and the operations that can be performed on these lists. A linked list can be perceived as a train or a sequence of nodes in which each node contains one or more data fields and a pointer to the next node.

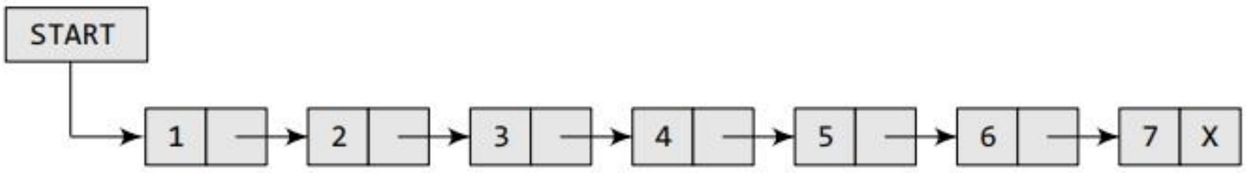

Figure 1: Visual depiction of a singly linked list

In Figure 1, we can see a linked list in which every node contains two parts, an integer and a pointer to the next node. The left part of the node which contains data may include a simple data type, an array, or a structure. The right part of the node contains a pointer to the next node (or address of the next node in sequence). The last node will have no next node connected to it, so it will store a special value called NULL . In Fig. 6.1, the NULL pointer is represented by X . While programming, we usually define NULL as –1. Hence, a NULL pointer denotes the end of the list. Since in a linked list, every node contains a pointer to another node which is of the same type, it is also called a self-referential data type.

Linked lists contain a pointer variable START that stores the address of the first node in the list. We can traverse the entire list using START which contains the address of the first node; the next part of the first node in turn stores the address of its succeeding node. Using this technique, the individual nodes of the list will form a chain of nodes. If START = NULL , then the linked list is empty and contains no nodes.

For further discussion on Linked Lists and their implementation read Chapter 6 of the book “Data Structures using C”, Oxford University Press, 2nd Edition by Reema Thareja.

In-Lab Tasks:

You are given the following three files for this lab;

1. ‘main.c’ // This file contains the main application (menu based)
2. ‘employee.c’ // This file contains the functions to implement linked list
3. ‘employee.h’ // This file contains the structure definition and prototypes of functions.

In-Lab Task 1:

Make a new project and add the above mentioned files to this project. Compile and run the program. Use the function **`'int getListLength(struct employee * emp)'`** defined in employee.c to print the length of the linked list. You will have to add an option in the menu so that the user may see the length of the list any time he wishes.

In-Lab Task 2:

Write a function to implement ‘***search by key***’ feature for the linked list. This function should be able to search the database by ‘***age***’ and by **‘name’** and list the appropriate records. It is okay at this stage if only the first record with the selected parameter (aga or name) is displayed.

Post-Lab Task:

Write appropriate function to implement the delete function. This function should allow the user to delete a node with a given ***parameter*** (name, age).

Make sure that you deallocate the memory for the deleted node (using C function free()).

******** End of Document ********