Module 9 – More Dynamic Memory Allocation

Tutorial Questions

Objectives

To understand some of the concepts relating to dynamic memory allocation and dynamic memory structures in C. To understand the important concepts involved in designing dynamic data structures (structures that can grow), using a linked list as a first example of this.

1. Implement a function to delete a 'person' from your list. It should take as its second parameter a c string that contains the person's last name.

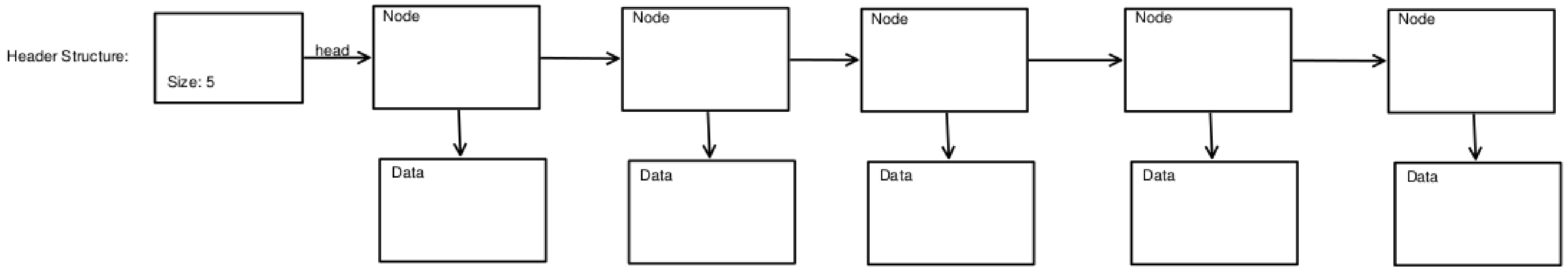
2. Implement a function called List_Print(List * list). It should call a NodePrint(Person * data) function to handle the printing of each list node's data.

3. Implement a function to Print the list in the reversed order. This time we will use recursion and the NodePrint() function that we defined in the previous question.

4. Implement a function to free the Person List we have created that can be called from main() before we exit the program.

5. Implement a main function which prompts the user to enter a series of “Persons”. You will need to declare a list and initialise it. You will need to allocate the memory for each person before calling the appropriate function to insert this user in the list. . If the user presses enter at a new line we know they have finished entering data. Finally, call the print and free functions that we implemented today.

The Linked List Data Structure


Header Structure:
Size: 5
head
Node
Node
Node
Node
Node
Data
Data
Data
Data
Data