COMPUTER PROGRAMMING
2016-2017 FALL TERM

HOMEWORK #9
Due Date : 24.12.2016 ; 23:55

PART 1 (70 pts):

Doubly Linked List : Doubly Linked List is a variation of Linked list in which navigation is possible in both ways, either forward and backward easily as compared to Single Linked List. Following are the important terms to understand the concept of doubly linked list.

- Doubly Linked List contains a link element called first and last, also called head and tail respectively.
- Each node carries a data field(s) and two link fields called next and prev.
- First element of the list carries a link as null to prev, and the last element carries a link as null to next.
- Each node is linked with its next node using its next link.
- Each node is linked with its previous node using its previous link.

Circular Linked List : Circular Linked List is a variation of Linked list in which the first element points to the last element and the last element points to the first element. Both Singly Linked List and Doubly Linked List can be made into a circular linked list.

- In singly linked list, the next pointer of the last node points to the first node.
- In doubly linked list, the next pointer of the last node points to the first node and the previous pointer of the first node points to the last node making it circular in both directions.

Consider the definitions of two different types of Linked Lists. You will implement both lists as libraries and you will use these data structures to store a data type **"Circuit"** that you defined; it contains **char current [3]** and **int volts**. You are expected to make following operations :

- Insert A Node : **insert_node(char *current, int volts, int position)**

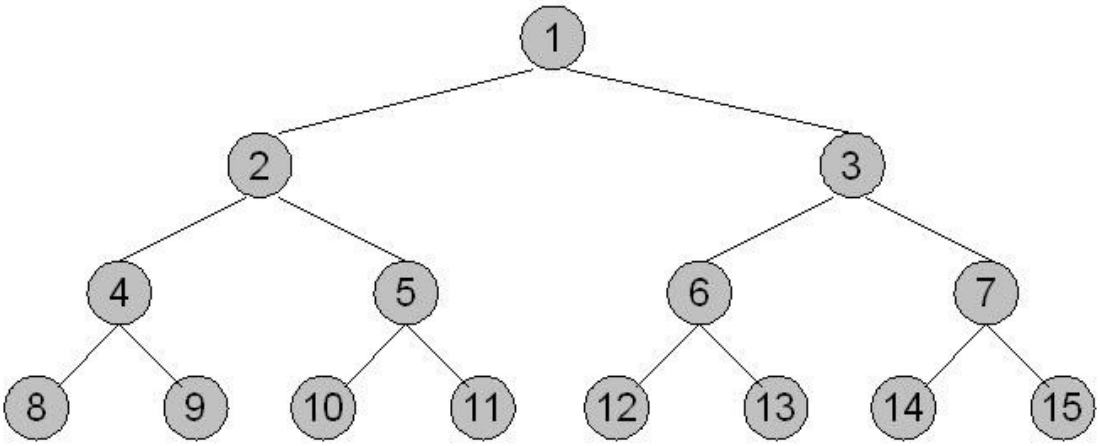
  A new node will be inserted at the position. For example position=1 means insert at the front, position=4 means insert after node 3 but if there are only 2 nodes in the list then an error should be printed.
- Delete A Node : **delete_node(node *n)**
- Display List : **display_list()**
- Size List : **size_list()**

  Returns size of the linked list

BONUS (20 pts): If you use Makefile while compilation and linking you will earn 20 more points for part 1.

PART 2 (30 pts):

Consider the following binary tree :



Assume a program visits nodes on the tree by pre-order, in-order and post-order traversal methods respectively. List the output of each method.

Hint : You **are not** expected to implement a binary tree or write a code for traversal methods.

RULES:

1. Obey honor code principles.
2. Read your homework carefully and follow the directives about the I/O format (data file names, file formats, etc.) and submission format strictly. Violating any of these directives will be penalized.
3. Obey coding convention.
4. Your submission should include the following file and NOTHING MORE (no data files, object files, etc):

   HW09_<Firstname>_<Lastname>_<student number>DoubleList.c

   HW09_<Firstname>_<Lastname>_<student number>DoubleList.h

   HW09_<Firstname>_<Lastname>_<student number>CircularList.c

   HW09_<Firstname>_<Lastname>_<student number>CircularList.h

   HW09_<Firstname>_<Lastname>_<student number>main.c

   Makefile (If there is)

   Do NOT compress the files you submit.
5. Do not use non-English characters in any part of your homework (in body, file name, etc.).