IOOM - Object Oriented Programming

Lab Assignment – 2 **Batch R1 & R2**

Evaluation: Monday August 20, 2018, at 2.00 PM

Note: i) Input should be taken only from the user and not hard coded.
ii) You need to make all data members either private or default.
iii) Make all methods either public or default.

Batch R1 & R2

Problem 1:
Create a class 'item' that contains the following data fields:
1. item_id(integer)
2. Quantity(integer)
3. item Label(character array of size 10)

and a suitable constructor .

Create another class 'Stack' which contains an array of such items and data member 'top' and 'size'. In the constructor, initialize the top to -1.

Create methods to push( ), pop( ) , is_empty( ) and display( ) the stack. Note that the push method will call the constructor of the 'item' class.

Problem 2:
Augment the above program to include the following two methods in the stack class:

a. A method to sort the elements in the stack only using the class 'Stack' methods defined previously (push(),pop(), is_empty()) such that the item with the largest quantity appears at the top.

b. A method that removes the middle element in the stack (using only the standard stack methods defined in question 1).

For both questions, create a main method. Create an object for Stack and verify the correctness of the methods you've defined by invoking them appropriately.

IOOM - Object Oriented Programming

Lab Assignment – 2 **Batch R3 & R4**

Evaluation Date & Time: Tuesday August 21, 2018 at 2.00 PM

Problem 1:
.A structure 'node' represents aggregate marks of a student with the following components:

1. Student Name (Character array of size 20).
2. Total_Marks (integer).
3. Pointer to the left child.
4. Pointer to the right child.

Create a class 'tree' holding the pointer to root 'node', a constructor where root is initialized to null value. Create methods for insertion, deletion and inorder traversal.

Problem 2:
Augment the above program by including the following two functionalities in the form of member functions.

a. Take the Total_Marks of one student node as an argument and find out it's cousin nodes.
   find_cousin(Total_Marks);
b. Also write an iterative function for finding the addition of nodes at the maximum level.
   Maxlevel_sum();

For both questions, create a main method. Inside the main method, create an object for class tree and verify the correctness of the methods you've defined by invoking them appropriately.