# Time Constrained Laboratory (TCL)

MODULE TITLE SOFTWARE ENGINEERING 2 (CSY2006)

YEAR OF COURSE TWO

# TIME ALLOWED 2.5 hours

**Section I: Brief:**

Implement solutions in C++ for all the tasks in Section II. Test your program/code using appropriate test cases. Upon completion, submit your solutions to NILE using the submission procedure in Section III below.

Please note that access to **only** teaching materials, syntax references and other resources posted on NILE is permitted/encouraged. However, access to any external information sources via internet (e.g. google) is strongly discouraged. The University of Northampton Policy on Academic Misconduct and Plagiarism will be strictly implemented.

**Section II: Tasks**

1. Traverse the following binary tree using in-order, pre-order and post-order. Specify the sequence for each. (15 marks)



1. Consider the following graph. Implement the following tasks:
2. List nodes in depth-first traversal (10 marks)
3. List nodes in breadth first traversal (10 marks)

A picture containing indoor

Description automatically generated

1. Design an inventory class that stores the following members:

serialNum: An integer that holds a part’s serial number

manufactDate: A member that holds the date the part was manufactured

lotNum: An integer that holds the part’s lot number

The class should have appropriate members for storing data into and retrieving data from these members. Design a program that uses the given stack class [DynStack.h]. The program should have a loop that asks the user if he or she wishes to add a part to the inventory or take a part from the inventory. The loop should repeat until the user is finished.

If the user wishes to add a part to the inventory, the program should ask for the serial number, data of manufacture and lot number. The data should be stored in an inventory object and pushed onto the stack. If the user wishes to take a part from the inventory, the program should pop the top-most part from the stack and display the contents of its member variables. When the user finishes the program, it should display the contents of the member values of all the objects that remain on the stack. (25 marks)

1. Write a function template that accepts an argument and returns its absolute value. The absolute value of a number is its value with no sign. For example, the absolute value of -5 is 5, and the absolute value of 2 is 2. Test the template in a simple driver program. (20 marks)
2. Extend the given MathStack [and IntStack] classes provided to implement the following additional functions:

mult – Pops the two values off the stack, multiplies them and pushes their product onto the stack.

div – Pops the top two values off the stack, divides the second value by the first, and pushes the quotient onto the stack.

addAll – Pops all values off the stack, adds them and pushes their sum onto the stack.

multAll – Pops all values off the stack, multiplies them, and pushes their product onto the stack.

Demonstrate the class with a driver program. (20 marks)

**Section III: Submission Procedure**

1. Copy and paste all your code into a WORD document. Then take a screenshot of the console output after running at least two test cases. The console output must show the file directory path, date and time of the program execution. Copy and paste the screenshot into the same WORD document together with the code. Go to “**Submit Your Work**” on NILE and click on *SourceCodeDocument* to submit the WORD document via turnitin.

2. The package containing your programs must be zipped and uploaded to NILE, e.g. *johnSmith.zip* where John Smith is your student name. Go to “**Submit Your Work**” on NILE and click on *TclEsubmission*. This will take you into the submission form, where you can upload your zipped folder using the ‘Attach File’ button (*Browse for Local File*) and then clicking the *Submit* button.

**Section IV: Marking Criteria**

The grade for this assessment will form one part (50%) of the overall assignment grade for the module. 25 marks have been allocated to each program/task above. Programs written by the user will be marked using the following criteria:

|  |  |
| --- | --- |
| **Aspect** | **Weighting** |
| Task Solution (Working Code) | 60% |
| Code Quality and Efficiency | 10% |
| Comments and Code Convention | 15% |
| Testing | 15% |

**Note:**

1. The specification given for each task (e.g., program name, method name, etc.) and all the submission guidelines must be strictly followed. If not, penalties will be imposed.

2. The University of Northampton Policy on Academic Misconduct and Plagiarism will be strictly implemented. You may be asked for an online viva-voce should there be any doubts in the originality of the submitted work.

3. The module tutor reserves the right to invite students to a meeting to discuss coursework submissions.

**END OF DOCUMENT**