

**COMP1003**

**Algorithms, Data Structures and Mathematics**

**20 CREDIT MODULE**

**ASSESSMENT: 100% Coursework W1: 30% Set Exercises**

**W2: 70% Report**

**MODULE LEADER: Thomas Wennekers**

**MODULE AIMS**

* To give students a basic understanding of algorithmic design and practice
* To give students a basic understanding of mathematical principles underlying Computing

**ASSESSED LEARNING OUTCOMES (ALO):**

1. Explain the importance of algorithmic design in optimizing use of computing resources.
2. Review fundamental mathematical and logical principles that underlie modern computer science.
3. Identify suitable structures and algorithms to implement programming tasks.
4. Synthesize the solution to a real-world task as a combination of two or more standard algorithms.

**Overview**

This document contains all the necessary information pertaining to the assessment of *COMP1003 Algorithms, Data Structures and Mathematics*. The module is assessed via **100% coursework**, across two elements: *30% Set Exercises* and *70% Report*.

The sections that follow will detail the assessment tasks that are to be undertaken. The submission and expected feedback dates are presented in Table 1. All assessments are to be submitted electronically via the respective DLE module pages before the stated deadlines.

|  | Submission Deadline | Feedback |
| --- | --- | --- |
| Set Exercises (30%) | **12/03/2024 3pm** | Within 20 working days |
| Report (70%) | **29/04/2024 3pm** | Within 20 working days |

Table 1: Assessment Deadlines

All assessments will be introduced in class to provide further clarity over what is expected and how you can access support and formative feedback prior to submission. Whilst the assessment information is provided early in the module, it is not necessarily expected you will start this immediately – as you will often not have sufficient understanding of the topic. The module leader will provide guidance in this respect.

**Assessment Element 1: Mathematics Exercises**

This is an individual assignment. It is worth 30% of the total module mark.

The mathematics part of the module is assessed by Set Exercises in the form of an online questionnaire on the DLE page of the module.

The module page is here: https://dle.plymouth.ac.uk/course/view.php?id=67618

The questions will be comparable to those in the weekly module worksheets during the first six weeks of the module. They will be randomly selected from a database of questions. There will be elementary questions, but also some more difficult ones. Some questions will be multiple choice, some require calculations, and some require textual input.

It is recommended to have paper and pencil at hand when attempting the questions. However, students will not need to upload workings. A calculator may also be useful. The Windows calculator in scientific mode would be sufficient.

To prepare, it is recommended that students do the weekly worksheets, study the solutions provided on the DLE, and watch any further supporting podcasts before attempting the questionnaire.

**The questionnaire will open 10/03/2024 at 15:00 and close 12/03/24 at 15:00.**

**Students have 2 hours to finish the questionnaire after they started it.** **They must finish before the deadline (12/03/24 at 15:00).** Therefore, students should start well before 13:00 on March 12th to be certain they can finish on time.

Students with registered Assessment Provisions that grant them extra time for assessments will be allowed extra time for the Questionnaire proportional to their allowance (for example, a student with 10 min/h allowance will have 2 hours and 20 minutes). They still must submit before the deadline.

**Extenuating Circumstances for the Set Exercise Questionnaire**

You can only apply for extenuating circumstances **before** taking the questionnaire. If you start the questionnaire this will count as an attempt, and you will receive marks according to your performance. Only one attempt is allowed. You will not be able to claim for ECs after you have started.

Students with 5-day extensions must finish the questionnaire within five working days after the original submission deadline.

Students with 10-day extensions must finish the questionnaire within ten working days after the original submission deadline.

**Threshold Criteria**

To achieve a pass (40%+), 40% of the total marks for this CW element

To achieve a merit (60%+), 60% of the total marks for this CW element

To achieve a distinction (70%+), 70% of the total marks for this CW element

**Assessment Element 2: Algorithms and Data Structures**

This is an individual assignment. It is worth 70% of the total module mark.

The Algorithms & Data Structures part of the module is assessed by a programming exercise in Visual Studio using C#.

The task is to implement a **Binary Search Tree** which in a basic form was covered in one of the module sessions and the corresponding worksheet.

For this task create a VS C# command line project called BinarySearchTree. The implementation must use the Program.cs file available on the DLE in the Assessment section. This file has been modified from the implementation provided for the worksheet on Binary Search Trees. All coding must be done in this single file imported into your Visual Studio project as the main Program file.

The file Program.cs contains further comments, help and instructions that you must follow.

Importantly:

You are only allowed to use the C-core of C# in your implementation, but not more advanced programming constructs (even if IntelliSense suggests them). If in doubt about what is in the C-core, inspect the Initial Coding Skill Requirements document in the General Documents and Information section of the COMP1003 DLE page. This constraint may cause VS to emit warnings, which you can ignore.

Also:

1. Do not rename any of the method names in the file Program.cs or change their arguments or return types or even their order in the file.
2. If you want to add methods on top of those requested, do this in the space indicated at the top of the Program between the lines THIS LINE and THAT LINE.
3. You can add data fields to the Tree, Node, and DataEntry structures if you find this necessary or useful. This may be required for some of the specific tasks outlined below. However, you are not allowed to add methods inside these classes (because we do not do Object Orientated Programming in COMP1003).

Specific Tasks

1. **Binary Search Tree**

[40 marks]

Complete the implementation of a binary search tree in the Program.cs file provided by filling in the code requested. The methods/functions to implement are:

static bool IsEqual(Node item1, Node item2)

static void InsertTree(Tree tree, Node item)

static bool SearchTree(Node tree, DataEntry value)

static bool SearchTreeItem(Node tree, Node item)

static void DeleteItem(Tree tree, Node item)

static int Size(Tree tree)

static int Depth(Node tree)

static Node Parent(Tree tree, Node node)

static Node FindMax(Node tree)

static void DeleteMin(Tree tree)

Short descriptions of each function and its arguments and return values are in the file. Do not change these method prototypes. Each method is worth a maximum of 4 marks individually.

1. **Set Data Type**

[12 marks]

Implement Set operations on top of your Binary Search Tree from part a). The methods/functions to implement are:

static Tree Union(Tree tree1, Tree tree2)

static Tree Intersection(Tree tree1, Tree tree2)

static Tree Difference(Tree tree1, Node tree2)

static Tree SymmetricDifference(Node tree1, Tree tree2)

This subtask interprets Trees as representing mathematical Sets and asks to implement the basic Set operations. Short descriptions of each function, its arguments and return values are in the file Program.cs. Do not change these method prototypes. Each method is worth a maximum of 3 marks individually.

1. **AVL Trees**

[18 marks]

Extend the implementation from a) by adding a tree balancing mechanism. Specifically, make your trees AVL Trees. Do not use another balancing mechanism. This is an advanced task and only recommended if you are a confident programmer. The same methods as in part a) are requested but with an AVL-type balancing mechanism. This will need extensions to the data structures defined in a) as well as, most likely, some auxiliary functions.

Start with reading about AVL Trees. The Wikipedia page about AVL Trees is a good start.

If you do task c) you do not need to implement task a) as both tasks require the same methods to be implemented. If you do c) you would receive up to 40 marks for the methods requested in a) and up to 20 marks for the balancing implementation. In other words, hand in only one solution, either for a) or for c) with the methods listed in a).

The Set functions/methods in b) should not depend on whether you implement balancing (a) or not (c). The Set functionality is built on top of the list of methods in a). These methods should be assumed to be opaque to the user and only their prototypes known.

An additional 10 marks are allocated for good and efficient use of data structures and algorithms. Try to make your implementation as efficient as possible.

An additional 10 marks are allocated for some error checking and/or handling of special cases, as well as some testing of your functions. Use the method stubs in the file Program.cs for this. Do not change the Main() method. (If useful, you are allowed to program individual test functions in the area in the Program between the lines THIS LINE and THAT LINE and call these in TreeTests() or SetTests().)

An additional 10 marks are allocated for good programming style, like proper layout, readability, identifier names, commenting, modularisation, and the like.

Do not use object-oriented programming other than method-free classes for the data containers. Do not use the C# Collection library. Keep your code simple and clear. The module’s DLE page contains an introduction into the C-core of C#. There are numerous examples in the lectures and labs showing what you can use.

**Format of coursework to be handed in:**

The complete Visual Studio project must be zipped up in its entirety and submitted on the DLE page of COMP1003. The project **must** run when unpacked without modification. All C# code **must** be in the file Program.cs in the project’s top-level directory. You don’t need to submit anything else.

**Threshold Criteria:**

To achieve a pass (40%+), 40% of the total marks for this CW element

To achieve a merit (60%+), 60% of the total marks for this CW element

To achieve a distinction (70%+), 70% of the total marks for this CW element

**General Guidance**

**Extenuating Circumstances**

There may be a time during this module where you experience a serious situation which has a significant impact on your ability to complete the assessments. The definition of these can be found in the University Policy on Extenuating Circumstances here:

<https://www.plymouth.ac.uk/uploads/production/document/path/7/7741/Extenuating_Circumstances_Policy_and_Procedures.pdf>

Students with valid EC’s may claim either for an extension period or for non-submission. Valid for Non-Submission means you will be asked to do a new piece of work. Please see below.

**Referral**

Please note that if you claim for non-submission and are offered a referral, you will be required to complete a NEW piece of work for the module. The new piece of work will assess that you have met the learning outcomes for the module but in a way that will be different to the original set piece. The referral is not a repeat or extension of the original coursework.

Carrying out a new piece of work means you will not be able to keep the marks already gained during the module. Eg: if you pass the set exercises (CW1) but do not submit the main work (CW2) AND are offered a referral, you will not keep the set exercises grade.

**Plagiarism**

All of your work must be of your own words. You must use references for your sources, however you acquire them. Where you wish to use quotations, these must be a very minor part of your overall work.

To copy another person’s work is viewed as plagiarism and is not allowed. Any issues of plagiarism and any form of academic dishonesty are treated very seriously. All your work must be your own and other sources must be identified as being theirs, not yours. The copying of another persons’ work could result in a penalty being invoked.

Further information on plagiarism policy can be found here:

Plagiarism: <https://www.plymouth.ac.uk/student-life/your-studies/essential-information/regulations/plagiarism>

Examination Offences: <https://www.plymouth.ac.uk/student-life/your-studies/essential-information/exams/exam-rules-and-regulations/examination-offences>

Turnitin (<http://www.turnitinuk.com/>) is an Internet-based 'originality checking tool' which allows documents to be compared with content on the Internet, in journals and in an archive of previously submitted works.  It can help to detect unintentional or deliberate plagiarism.

It is a formative tool that makes it easy for students to review their citations and referencing as an aid to learning good academic practice. Turnitin produces an ‘originality report’ to help guide you. To learn more about Turnitin go to:

<https://guides.turnitin.com/01_Manuals_and_Guides/Student/Student_User_Manual>

**Referencing**

The University of Plymouth Library has produced an online support referencing guide which is available here: <http://plymouth.libguides.com/referencing>.

Another recommended referencing resource is [Cite Them Right Online](http://www.citethemrightonline.com.plymouth.idm.oclc.org/); this is an online resource which provides you with specific guidance about how to reference lots of different types of materials.