CSP2348 Data Structures

Assignment 3: Paired programming project

- A mini-programming project in Python
- Video presentation: demonstration and reflection

Objectives from the Unit Outline

- Apply mathematical theory to algorithm analysis.
- Design algorithms using ADT and various data structures.
- Employ Python classes to encapsulate ADT and implement algorithms.

General Information:

- Assignment 3 (or A3 for short) is the major assessment of the unit. It is a group assignment with teamwork and individual work. The teamwork component is a **team assessment** to be completed by a group of **up to two students.** Your group needs to complete a mini programming project and write a project report. The individual work component is a **video presentation** to be done by individual team members. Therefore, two batches of submissions are required.
  - The first batch of submission includes submission of the project implementation (including source code/s, and executables if applicable) and a written report. This part is a *group* task so it should be completed by all group members and only submitted by the team leader (that is, the other team member should *not* submit it repeatedly).
  - The second batch of submission consists of one *video link* and an *optional Peer-Review* document. This is an individual task/work so each team member of the group should prepare and submit their own version of the work seperately. The video link is the web link to a video presentation, which should contain two parts: a demonstration of the project completed by the group, and an individual's *reflection* on the project work. You can also include an (optional) *peer-review* document within this batch of submission.
- Group formation for A3:
  - You are responsible for forming your team for the team-based project. You can form your team by signing in a group via the A3 group sign-up page on Canvas (e.g., via "People→Groups" section), or alternatively if you have decided to form a group with a friend/classmate, you can email your tutor with the details of all team members so he/she can help you set up a group. Make sure to form your team by week 8 or before.
  - Once a group is formed, the team members will be locked. - If you want to change the team members of your group, you must email your tutor to request a change. Group membership can only be changed within the first 9 weeks.
  - If you really wanted to do A3 yourself, that's fine. In such a case, you must let you tutor know your decision by or before week 8 and you should not expect a reduction in the workload of the assignment.
- The programming project consists of three tasks: The first task requests you to design and implement a simple algorithm to generate a balanced BST. The second task is to modify some existing tree-based algorithms/methods to implement some specific application scenario/s. The third is a dummy mini-project using AVL-tree ADT. Video presentations (to be submitted separately) are primarily for demonstrating your project completed and for a personal reflection on the work completed.

Due: *(See Due Date in Assignments Section on Canvas)*

Marks: 100 marks (which will be converted to 50% of unit mark)

1. The project

The project is divided in to three tasks/questions:

Q1: Balanced BST generation (10 marks)

Given a sequence of integers, e.g.,

9, -1, 45, 6, 8, 21, 34, 5, 55, 65, 543, 18, 90, 122, 132, 0, 66, 100, -12, 17

design an algorithm/method to re-arrange the order of the data items (i.e., form a new sequence of the integers) so that when the data items are inserted sequentially into an initially empty BST, the newly created BST will be a balanced BST.

a) Design the algorithm (in pseudo code) and analyse your algorithm.
b) Write a Python method to implement your algorithm.
c) Write a program that invokes the above method, thereby demonstrating your algorithm by
   (i). printing the re-arranged data items (i.e., a new sequence) generated by the algorithm, and
   (ii). building a BST using the newly generated data sequence as input (that is, insert the data items, one-by-one, into an initially empty BST). Print the tree shape of the BST.

   (Note: The program should use at least two datasets to demonstrate the algorithm, one of which is the dataset given above).

Q2: Application to binary tree traversal & BST (30 marks)

The *in-order* traversal algorithm is one of the three typical binary tree traversal algorithms. It traverses a binary tree in the order of "left-subtree; root node; right-subtree" (refer to *CSP2348_M6_Binary Trees.pptx* in Module 6). If you apply the *in-order* traversal algorithm to a BST (of integer keys), it will print the integer keys in in *ascending* order.

We now define a new binary tree traversal algorithm, called *inverse-in-order* traversal algorithm. It traverses the binary tree in the order of "right-subtree; root node; left-subtree". As a comparison, when it is applied to a BST (of integer keys), it prints the BST nodes in *descending* order.

Please refer to the Lab code, *TreeTest.py*, in Module 06. For a given BST, rooted at N, it prints the *Pre-order, In-order,* and *Post-order* traversal sequences of the BST. Using the *TreeNode* class and *BST* class defined in the Lab code/s in Module 6,

a) Modify the *in-order* traversal algorithm to implement the *inverse-in-order* traversal algorithm. Then, convert the algorithm to a method (e.g., inverse_inorder (self) if in Python) and add it into the BST class. Use some test data to test the new method.

b) Modify the *in-order* traversal algorithm to form two code versions (e.g., leaf_BST(self) and non_leaf_BST(self), if in Python): one prints all leaf nodes, and the other prints all non-leaf nodes of the BST;

c) Modify the *pre-order* traversal algorithm to form a new method (e.g., with method name of total_nodesBST(self, N), if in Python) such that, for a given node *N* in a BST, it counts the total number of nodes of the sub-tree rooted at N, and prints all nodes, including N, of the sub-tree.

d) Modify the *search* tree node algorithm to form a new method (e.g., with method name of depth_nodeBST(self, N), if in Python) so that, for a given node *N* in a BST, it calculates the depth of the node N (in the BST).

e) Modify the *post-order* traversal algorithm to form a new method (e.g., with method name of depth_subtreeBST(self, N), if in Python) so that, for a given node *N* in a BST, it calculates the depth of the sub-tree rooted at N.

f) Refer to the PPT CSP2348_M6_Binary Trees.pptx in Module 6, design an algorithm to delete a node from a BST. Then, convert the algorithm to a method (e.g., delete_ (self, key) if in Python) and add it into the BST class. Use some test data to test the new method.

Up on completion of the above steps, write Python code/s to implement a tiny BST application system that allows user to build a BST first and then do specific operations on the BST.

The system first displays a menu (let's call it the level-1 menu, see below) to build a BST.

1. Pre-load a sequence of integers to build a BST
2. Manually enter integer values, one by one, to build a BST
3. exit

This allows the user to build a BST in one of the two ways: The system takes a list of integers (i.e., either via an array or by key-in manually), inserts them one by one into an initially empty BST (note: do not balance the BST). It then displays a *Menu* (called *leve-2 menu*, see below) that allows the user to browse and execute the following *Menu* options:

1. Display the tree shape of current BST, and then show the *pre-order*, *in-order*, *post-order* and *inverse-in-order* traversal sequences of the BST
2. Show all leaf nodes of the BST, and all non-leaf nodes (separately)
3. Show a sub-tree and count its nodes
4. Show the depth of a given node in the BST
5. Show the depth of a subtree of the BST
6. Insert a new integer key into the BST
7. Delete an integer key from the BST
8. Exit

The system continues to loop and prompt user to choose one of the options until the user chooses to exit (i.e., menu option 8) .

For menu option 1, you can use the printTree (…) method (which is included in one of the Lab codes, e.g., *TreeTest_withTreeShape.py*, or develop your own method) to display the current tree shape of the BST. To display the inverse-in-order traversal order, the system invokes the method/algorithm developed in a).

For menu option 2, the system invokes the algorithm/method developed in b) to show all leaf nodes, and non-leaf nodes of the BST, respectively.

For menu option 3, the user is prompted to enter an integer N (as the key value of the BST). The system then searches the integer N from the BST: If found, it displays the sub-tree rooted at N, and it also counts and prints the total number of nodes of the sub-tree (e.g., by invoking the algorithm/method developed in c)). Otherwise, it displays "ERROR: Node <N> not found!", where <N> is the key value entered.

For menu option 4, the user is prompted to enter an integer N (as the *key* of a node). The system then searches for node N from the BST: If found, it calculates and prints the depth of the node N in the BST by invoking the related algorithm/method developed in d). Otherwise, it prints "ERROR: Node <N> not found!".

For menu option 5, the user is prompted to enter an integer N (as a node). The system then searches for node N from the BST: If found, it calculates and prints the depth of the subtree rooted at N by invoking the related algorithm/ method developed in e). Otherwise, it prints "ERROR: Subtree rooted at node <N> not found!".

For menu option 6, the user is prompted to enter an integer N. The system then searches the BST for the integer N: If found, it prints a message "ERROR: node key <N> already exists in the BST!". Otherwise, it inserts N as a new node of the BST (and it then displays the inverse-in-order traversal sequence of nodes of the BST, thus verifying that the new node was inserted).

For menu option 7, the user is prompted to enter an integer N. The system then searches the BST for an integer N: If found, it deletes the node from the BST (by calling the algorithm/method developed in f), and then displays the inverse-in-order traversal sequence of nodes of the BST, thus verifying that the node was deleted). Otherwise, it prints "ERROR: Node <N> not found!".

Menu option 8 will exit the current level-2 menu and force the program back to the level-1 menu.

To test your code, use the following list of integers (as a *Sample dataset*) to build a BST

58, 84, 68, 23, 38, 82, 26, 17, 24, 106, 95, 48, 88, 54, 50, 51, 53, 49, -6, -46

Q3: AVL tree: deleting a node (20 marks)

Refer to the *AVLTree* class given in the Lab code (see testAVLTree.py) in Module 07. For a given list of integers, the code inserts the integers, one by one, into an initially empty AVL tree. It then prints the *in_order traversal* sequence of the AVL tree, and finally displays the structure of the AVL tree in a specific format (e.g., for each node, it shows its height, balance factor, together with left-subtree, right-subtree, and leaf node indicators, etc. in a particular hierarchical structure).

Read the code and make sure you understand all methods in the class, *AVLTree*, including the methods insert(…), rebalance(…), lrotate(…), rrotate(…), and inorder_traverse(…), display(…), etc.

Your Task:

a) Refer to the PPT CSP2348_M7_AVL Trees.pptx in Module 7 (together with the content covered by Module 6), design an algorithm to delete a node from an AVL tree. Then, convert the algorithm to a method (e.g., delete_ (self, key) if in Python) and add the new method into the AVLTree class. Use some test data to test the new method.

b) Refer to Q2, add the *pre-order* and *post-order* traversal methods to the AVLTree.

Up on completion of the above steps, write Python code/s to implement a tiny AVLTree application system that allows user to build an AVL tree first and then do specific operations on the AVL tree.

To build an AVL tree, the system accepts a list of integers (as input), inserts them, one by one, into an (initially) empty AVL tree. Similar to Q2, you are required to provide two options for the user to build an AVL tree, e.g., via a level-1 menu of:

1. Pre-load a sequence of integers to build a AVL tree
2. Manually enter integer values/keys, one by one, to build an AVL tree
3. Exit

The system then displays a level-2 *menu* (see below) that allows the user to navigate through and execute the following *Menu* options (on the AVL tree built):

1. Display the AVL tree, showing the height and balance factor for each node.
2. Print the *pre-order*, *in-order,* and *post-order* traversal sequences of the AVL tree.
3. Print all leaf nodes of the AVL tree, and all non-leaf nodes (separately).
4. Insert a new integer key into the AVL tree.
5. Delete an integer key from the AVL tree.
6. Exit

The system continues to loop and prompt user to choose one of the options until the user chooses to exit (Note: you can use the codes/methods in Module 6 & 7, e.g., the display method for menu option 1).

To test your code, use the following list of integers (as a *Sample dataset*) to build your initial AVL tree:

58, 82, -55, 20, 35, 79, 23, 14, 0, -21, 103, 92, 44, 84, 50, 46, 47, 49, 45, 72, 89.

Some requirement/restriction on coding:

(i) Your codes must be self-supporting: you can use the standard Python library, codes or snippets from the unit's website, and the codes from our textbook. No other third-party programming packages or code snippets should be imported. - If you do need to include any other package than the above, please discuss this with your tutor in advance. If you "import" any non-standard package/s into your code that prevents your code from running in our lab environment, you do so at your own risk (of losing marks). In other words, you should make sure your codes work in our lab environment. - It is important to test your codes in our lab environment before submission.

(ii) The program should contain a portion of code for building a BST/AVL tree for testing with the sample dataset/s provided. You should also show the effects of using the sample dataset/s in your video demo.

2. The written report (15 marks)

Refer to the **Format requirement of the assignment report** in page 7. Write a project report to cover (but not limited to):

i. A brief introduction of the project and requirements.
ii. Algorithm design and implementation procedure (separated by Q1, Q2 and Q3).
iii. *(Optional)* User manual: application setting-up and usage.
(Note: If your project is implemented in a programming language other than Python, you must include this section to instruct your tutor in installing the required programming environment and your project code/s so that they can test your project code/s).
iv. *Test cases* to be used in your video demo (including *input/s* and *expected output/s* of each test case). Make sure to include test cases against the sample dataset/s provided in separate questions.
v. Snapshots showing application running status and input/outputs.

vi. Conclusions or summaries of the completed project.

Notes: (1) Project source codes are required and must be included separately in the submission zip file.

(2) When forming test cases, please consider the following points:

- How does your code react to correct input?
- How does your code react to incorrect input?
- Can you exit the program safely after testing a chosen task (or sub-task)?
- Do the test cases cover all primary functions of the system?
- Do all menu options functioning properly?
- Does the program exit as expected?
- Is it formatted well enough?

3. The video demonstration and reflection (20 marks)

This part of the assignment is individual work (not a group work), therefore each team member should produce their own version of video/s. The video should cover two parts of contents: a video demo and a personal video reflection

(Note: It is fine if you wish to separate the video in two short video clips, one covering the video demo and other reflection. In the latter case you should submit two video links)

(a) The video demonstration

This (first part of the) video should mainly be a demonstration of the project completed by your group, although you may also present other contents such as explanation of the project report, etc.

The requirements include:

- The duration of video should be between 5 and 10 minutes.
- Student identity verification (SIV): For academic integrity, a SIV is required for the video demonstration. When the video starts, you should take a few seconds to briefly introduce yourself while your computer camera captures your face, or your student ID card.
- The video should cover demonstrations over some test cases:
  (i) the case/s where a sequence of valid data is entered for BST;
  (ii) the case/s where a sequence of valid data is entered for AVL tree;
  (iii) the cases with some invalid data input (e.g., inserting a duplicated integer into a BST/AVL tree, deleting/searching a non-existing key from a binary tree, etc.), thus triggering the system to handle/display error messages, etc.
- Any special features of the project that you want to show.
- You should mention which part of the work (e.g., coding or report) is primarily your contribution to the group project.

(b) The video reflection

This is the second part of the video. This portion of the video should not exceed 5 minutes in length. In this video section, you should explain your own contribution to the teamwork, providing a personal commentary oo the teamwork. The following is a list of points you might consider in your reflection:

- What was your project about?
- How successful was your team?
- What was your key role in completing this project?
- How did completion of this task link to your work in this unit or other units in your study?
- What was your key takeaway from the teamwork/task?
- What did you learn from the process?
- What difficulties you (or your team) encountered during the project development and how did you overcome them?

- Which parts of the tasks were the most challenging, if any?
- Which parts were done well, and which parts of the work could be improved if you are given a chance? etc.

(Note: This video section may also be used to support un-even mark distribution between team members working in the group if the project contributions made by team members differ significantly.)

4. The Peer review *(optional)*

This is to grade performance for each team member, including yourself

This task is *optional* – If you choose to do it, please download the *peer review* form template from Canvas, fill in the form and make necessary remarks/comments, and finally submit it.

==========

Submission Instructions (3 marks)

- ***Project & written report** submission* (one submission per team)
  - This submission should include all codes implementing the project (e.g., 3 or more Python source codes, SQL codes, and executable/s if applicable), a written report and any additional documentation associated with the report/project. This submission should be submitted by the team leader only (i.e., through the team leader's Canvas submission portal - the other team member should not submit duplicated assignment version).
    - This submission should be one single compressed file (e.g., in .zip format), containing all your documents (i.e., the written report, source codes, executable files and any other support documents, if any). The written report file must be in Word or PDF format (please refer to the section of *Format requirement of the assignment report*). Please rename the .zip file in a format of

      <Team-leader's Student ID>_< team-leader's last & first name>_< the other team member's ID_ last name>_CSP2348_A3.zip.

      If the assignment is done by an individual, please rename the .zip file in a format of

      <Student ID>_last name>_< first name>_CSP2348_A3.zip.

      > *As an example, if the team leader's ID is 12345678, his name is Ben SMITH, the other team member's ID is 15555555 and his last name is MAK, the submission file should be named 12345678_SMITH Ben_15555555_MAK_CSP2348_A3.zip.*
      > *If the assignment is done by Ben SMITH alone, the submission file should be named 12345678_SMITH Ben_CSP2348_A3.zip*

    - Note that files found to contain viruses or other infecting mechanisms shall be awarded a ZERO mark (to all team members).
  - Your attention is drawn to the university rules governing cheating/plagiarism and referencing. Please refer to the section of *Academic Integrity and Misconduct* in page 9.
  - ECU rules/policies will apply for late submission of this assignment.
- ***The Video** submission* (one per team member)
  You should submit the video link/s only. The video link/s should be properly renamed and contains your own student ID and name.
- ***Peer review submission*** (*optional):*
  Fill in the blank form and submit it (note that a form template is provided on Canvas).

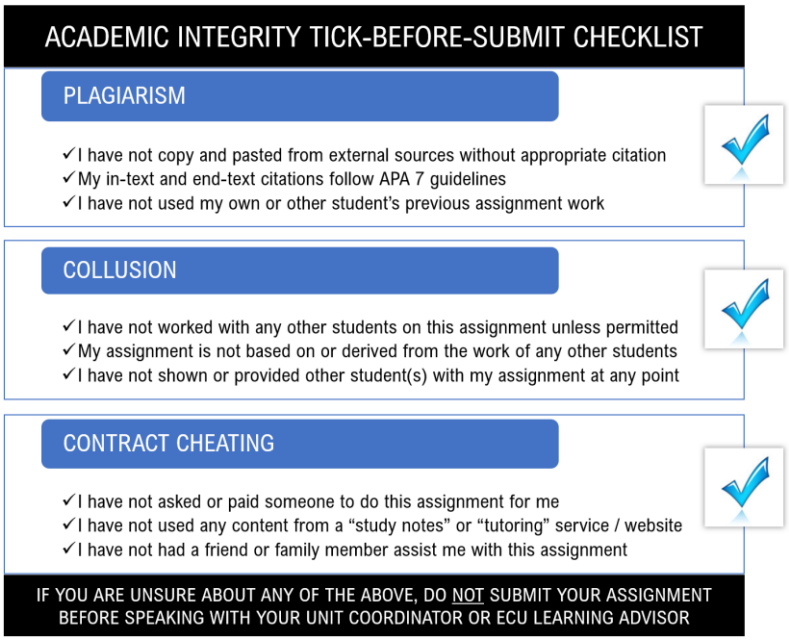
Note that separate submission links are available for each submissions.

Format requirement of the assignment report (2 marks):

<table>
<tr><td rowspan="4">Required Content</td><td>Cover page
Must show unit code/name, assignment title, the student IDs and names of all team members, due date, etc.</td></tr>
<tr><td>Executive Summary (optional)
This should represent a snapshot of the entire report that your tutor will browse through and should contain the most vital information needed.</td></tr>
<tr><td>Table of Contents (ToC)
This must accurately reflect the content of your report and should be generated automatically in Microsoft Word with clear page numbers.</td></tr>
<tr><td>Introduction
Introduce the report, define its scope, and state any assumption if any ; Use in-text citation/reference where appropriate.
The main body (note: you should use appropriate section title/s)
This part should contain (but not limited to):
• A short description of the programming tasks (e.g., the project scope: what are to be done, and what are excluded from the project, etc.)
• Any strategies used to solve the problems (e.g., an approach to develop a solution, if any).
• The questions being solved/answered, for example,
Q1. (should include but not limited to)
(i) The algorithm and analysis.
(ii) The snapshots: Input data sequence.
New data sequence generated by the algorithm.
The tree shape generated.
Q2: (should include but not limited to)
(i) (modified) algorithm/s (in pseudo codes) for sub-question (a) ~ f).
(ii) for the main system, a few screen shots (of running results of the code/s, e.g., one screenshot per menu option).
(iii) Test cases that your team members use to demonstrate the project in the video demo, etc.
Q3: (should include but not limited to)
(i) Algorithm/s (in pseudo codes) for sub-question a).
(ii) for the main system, take a few screen shots (of running results of the code/s).
(iii) Test cases that your team members use to demonstrate the project in the video demo, etc.
• For all questions, you may also include
(iv) Discussion/critique of the solutions achieved, if any.
(v) Any other contents you think necessary to be included.
• Python codes should not be included in body of the written report (but they should be saved as separate runnable codes and included in the submission zip file).
Conclusion
Summary of the works done in this project assignment.
References
A list of end-text reference, if any, formatted according to the ECU requirements using the APA format (Web references are also OK).</td></tr>
<tr><td>Other requirements</td><td>The document/report should be no more than 8 pages (excluding references, snapshots, and diagrams). The text must use font Times New Roman and be not smaller than 12pt.</td></tr>
</table>

Academic Integrity and Misconduct

- This assignment is a **group assignment** (with individual tasks). Your entire assignment must be your own work of the team (unless quoted otherwise) and produced for the current instance of the unit. Any use of uncited content that was not created by your team constitutes *plagiarism* and is regarded as Academic Misconduct. All assignments will be submitted to plagiarism checking software, which compares your submission version with previous copies of the assignment, and the work submitted by all other students in the unit (in the current semester or previous years/semesters).
- *Never give any part of your assignment to someone else or any other team*, even after the due date or the results are released. Do not work on your team assignment with other students or teams. – You can help someone (in other team) by explaining concepts, demonstrating skills, or directing them to the relevant resources. But doing any part of the assignment for them or with them or showing them your work is inappropriate. An unacceptable level of cooperation between students/groups on an assignment is *collusion* and is deemed an act of Academic Misconduct. If you are not sure about plagiarism, collusion or referencing, simply contact your lecturer or unit coordinator.
- You may be asked to explain and demonstrate your understanding of the work you have submitted. Your submission should accurately reflect your understanding and ability to apply the unit content and the skills learnt from this unit.
- Before submitting your assignment 3 (e.g., project and report), team leader and individuals should make sure they have checked all items in the **Academic Integrity tick-before-submit checklist** below and watch the video by clicking the link next to the checklist image:



Watch this video before submitting your assignment

Indicative Marking Guide:

	Description	Allocated Marks	You got
The project (Q1~Q3)	Q1: Balanced BST generation Algorithm/s, analysis, and implementation. Code runs, functions, and works for the given exemplar inputs. Overall quality of the work.	**10**	
	Q2: BST based application system development: Algorithm design & analysis for tasks a) ~ f). Implementation of *Menu* options (of 2 levels). Code runs, functions, and works for the given exemplar inputs Overall quality of the work.	**30**	
	Q3: AVL tree-based application system development: Algorithm design & Analysis for task a) (i.e., deleting a node from an AVL tree) Code/s for task b) Implementation of *Menu* options (of 2 levels). Code runs OK, functions, and works for the given exemplar inputs! Overall quality of the work.	**20**	
The written report	Executive summary: abstraction of the report & vital information as a whole; Thought/idea organization: conjunctions & cohesion; Clarity: discussion flow and integrity; Citations to References; Test cases; Conclusions achieved, etc. Quality of the report: Report presented as per Format requirement; Report length – not too long and too short (e.g., 1200-3000 words, in 5-10 pages?).	**15**	
Report format	Document presented as per format requirement	**2**	
Project/report Submission	All submissions (project code & report) are submitted as per submission requirement	**3**	
Video demo & reflection	As per requirement	**20**	
Peer review	*(optional)*	**0**	
Penalty	(*Possible Plagiarism or Late submission penalty*)		
Total	Total mark of 100 which is then converted to 50% of unit mark	**100 (/50%)**	

The END of the Assignment Description