Assignment 2 (15% of total marks)

Due date: Wednesday, 12 February 2025 by 9:00 pm (21:00 hours) Singapore time.

Scope

The tasks of this assignment consist of exercises in the areas of **Data Structures and Algorithms**, focusing on Data Structures and Sorting algorithms. The assignment covers the topics discussed in **Topics 3, 4, 5 and 6**.

The assignment consists of:

- **Part A**: Theory questions to assess your understanding of the subject matter and support the programming question's objective.
- **Part B**: Programming questions requiring the implementation of concepts related to algorithms and their complexity.

Marks for each question are indicated next to the respective question. The total mark for Assignment 1 is **100 marks**.

Marks Distribution

- **Part A**: 70%
- **Part B**: 30%
- **Total Marks**: 100
- **Weightage**: 15% of the total subject mark

Assessment Criteria

Marks will be awarded for:

- **Correctness**: Solutions must meet the requirements of the question.
- **Comprehensiveness**: Solutions should demonstrate a complete understanding of the topic.
- **Appropriate Application**: Answers should reflect proper application of the materials covered in this subject.

Deliverables

Part A: Theory Questions (Questions 1 to 3)

- Type your answers using **MS Word** and save the completed document as a **PDF file**.
- Alternatively, you may write your answers by hand onto a piece of paper or using tablet. Ensure your handwriting is **neat**, then scan or save your handwritten solutions into a **PDF file**.

Part B: Programming Question (Question 4)

- Implement your solution following the standard requirements provided for the question. You may use **C, C++, Java, or Python**.
- Execute your program and take a **screenshot of the output**.
- Save your source code in **pure ASCII text format** with the appropriate file extension (e.g., **.cpp** for C++, **.java** for Java, or **.py** for Python).
- Ensure your program is appropriately documented with comments.
- Provide clear **compilation instructions** in a readme.txt file. Include batch or makefiles if applicable for C or C++ programs.
    - *Note*: If no compilation instructions are provided, your lecturer will attempt compilation based on their environment. Any failure due to incompatibility will result in a "failure to compile" outcome.

Submission Instructions

- Combine the following into a single **ZIP file** named **SolutionA2.zip**:
    - Your solutions to the theory questions (in **PDF format**).
    - Your source code for the programming question (in **ASCII text format**).
    - The screenshot of your program’s output.
- Do not include subdirectories in your submission.

Part A: Theory Questions (70 Marks)

Question 1 – Linear Structure (15 Marks)

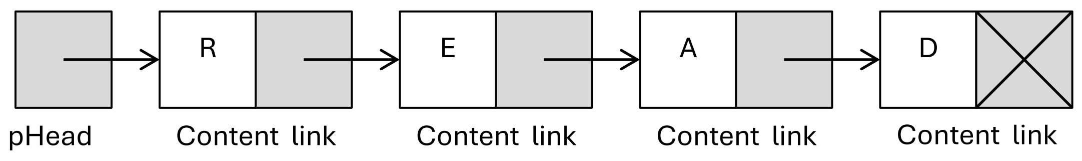
a) What would be the contents of queue Q1 and stack S1 after the following code is executed and the following data are entered?

```
$Q1 = createQueue$
$S1 = createStack$
$loop\ (not\ end\ of\ file)$
    $read\ number$
    $if\ (number\ not\ 0)$
        $pushStack\ (S1, number)$
    $else$
        $popStack\ (S1, x)$
        $popStack\ (S1, x)$
        $loop\ (not\ empty\ S1)$
            $popStack\ (S1, x)$
            $enqueue\ (Q1, x)$
        $end\ loop$
    $end\ if$
$end\ loop$
```

The data are 3, 27, 9, 0, 61, 38, 14, 48, 25, 19, 7, 0, 24, 39, 8, 29, 14, 0

(5.0 marks)

b) Imagine we have the linked list shown below.



```
Integer       nodePosition = 0;
nodePointer  Walker = null;
nodePointer      TempPointer = null;


Node:
```

Write a sequence of statements (in pseudocode) to update the 4th node that contain the value ‘D’ to ‘L’.

(5.0 marks)

c) A common problem for compilers and text editors is to determine if the parentheses or other brackets in a string are balanced and properly nested. For example, the string “((())())()” contains properly nested pairs of parentheses, but the string “)()(“ does not, and the string “())” does not contain properly matching parentheses.

Give an algorithm that returns the position in the string of the first offending parenthesis if the string is not properly nested and balanced. That is, if an excess right parenthesis is found, return its position; if there are too many left parentheses, return the position of the first excess left parenthesis. Return -1 if the string is properly balanced and nested. Use a stack to keep track of the number and positions of left parentheses seen so far.

(5.0 marks)

Question 2 – Non-linear Structure (25 Marks)

a. The **INORDER traversal** output of a binary tree is: M, O, Z, Z, A, R, E, L, L, A. The **PREORDER traversal** output of the same tree is: A, Z, M, O, E, A, Z, R, L, L.

   Using this information, construct the binary tree and determine the **POSTORDER traversal** output. Note that the tree you construct may differ from the trees constructed by your classmates, depending on how you choose to split the tree during construction. However, the **INORDER**, **POSTORDER**, and **PREORDER** traversal outputs must be consistent with your constructed tree. **(3.0 marks)**

b. One main difference between a Binary Search Tree (BST) and an AVL (Adelson-Velski and Landis) tree is the balance condition of the AVL tree. In an AVL tree, for every node, the heights of the left and right subtrees differ by at most 1.

   Starting with an empty BST and an empty AVL tree, insert elements with the following keys into both trees. If a key already exists in the tree, insert the duplicate into the **right subtree**.

   Keys to Insert:
   27, 32, 30, 17, 27, 38, 15, 30, 28, 29

   (6.0 marks)

c. Based on your work for Question 2b, determine the big-O (worst-case) time complexity for building a Binary Search Tree (BST) consisting of n nodes.

   Important: Provide a detailed explanation for your answer, as answers without explanations will not be awarded any marks.

   (Hint: This is a tricky question. Consider the worst-case scenario carefully before answering.)

   (3.0 marks)

d. Similarly, from what you have done for Question 2b, what is the big-o (worst case) complexity of the total time required to build an AVL tree consisting of n nodes?

   Important: Provide a detailed explanation for your answer, as answers without explanations will not be awarded any marks.

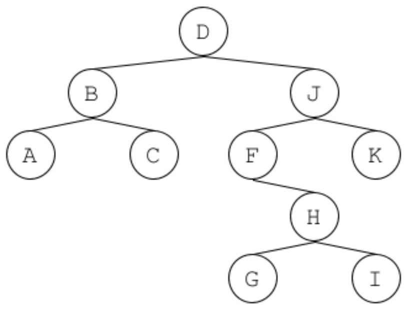
   (Hint: This is a tricky question. Consider the worst-case scenario carefully before answering.)

   (3.0 marks)

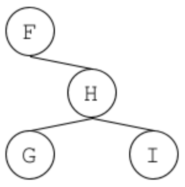
e. Describe an algorithm to determine whether a binary tree T is a **subtree** of another binary tree P. A tree T is a subtree of P if there exists a node in P such that the subtree rooted at that node is structurally identical to T.

   Use the provided diagrams to illustrate your explanation:

   - The top two diagrams show a case where T **is a subtree** of P.
   - The bottom two diagrams show a case where T **is not a subtree** of P.

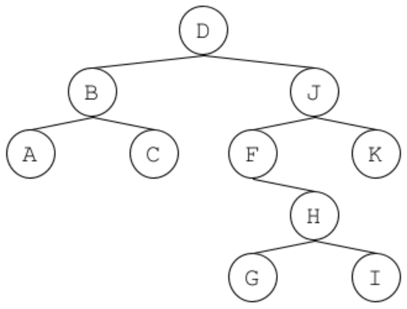


Tree P rooted at D

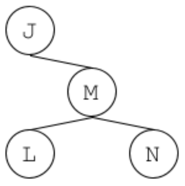


Tree T rooted at F

The tree T rooted at F is a subtree of the binary tree P rooted at D.



Tree P rooted at D



Tree T rooted at J

The tree T rooted at J is a subtree of the binary tree P rooted at D.

(10.0 marks)

Question 3 – Associative Table (30 Marks)

Consider a hash table of size 11 with hash function *h(x) = x mod 11*. Draw the table that results after inserting, in the given order, the following values: 55, 35, 54, 30, 77, 59, 65, 96, 125 for each of the three scenarios below:

a) When collisions are handled by separate chaining; **(5.0 marks)**

b) When collisions are handled by linear probing; **(5.0 marks)**

c) When collisions are handled by double hashing using a second hash function $h'(x) = (x \, mod \, 5) + 1$. Hint, the overall (combined) hash function is $H(x) = (\, h(x) + i \times h'(x) \,) \, mod \, 11$, where *i* = 0, 1, 2, 3, ... **(5.0 marks)**

d) When collisions are handled by quadratic probing with a quadratic probe function $h'(x, i) = (h(x) + 0.5 \, i + 0.5 \, i^2) \, mod \, 11$ where $i = 1, 2, 3, \ldots.$ **(5.0 marks)**

Part B: Programming Question (30 Marks)

Question 4 (30 Marks)

Specification:

Your task for this assignment is to investigate some of the properties of queues.

You should write a Java, C++, or Phyton program which simulates the queuing system in an email server.

Scenario:
Queues are widely used in network systems. For instance, emails are placed in a queue while waiting to be sent or after arriving at the recipient's mailbox. Problems may arise if the outgoing mail processor cannot send some messages in the queue, such as when the recipient's system is unavailable.

Task:
Write an email simulator that processes emails at an average rate of **30 messages per minute**. As emails are received, they are placed in a queue. Your program must adhere to the following requirements:

1. **Message Arrival**:
    - Messages arrive at an average rate of **30 messages per minute**, but the arrival time must be randomized. Use a random number generator to simulate this.
2. **Message Processing**:
    - Each minute, the simulator dequeues **up to 30 messages** and attempts to send them.
    - **25% of messages cannot be sent** in any given processing cycle. Use a random number generator to determine whether a message can be sent.
    - Messages that cannot be sent must be requeued (added back to the end of the queue).
3. **Simulation**:
    - Run the simulation for **15 minutes**.
    - Track the number of times each message is requeued during the simulation.

4. **Statistics**:
   At the end of the simulation, print the following statistics:
   - **Total messages processed**.
   - **Average arrival rate**: The average number of messages arriving per minute.
   - **Average messages sent per minute**.
   - **Average queue size per minute**.
   - **Breakdown of sent messages**: The number of messages sent on the first attempt, second attempt, etc.
   - **Average number of requeues**: Exclude messages sent on the first attempt from this calculation.
5. **Implementation Requirement**:
   - You **must implement your own Queue data structure** for this exercise.
   - Using any built-in Queue libraries or functions in your programming language is **not allowed**.

Sample Output:

```
Please enter the total minutes to run: 30

Total number of messages processed                    : 818

Average arrival rate                                  : 29.03

Average number of messages sent per minute            : 19.97

Average number of messages in the queue per minute    : 122.77

Number of messages sent on 1st attempt                : 491

Number of messages sent on 2nd attempt                : 87

Number of messages sent on 3rd attempt                : 18

Number of messages sent on 4th attempt                : 1

Number of messages sent on 5th attempt                : 2

Average number of times messages had to be requeued   : 1.30
```

These are just sample answers to show the output format required from your program. They are NOT necessarily the output that your program must produce because the data generated for this program were randomly generated.

Standard Requirements for Part B (Programming Question):

1. **Programming Environment**:
   - **Java**: Use JDK 6 Update 17 or higher (Windows).
   - **Python:** Python 3.x. Recommended version – Python 3.8 or higher. Use IDEs such as PyCharm (Community Edition) or VS Code.
   - **C++/C**: Use g++ 4.0 or higher (Windows or Ubuntu).
     - If using Ubuntu via a virtual machine (VM), ensure proper use of memory functions to avoid segmentation errors when compiling in a Windows environment.
2. **Original Work**:
   - All code must be your own work.
   - Do not use standard data structure or algorithm libraries (e.g., STL) or code from textbooks, the internet, or other external sources.
3. **Documentation**:
   - Include comments in your code to document its functionality.
4. **Program Execution**:
   - Execute your program and **include screen captures of the output** in your submission.
   - Submit all source code and any additional libraries used.
5. **Compilation Instructions**:
   - Include a **readme.txt file** with detailed instructions for compiling and running your program.
   - If instructions are unclear or missing, your lecturer will compile the code based on their own system settings. Any resulting incompatibility will be considered a failure to compile.
6. **Submission Naming Convention**:

- Follow the specific filename format provided in the submission instructions.
- Do not use your own filenames.

Sample solution includes but is not limited to the following:

Sorry, there is no sample solution for programming question.

Notes on Submission

1. This assignment is due on **Wednesday, 12 February 2025 by 9:00 pm (21:00 hours) Singapore time**.
2. Submission must be made via **Moodle**. Ensure all files are combined into a single **ZIP file** without subdirectories.
3. Clearly label all components of your submission.
4. Late submissions will incur a **5% deduction per day**, including weekends.
5. Submissions more than **seven** days late will not be marked unless an extension is granted.
6. Extensions must be applied for via **SOLS** before the assignment deadline.
7. Plagiarism is treated seriously. Instances of plagiarism may result in a zero mark for the assignment.

Submit the file **SolutionA2.zip** through Moodle in the following way:

1) Access Moodle at **http://moodle.uowplatform.edu.au/**
2) To login use a Login link located in the right upper corner the Web page or in the middle of the bottom of the Web page
3) When successfully logged in, select a site **CSCI203 (SP125) Algorithms and Data Structures**
4) Scroll down to a section **Submissions of Assignments**
5) Click at **Submit** your Assignment 2 here link.
6) Click at a button **Add Submission**
7) Move the files created into an area provided in Moodle. You can drag and drop files here to add them. You can also use a link *Add…*
8) Click at a button **Save** changes,
9) Click at check box to confirm authorship of a submission,
10) When you are satisfied, remember to click at a button **Submit assignment**.

A policy regarding late submissions is included in the subject outline. Only one submission per student is accepted.

Assignment 2 is an individual assignment, and it is expected that all its tasks will be solved individually without any cooperation with the other students. Plagiarism is treated seriously. Students involved will likely receive zero. If you have any doubts, questions, etc. please consult your lecturer or tutor during lab classes or over e-mail.

End of specification

If you have any questions or require clarification, please contact your lecturer as early as possible. Good luck!