Assignment 2 (20% of total marks)

Due date: 15 February 2024, Thursday

Scope:

The tasks of this assignment cover the **data structure and algorithm**. The assignment covers the topics discussed in topics 3 and 4.

The assignment is divided into two parts – Part One covers the theoretical aspect of the materials discussed during classes, and Part Two covers the practicality of the concepts. The total mark for Part One is 75, and Part Two is 25.

Assessment criteria:

Marks will be awarded for:

- Correct,
- Comprehensive, and
- Appropriate

application of the materials covered in this subject.

Marks:

Total mark: 100
Weightage: 20% of total subject mark

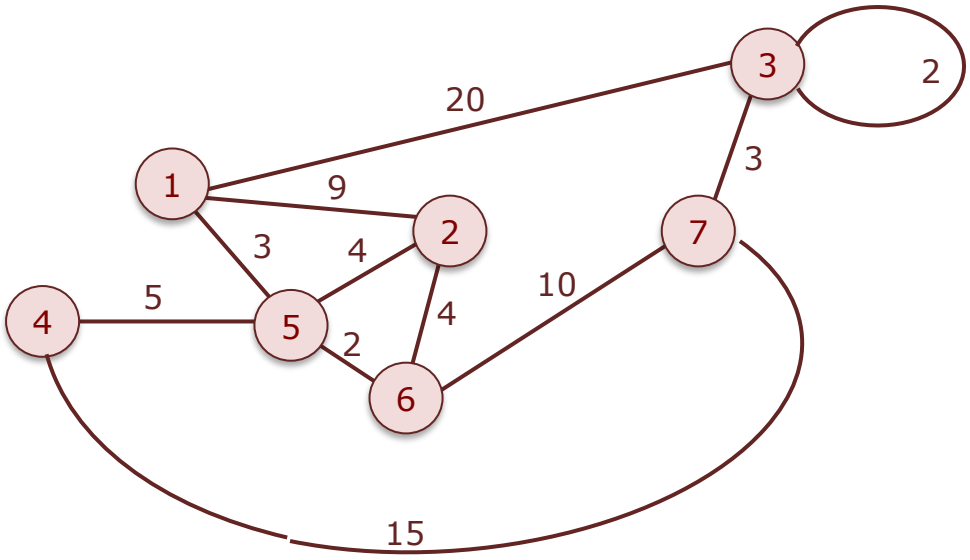
Assignment Specification:

Part A: (75 marks)

Question 1 (25 marks)

a. The INORDER traversal output of a binary tree is A,B,N,O,R,M,A,L,L,Y and the PREORDER traversal output of the same tree is Y,N,A,B,M,O,R,L,A,L. Construct the tree and determine the output of the POSTORDER traversal output. **(6.0 marks)**

b. Given the following undirected graph:



Represent the graph as:

(i) Adjacency matrix **(3.0 marks)**
(ii) Adjacency list **(3.0 marks)**
(iii) Incidence matrix **(3.0 marks)**

c. Starting with an empty 2-4 tree, construct a 2-4 tree with the following keys. Show the major working steps.

14, 12, 11, 13, 16, 15

(10.0 marks)

Question 2 (25 marks)

Do a dry run on the two algorithms (ALGORITHM 1 and ALGORITHM 2) shown below.

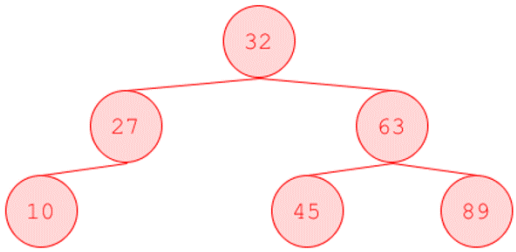
```
ALGORITHM 1
Function A1(root)
  if (root == NULL)
   return −1
  endIf
  x = root.data
  Q = new Queue()
  ENQUEUE(Q,root)
  while (!ISEMPTY(Q)) do
     t = PEEK(Q)
     if (t.data > x)
        x = t.data
     else
        ENQUEUE(Q,t.right)
        ENQUEUE(Q,t.left)
        DEQUEUE(Q)
     endIf
   endWhile
   return x
End of function A1
```

```
ALGORITHM 2
Function A2(root)
  if (root == NULL)
   return −1
  endIf
  t = root
  while (t.right != NULL) do
     t = t.right
  endWhile
  return t.data
End of function A2
```

Note: the function ENQUEUE only inserts a new element in the queue if this element is different from NULL.

a) Briefly explain what the purposes of the two algorithms are and state the asymptotic run-time complexity of each of the algorithms.

b) For the following Binary Search Tree (BST):



What is returned by the function call A1(root)?

(5.0 marks)

c) For the Binary Search Tree (BST) in part (b), provide a detail analysis on the run-time complexity of the ALGORITHM 2, as explained in lecture.**(5.0 marks)**

d) Re-write the function A2, in pseudocode, using recursive function calls. You **may not** use any form of iteration. **(10.0 marks)**

e) For a general Binary Search Tree (BST) of N elements, which of the two algorithms A1 and A2, should you use? Give your choice and explain your reasoning. **(5.0 marks)**

Question 3 (25 marks)

The data structure shown in Figure 1 depicts an implementation of a list of two lists, that is, two linked list joined into one. In this example, the data structure is implemented to group two separate lists of numbers, one consists of lists of even numbers and the other consists of lists of odd numbers.

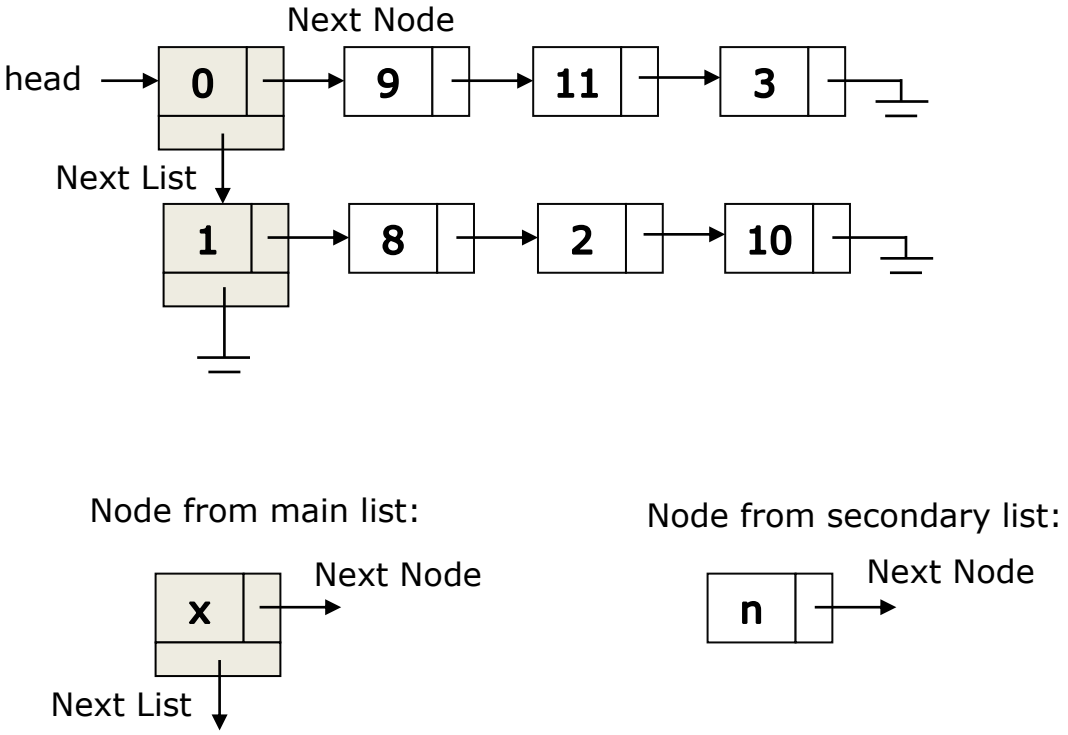


Figure 1: A list of two lists

Two types of linked list are used in the above implementation. The node of the main list (highlighted in grey) consists of a content field, a next node field, and a next list field. The content node holds a value 0 for an odd list and a value 1 for an even list. The node also contains two reference fields. The first is a 'next node' that links to the first node of a secondary list, and the second is a 'next list' that links to the next link list.

The node of the secondary list (non-highlighted) consists of a content field and a next node field. For an even list, the content holds an even number and for an odd list, the content holds an odd number. The node has a 'next node' field that link it to the next secondary list node.

In a single linked list implemented in Java, this is the definition of a node:

```
class Node {
   int data;
   Node nextNode;

   // Constructor to create a new node
   // nextNode is by default initialized as null
   Node(int d) { data = d; }
}
```

a) Assuming that the above definition is kept for the nodes of the secondary lists, device a new definition of node for the nodes of the main list. Call this type of node MainNode and use the names of pointers NextNode and NextList respectively in your implementation. **(5.0 marks)**

b) Write the pseudocode of the function INSERT(head,x) that inserts a new number in this data structure. If the number is even it must go to the second secondary list, the list that starts with node 1 in the main list. Otherwise, it must go to the first secondary list. Assume the data structure already has the main list created and numbers are inserted at the start of the secondary list. **(5.0 marks)**

c) Write the pseudocode of the function SEARCH(head,x) that returns TRUE if the number x is in the data structure (in any of the secondary lists) and FALSE otherwise. **(5.0 marks)**

d) Write the pseudocode of the function DELETE(head, b) that receives as input arguments the head of the last of two lists and a Boolean value. The function DELETE(head,b) deletes one of the main nodes. If b equals 0, then the node storing number 0 is deleted. Otherwise, the main node storing number 1 is deleted. Consider the following cases: the main list is empty and it has only one node (node 0 or 1). **(10.0 marks)**

Part B: (25.0 marks)

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.

Queues are commonly used in network systems. For example, e-mail is placed in queues while it is waiting to be sent and after it arrives at the recipient's mailbox. A problem occurs, however, if the outgoing mail processor cannot send one or more of the messages in the queue. For example, a message might not be sent because the recipient's system is not available.

Write an e-mail simulator that processes mail at an average of 20 messages per minute. As messages are received, they are placed in a queue. For the simulation, assume that the messages arrive at an average rate of 30 messages per minute. Remember, the messages must arrive randomly, so you will need to use a random number generator to determine when messages are received.

Each minute, you can dequeue up to 20 messages and send them. Assume that up to 25% of the messages in the queue cannot be sent in any processing cycle. Again, you will need to use a random number to determine whether a given message can be sent. If it cannot be sent, put it back at the end of the queue or enqueue it.

Run the simulator for 15 minutes, tracking the number of times each message had to be requeued. At the end of the simulation, print the statistics that show:

1 The total messages processed.
2 The average arrival rate, that is, the average number of messages arriving per minute.
3 The average number of messages sent per minute.
4 The average number of messages in the queue in a minute.
5 The number of messages sent on the first attempt, the number of messages sent on the second attempt, and so forth.
6 The average number of times messages had to be requeued (do not include the messages sent the first time in this average.)

NOTE: Since the question is to assess your understanding of the concept of Queue, you are NOT allowed to use the library of the language that implement queue. You need to write the codes (implementation) of Queue for this exercise. (See point (iii).)

Sample Output:

Please enter the total minutes to run: 30

Total number of messages processed	: 565
Average arrival rate	: 29.60
Average number of messages sent per minute	: 28.7
Average number of messages in the queue per minute	: 65.07
Number of messages sent on 1st attempt	: 391
Number of messages sent on 2nd attempt	: 87
Number of messages sent on 3rd attempt	: 18
Number of messages sent on 4th attempt	: 4
Number of messages sent on 5th attempt	: 2
Average number of times messages had to be requeued	: 1.30

Note: 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 numbers shown were randomly generated.

Standard Requirements for Part B (Programming question):

(i) Java Version – JDK 6 update 17 or higher (Using Windows), or
(ii) C++ / C compiler – g++ 4.0 or higher (Using Windows or UBUNTU). In the event that you use UBUNTU via VM, be careful with the memory function of the language and make sure that the functions are used properly and do not cause any segmentation error when the codes are compiled in Windows environment.
(iii) All coding must be your own work. Standard libraries of data structures and algorithms such as STL may not be used.
(iv) Programs should be appropriately documented with comments.
(v) Execute your program and screen-capture the output. Include in your submission all source code and libraries plus the screen-captures.
(vi) Students are to place all compilation and instructions on how to run the program inside a readme.txt file. Your lecturer will refer to this file when marking. Without a readme.txt or clear instructions for compilation, your lecturer will compile based on his/her computer setting; any incompatibility, will deem as failure to compile the program.
(vii) Submission filenames are to follow the naming convention given in the submission instruction below. Do not use your own filename.

Submissions

This assignment is due by 9:00 pm Singapore time on Thursday, 15 February 2024.

- For Part A, **type or hand-written** your answer for each question in a MS Word or equivalent document format and save it in a pdf formatted file, name your file as YourUOWStudentNumber-A2-SolPartA.pdf.
- For Part B, the name of your program should be QueueSim.cpp, QueueSim.java, or QueueSim.py depending on the programming language that you use to develop your program. Execute your program and **screen capture** your output. Next, zip your source code, libraries, readme.txt together with your screen capture and name your file as YourUOWStudentNumber-A2-SolPartB.zip.
- Zip together YourUOWStudentNumber-A2-SolPartA.pdf and YourUOWStudentNumber-A2-SolPartB.zip and name your file as YourUOWStudentNumber-A2.zip. Do not use your own filename.
- All assignments that do not satisfy the submission requirements listed above will not be evaluated and will be returned to the students with 0 marks.

Submit the files **YourUOWStudentNumber-A2.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 (SP124) 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 a file, for example, **YourUOWStudentNumber-A2.zip** into the submission area. 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 a button Submit assignment,
10) Click at the checkbox with a text attached: By checking this box, I confirm that this submission is my own work, … in order to confirm authorship of your submission,
11) Click at a button Continue.

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.