**SUTD 50.001 Introduction to Information Systems and Programming**

**Problem set 3 Part B**

Note: Please submit your answer to eDimension

**Cohort Questions (Week 6)**

**Session 1**

Title: **Stack and Queue with Singly Linked List**

Q1 [15 points] Using the SinglyLinkedList class discussed in the lecture, develop i) the Stack class and ii) the Queue class.

|  |  |
| --- | --- |
| **Operations of Stack**   * + IsEmpty: return true if stack is empty, return false otherwise   + Top: return the element at the top of stack   + Push: add an element to the top of stack   + Pop: delete the element at the top of stack   + DisplayStack: print all the data in the stack   **Sample menu**  1. Push  2. Top  3. Pop  4. isEmpty  5. displayStack  **Sample input/output**  1  Enter integer element to push  1000  Stack = 1000  3  Item popped = 1000  Stack = empty  … | **Operations of Queue**   * + IsEmpty: return true if queue is empty, return false otherwise   + Enqueue: add an integer element to the rear of queue   + Dequeue: delete the element at the front of queue   + DisplayQueue: print all the elements in the queue without altering it   **Sample menu**  1. Enqueue  2. Dequeue  3. DisplayQueue  4. isEmpty  **Sample input/output**  3  Queue=1000->2000->3000  Do you wish to continue?  Y  Enter choice  2  Item dequeued: 3000  Queue = 1000->2000  … |

**Session 2**

Title: **Trinary Search Tree**

Q2. [15 points] Design a Trinary Search Tree based on the concept of binary search tree. Instead of each node having two children, a node now has three child nodes, a left, a middle and a right child. Do preorder, inorder and postorder traversals of the given tree using the test case provided. You may use either iterative or recursive solutions. Modify binary search tree node to hold an **int** instead of **char**. If a node has a single value your algorithm should insert into that node rather than proceed to the next level. Assume that you are dealing only with positive integers and -1 is used to represent a null value. If you compare the Trinary Search Tree with Binary Search Tree (with int instead of char), do you see any difference in traversal results? Why?

Please enter your answers to these questions as comments to the code. Alternatively you may use the text box within eDimension to submit your answers. Augment the TrinarySearchTree with the methods

* public int search (Node root, int x, int depth)
* public int findMin(Node root)
* public int findMax(Node root)

**Input:**

**int[] num = {100, 200, 590, 70, 80, 360, 110, 105, 102,107};**

**Output:**

**PreOrder Traversal:**

**100 200 : 70 80 : 105 110 : 102 -1 : 107 -1 : 360 590 :**

**InOrder Traversal:**

**70 : 80 : 100 : 102 : -1 : 105 : 107 : -1 : 110 : 200 : 360 : 590 :**

**PostOrder Traversal:**

**70 80 : 102 -1 : 107 -1 : 105 110 : 360 590 : 100 200 :**

**Input:**

t.search(t.*root*, 100, 0);  
t.search(t.*root*, 590, 0);  
t.search(t.*root*, 107, 0);

**Output:**

**found 100 at depth 0 found 590 at depth 1 found 107 at depth 2**

**Input:**

System.***out***.println(**"Minimum value: "** + t.findMin(t.*root*));  
System.***out***.println(**"Maximum value: "** + t.findMax(t.*root*));

**Output:**

**Minimum value: 70**

**Maximum value: 590**

**Homework Questions (Week 6)**

Title: **StockObjectList**

Q1. [20 points] You have to store a number of StockObjects in a doubly linked list, where each object has a header and value field. For the purpose of the homework you may store a random String in the value field and the size of the string in the header. Use the DoublyLinkedList to store the strings in sorted manner, sorted according to the length of each string, by the header. Support the functions: Enter object, Delete object, Display object, Display Structure.

Note: Though the StockObjectList can be implemented using SinglyLinkedList, the choice of DoublyLinkedList is to facilitate insertion and traversal in both directions if needed in future.

**Operations of StockObjectList**

* + enterObject: Allows the user to enter a stock object in a string format. It must be stored in increasing order by string length.
  + displayObject(pos): returns the string at position pos in the StockObjectList
  + dumpObjects: prints out all the objects in ascending order of string-length
  + isEmpty: returns true if the StockObjectList is empty, false otherwise.

**Input/output sample:**

**Enter object:**

**Spanner 6 inch**

**Enter object:**

**Screw Driver – Philips**

**Display object(2)**

**Screw Driver – Philips**

**Dump**

**Stock Object List = Spanner 6 inch Header: 14 <-> Screw Driver - Philips Header: 22**

Title: **ArrayOfTrees**

Q2. [20 points] Implement an ArrayOfTrees in order to do a simple word occurrence counter in a piece of text. Your goal is to count the frequency of occurrence of each word in the text. Ignore case. Use the first letter of each word to index into an array of trees. Thus there will be twenty-six trees, one for each letter of the alphabet. You may ignore any string in the text that starts with a non-alphabetic (not a-z) character. Ignore punctuation. Hyphenated words count as one word.

Return the count of occurrences of the words that are supplied in the test case.

**Example test input:**  
**"This is a piece of text from which hopefully you will make some trees."**

**Output:**

**Inorder Traversal: a (1)**

**Inorder Traversal: from (1)**

**Inorder Traversal: hopefully (1)**

**Inorder Traversal: is (1)**

**Inorder Traversal: make (1)**

**Inorder Traversal: of (1)**

**Inorder Traversal: piece (1)**

**Inorder Traversal: some (1)**

**Inorder Traversal: This (1) text (1) trees. (1)**

**Inorder Traversal: which (1) will (1)**

**Inorder Traversal: you (1)**

Title: **kd Tree**

Q3. [10 points] Familiarize yourself with the kd Tree code given out in the starter code. Try to increase the dimensionality from 2 to k > 2, for example 3 or 4. Please answer (briefly) the following questions: (please submit your answers in a pdf file with the rest of your answers)

1. What is the complexity of the find Nearest Neighbour algorithm?
2. What would be some good applications of the kd tree?

Title: **AVL Tree**

Q4. [10 points] Familiarize yourself with the AVL Tree code given out in the starter code. Try to get an estimate for its performance by increasing the size of the tree in terms of n, the number of keys stored in the tree. Please answer (briefly) the following questions: (submit in a pdf file)

1. What is the time complexity of the insert algorithm?
2. What would be some good applications of the AVL tree?

Title: **B Tree**

Q5. [10 points] Familiarize yourself with the B Tree code given out in the starter code. Try to vary the size and dimensionality of the B-Tree. Please answer (briefly) in a pdf file, the following:

i) What is the time complexity of the insert algorithm as a function of k, the dimensionality of the B-Tree?

ii) What would be some good applications of the B tree? If SSD RAM were to become the norm in laptops and notebooks, would we still need the B tree? Explain your answer briefly.