**EECS2040 Data Structure Hw #4 (Chapter 5 Tree)**

**due date 5/30/2021, 23:59**

***Format***: Use a text editor to type your answers to the homework problem. You need to submit your HW in an HTML file or a DOCX file named as **Hw4-SNo.docx** or **Hw4-SNo.html**, where SNo is your student number. Submit the **Hw4-SNo.docx or Hw4-SNo.html** file via eLearn. Inside the file, you need to put the **header and your student number, name (e.g., EECS2040 Data Structure Hw #4 (Chapter 5 of textbook) due date 5/30/2021 by SNo, name)** first, and then the **problem** itself followed by your **answer** to that problem, one by one. The grading will be based on the correctness of your answers to the problems, and the **format**. Fail to comply with the aforementioned format (file name, header, problem, answer, problem, answer,…), will certainly degrade your score. If you have any questions, please feel free to ask me.

**Part 1**

1. (8%) What is the maximum number of nodes in a k-ary tree of height h? Prove your answer.

Sol:

Proof: h=1: n=1, h=2: n=1+k, h=3: n=1+k+k2 ⇒

1. (12%) For a simple tree shown below,
2. Draw a list representation of this tree using a node structure with three fields: tag, data/down, and next. (For convenience, you can omit the tag field.)
3. Write down a generalized list expression form for this tree.
4. Convert the tree into a left-child and right-sibling tree representation
5. Draw a corresponding binary tree for this tree based on (c).



Sol:



(b) 🡪 (A (B(…), C(…), D(…))) 🡪 (A (B(E(…), F), C(G), D(H(…), I, J)))

🡪 (A (B(E(K, L), F), C(G), D(H(M), I, J)))

(c)



(d)



1. (10%) Draw the internal memory representation of the binary tree below using (a) sequential and (b) linked representations.



Sol:

1. Sequential representation: tree[]: (height=4⇒at most 24 -1 = 15 nodes ⇒ tree[16])

| - | A | B |  | C | D |  |  | E |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

subscript 0 1 2 3 4 5 6 7 8 9

(b) linked representation:



1. (4%) Extend the array representation of a complete binary tree to the case of complete trees whose degree is d, d > 1. Develop formulas for the parent and children of the node stored in position i of the array.

Sol: A complete d-ary tree with height h would have at most (dh – 1)/(d-1) nodes.

Number the nodes from 1 ~ n (corresponding to array position) where n is the number of nodes of the tree and n ≤ (dh – 1)/(d-1).

It’s easy to show that for a node i, its children starts from di-(d-2), …, di, di+1.So its leftist child would be at di-(d-2), and its rightist child is at di + 1, if di-(d-2), …,di, di+1 ≤ n. Otherwise the corresponding child does not exist.

For node i, its parent would be at

(check d = 2)

1. (16%) Write out the inorder, preorder, postorder, and levelorder traversals for the following binary trees.



Sol: (a) inorder: A–B\*C\*D+E

Preorder: +\*\*-ABCDE

Postorder: AB-C\*D\*E+

Level order: +\*E\*D-CAB

(b) inorder: HDJBEAFCG

Preorder: ABDHJECFG

Postorder: HJDEBFGCA

Level order: ABCDEFGHJ

1. (30%) Given a sequence of 11 integer number: 50, 5, 30, 40, 80, 35, 2, 20, 15, 60, 70.

(a) Assume a Max heap tree is initialize with these 11 numbers placed into nodes of the tree according to node numbering of complete binary tree. Please draw the final Max heap tree after initialization process.

(b) Repeat (a) for Min Heap.

(c) Using the BST Insert function, (manually) insert the 11 number sequentially to construct a binary search tree. Draw the final 11-node BST.

Sol:



Original initial binary tree corresponding max heap



Corresponding min heap



Constructed binary search tree using a series of 11 Insert()

1. (20%) An 8-run with total of 25 numbers are to be merged using Winner tree and Loser tree. The numbers of the 8 runs are shown below. The first numbers form each of the 8 runs have been placed in the leaf nodes of the tree as shown. Then these eight numbers enter the tournament to get the overall winner.



1. Draw the winner tree and indicate the overall winner of this tournament.
2. Draw the loser tree and indicate (draw) the overall winner of this tournament.

Sol:



**Part 2 Coding**

You should submit:

(a) All your source codes (C++ file).

(b) Show the execution trace of your program.

1. (30%) Develop a complete C++ template class for binary trees shown in **ADT 5.1**. You must include a **constructor**, **copy constructor**, **destructor**, the four traversal methods together with forward iterators for each, functions in ADT 5.1, … as shown below.

void Inorder()

void Preorder()

void Postorder()

void LevelOrder()

**void** NonrecInorder()

**void** NoStackInorder()

**class** InorderIterator

**class** PreorderIterator

**class** PostorderIterator

**class** LevelOrderIterator

**bool** **operator ==** (**const** BinaryTree& t) **const**

Write 2 setup functions to establish 2 example binary trees (e.g., the trees in Part 1 Question 5). Then **demonstrate** the functions you wrote.

**ADT 5.1 BinaryTree**

**template**<**class** T>

**class** BinaryTree

{ // objects: A finite set of nodes either empty or consisting

// of a root node, left BinaryTree and right BinaryTree

**public**:

BinaryTree(); // constructor for an empty binary tree

**bool** IsEmpty(); // return true iff the binary tree is empty

BinaryTree(BinaryTree<T>& bt1, T& item, BinaryTree<T>& bt2);

// constructor given the root item and left subtrees bt1 and right subtree bt2

BinaryTree<T> LeftSubtree(); // return the left subtree

BinaryTree<T> RightSubtree();// return the right subtree

T RootData(); // return the data in the root node of \***this**

};

1. (35%) (a) Write a C++ class MaxHeap that derives from the abstract base class in **ADT 5.2 MaxPQ** and implement all the virtual functions of MaxPQ.

**ADT 5.2 MaxPQ**

**template** <**class** T>

**class** MaxPQ {

**public**:

**virtual** ~MaxPQ() {} // virtual destructor

**virtual** **bool** IsEmpty() **const** = 0; //return **true** iff empty

**virtual** **const** T& Top() **const** = 0; //return reference to the max

**virtual** **void** Push(**const** T&) = 0;

**virtual** **void** Pop() = 0;

};

(b) Write a C++ abstract class similar to ADT 5.2 for the ADT **MinPQ**, which defines a min priority queue. Then write a C++ class MinHeap that derives from this abstract class and implement all the virtual functions of MinPQ.

Use Part1 Q6 example to demonstrate your program.

1. (35%) A Dictionary abstract class is shown in **ADT5.3 Dictionary**. Write a C++ class BST that derives from Dictionary and implement all the virtual functions. In addition, also implement

pair<K, E>\* RankGet(**int** r),

void Insert(int r, pair<K,E>& thePair)

void Delete(int r),

**void** Split(**const** K& k, BST<K, E>& small, pair<K, E>\*& mid, BST<K, E>& big)

**ADT5.3 Dictionary**

**template** <**class** K, **class** E>

**class** Dictionary {

**public**:

**virtual** **bool** IsEmptay() **const** = 0; // return true if dictionary is empty

**virtual** pair <K, E>\* Get(const K&) **const** = 0;

// return pointer to the pair w. specified key

**virtual** **void** Insert(**const** pair <K, E>&) = 0;

// insert the given pair; if key ia a duplicate, update associate element

**virtual** **void** Delete(**const** K&) = 0; // delete pair w. specified key

};

Generate at least two sets of 20 (key, value) pairs each to construct the BST. Demonstrate your functions using these two sets of records.