COMP 2004 Data Structures and Algorithms

Assessment 2

Total Points: 200

Submission Instructions

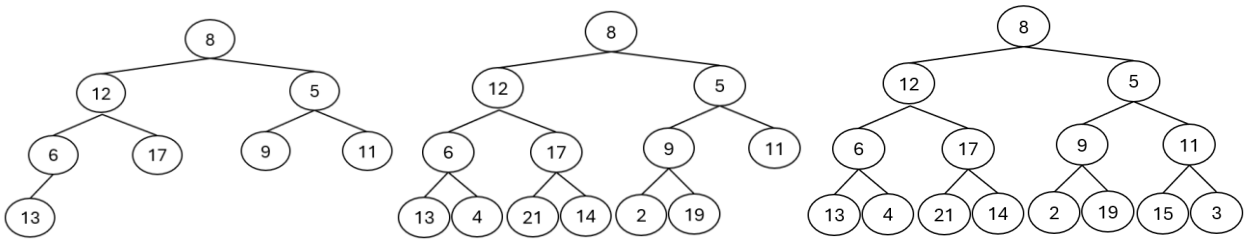
1. All questions in this assignment require software code answers. Submit your C source code only. Do not submit object code (.o files) or executables (.exe files)
2. gcc compiler in VSCode IDE will be used for compiling and testing the code
3. All submissions, code and text answers, must be made via Canvas only. Do not email your submissions. Emailed submissions will not be graded.
4. Name your submitted files as <Your last name>_<Assignment number>_<question number>.c For example, if your last name is Smith, for assignment 2, question number 1 your answer (C code) should be in a file named Smith_2_1.c
5. Put all your submission files inside a folder named as <Your last name>_<Assignment number>. For example, if your last name is Smith, for assignment 2, the submission folder should be named Smith_2
6. Late submissions and partial credits will be according to the class policies given in the first lecture slides.
7. All students must adhere to the honor code of the course that they signed on the first day of classes.

For Questions 1-3 use the COMP2004_Assignment2_driver.c and binary_tree_from_list.c files provided with Assessment 2.

Inputting a Binary Tree From Command Line

First a few words about these programs. The binary_tree_from_list.c file takes the the level order traversal of a binary tree (that we discussed in the live lesson on March 1, 2024; also discussed here) and makes a binary tree from it. The level order traversal gives the data elements of the tree level by level, starting from level 0, that is, at the root.

The program requires the input to be specified as a complete binary tree – a tree where every level is completely filled, except possibly the lowermost level **and** all nodes at the lowermost level are as far left as possible. Three examples of complete binary trees are shown below:



The COMP2004_Assignment2_driver.c program asks the user to input the number of nodes and the level order traversal of the tree from command line as a space separated list of integers.

For the leftmost tree above, the input would look like this:

```
Enter the number of data elements in the tree: 8
Enter the tree data elements: 8 12 5 6 17 9 11 13
```

For the middle tree above, the input would look like this:

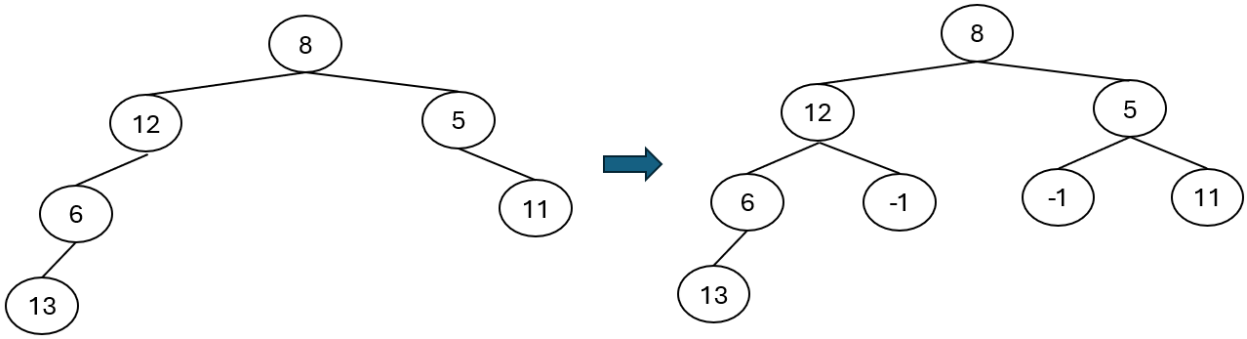
```
Enter the number of data elements in the tree: 13
Enter the tree data elements: 8 12 5 6 17 9 11 13 4 21 14 2 19
```

It then calls the binary_tree_from_list.c, which returns the root of the constructed tree. The driver program also prints the InOrder traversal of the constructed tree to make sure that it was constructed correctly. You can look into the programs to see how these work.

Bottomline, you don't necessarily need to understand the inner workings of binary_tree_from_list.c for this assignment, but just need to know that it returns the root to a tree structure given a level order traversal from command line. These programs are given just to make the input of binary trees simpler via command line, instead of students have to write code to construct the tree.

How to Specify Incomplete Binary Trees using Above Programs

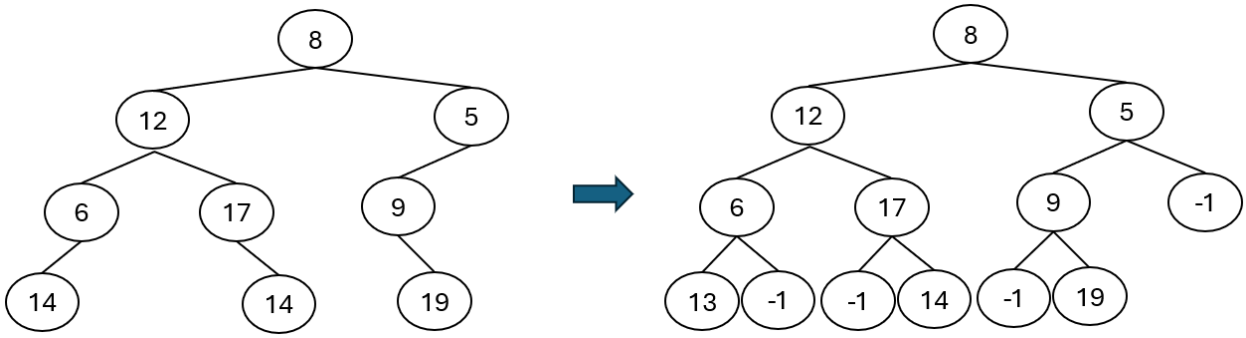
We can also specify incomplete binary trees with the binary_tree_from_list.c program. BUT we will have to pad all the internal NULL nodes with dummy nodes, for example, dummy nodes with a sentinel value -1. The binary_tree_from_list.c will automatically remove the -1 nodes to construct the tree on the left with the NULL pointers at the correct places. Two examples are shown below:



The corresponding command line input would be:

```
Enter the number of data elements in the tree: 8
Enter the tree data elements: 8 12 5 6 -1 -1 11 13
```

You can check the inorder traversal printed by the driver program to make sure that it was constructed correctly. It should be 13 6 12 8 5 11.



The corresponding command line input would be:

```
Enter the number of data elements in the tree: 13
Enter the tree data elements: 8 12 5 6 17 9 -1 13 -1 -1 14 -1 19
```

The inorder traversal of the constructed tree will print as 13 6 12 17 14 8 9 19 5.

Now, we are ready to start with the programming for the assignment questions.

Question 1 (60 points; 15 points for each part)

Make a copy COMP2004_Assignment2_driver.c and rename the copy to <Your Lastname>_2_1.c

Add functions to the COMP2004_Assignment2_Q1.c file for each of the sub-parts given below.

Write a menu driven interface for your program to call the functions and print the returned value, as shown below:

```
Enter the number of data elements in the tree: 9
Enter the tree data elements (space separated integers representing a
complete binary tree): 12 8 3 4 -1 5 -1 -1 7

Enter a choice:
[1] Count Nodes
[2] Sum Keys
[3] Max Key
[4] Print Below (this choice should prompt the user to input the value 'v')
[5] Exit
```

a) Write a function `ElementType count_nodes(BinaryTree T)` that counts and returns the number of items in a binary tree.
b) Write a function `ElementType sum_keys(BinaryTree T)` that returns the sum of all the keys in a binary tree.

c) Write a function `ElementType max_key(BinaryTree T)` that returns the maximum value of all the keys in a binary tree. Assume all values are nonnegative; return -1 if the tree is empty.
d) Write a function `void print_below(BinaryTree T, ElementType v)` that prints all the keys less than a given value `v` in a binary tree.

Question 2 (30 points;15 points for each part)

As in Question 1, make a copy COMP2004_Assignment2_driver.c and rename the copy to <Your Lastname>_2_**2**.c

Add functions to the COMP2004_Assignment2_Q2.c file for each of the sub-parts given below.

Write a menu driven interface for your program to call the functions and print the returned value, as shown below:

```
Enter the number of data elements in the tree: 9
Enter the tree data elements (space separated integers representing a
complete binary tree): 12 8 3 4 -1 5 -1 -1 7

Enter a choice:
[1] Tree Height
[2] Max Path Cost
[3] Exit
```

a) The height of a tree is the maximum number of edges on a path from the root to a leaf node. Write a C function `int height(BinaryTree T)` that returns the height of a binary tree.
b) The cost of a path in a tree is sum of the keys of the nodes participating in that path. Write a C function `int path_cost(BinaryTree T)` that returns the cost of the most expensive path from the root to a leaf node.

Question 3 (30 points; 15 points for each part)

Once again, make a copy COMP2004_Assignment2_driver.c and rename the copy to <Your Lastname>_2_**3**.c

Add functions to the COMP2004_Assignment2_Q3.c file for each of the sub-parts given below.

Write a menu driven interface for your program to call the functions and print the returned value, as shown below:

```
Enter the number of data elements in the first tree: 9
Enter the tree data elements (space separated integers representing a
complete binary tree): 12 8 3 4 -1 5 -1 -1 7

Enter the number of data elements in the second tree: 5
Enter the tree data elements (space separated integers representing a
complete binary tree): 3 8 12 5 4

Enter a choice:
[1] Structure Identical Check
```

```
[2] Structure and Value Identical Check
[3] Exit
```
.

a) Given two binary trees, return true if and only if they are structurally identical (they have the same shape, but their nodes can have different values).
b) Given two binary trees, return true if they are identical (they have nodes with the same values, arranged in the same way).

Question 4 (80 points: 20 points for part a), 20 points for part b), 40 points for part c))

For this question you should use the code discussed in the BST module (`bst.h, bst.c and test_bst.c`).

Add functions to the `bst.c` file for each of the sub-parts given below. For each function, remember to add the corresponding function prototype in `bst.h` file. Also include a menu driven interface for your code, as shown below:

```
Enter a choice:
[1] Insert (this choice should prompt the user to input the data element of the node)
[2] findMin_K (this choice should prompt the user to input the value 'k')
[2] numLeafNodes
[3] LevelOrder
[4] Exit
```

a) Write a function `Position findMin_K(SearchTree T, int k)` to find the kth smallest element in the binary search tree.
b) Write a function `int numLeafNodes(SearchTree T)` which returns the number of leaf nodes in a binary search tree
c) To traverse a binary tree in level order means to start at the root, and then visit all nodes one node away from the root (starting with the left node, if not null); and then visit all nodes two nodes away from the root (again, left-most to right-most), as they exist; and then visit all nodes 3 nodes away from the root and so on. Write a function `ElementType* LevelOrder(SearchTree T)` that returns a queue of the keys of the binary search tree in level order.