Programming Assignment #2

Lecturer: Prof. Seung-Hwan Baek
Teaching Assistants: Chunghyun Park, Kanghee Lee, Seungjoo Shin, Dongmin You

***** PLEASE READ THIS GRAY BOX CAREFULLY BEFORE STARTING THE ASSIGNMENT *****

Due date: 11:59PM October 19, 2022

Evaluation policy:

- Late submission penalty
    - 11:59PM October 19 ~ 11:59PM October 20
        - Late submission penalty (30%) will be applied to the total score
    - After 11:59PM October 20
        - 100% penalty is applied for that submission
- Your code will be automatically tested using an evaluation program
    - Each problem has the maximum score
    - A score will be assigned based on the behavior of the program
- We won't accept any submission via email - it will be ignored
- Please do not use the containers in C++ standard template library (STL)
    - Such as:
        - #include <queue>
        - #include <vector>
        - #include <stack>
    - Any submission using the containers in STL will be disregarded

File(s) you need to submit:

- pa2.cpp, tree.cpp, tree.h, heap.cpp, heap.h (Do not change the filename!)

Any questions? Please use PLMS - Q&A board.

0. Basic instruction

a. Please refer to the attached file named "DataStructure_PA_instructions.pdf".

1. Quiz (2 pts)

1.1. Let T is a general tree, and T' is a binary tree converted from T. Which of the following traversal visits the nodes in the same order as **the inorder traversal** of T'?

(1) Preorder traversal of T
(2) Inorder traversal of T
(3) Postorder traversal of T
(4) None of the aboves

1. What is the time complexity of **rearranging** min-heap into a max-heap?

(5) O(1)
(6) O(log n)
(7) O(n)
(8) O(2^n)

- Example execution
  - If you choose "(1) Preorder traversal of T" for 1-1., print your answer as shown below

```
>> ./pa2.exe 1 1
[Task 1]
1
```

  - If you choose "(1) O(1)" for 1-2., print your answer as shown below

```
>> ./pa2.exe 1 2
[Task 1]
1
```

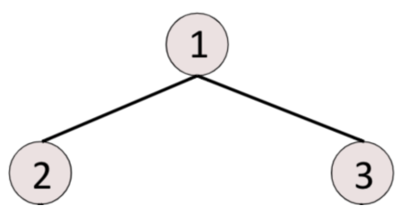
pre-2. Construct Binary Tree

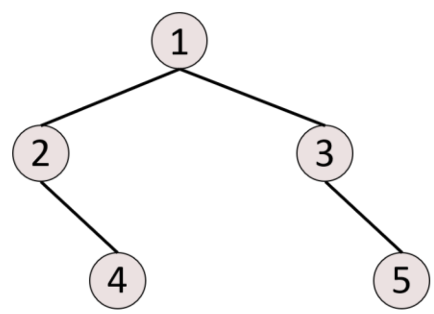
Note: pre-2 is not a problem that will be evaluated, but this is a short pre-requisite to solve problems 2,3, and 4.
Don't worry. We are providing utility functions to help you.

a. For problems 2, 3, and 4, you would need to implement member functions of `BinaryTree` class. To construct a `BinaryTree` class instance from an input, we use the string with bracket representation as input. The recursive definition of the bracket representation is as follows.

   `Tree = Root(LeftChild)(RightChild).`

   Below are some examples.
   The left tree is represented as `1(2)(3),` and the right tree is `1(2()(4))(3()(5))`





b. To implement "a", we provide a function to construct `BinaryTree` class from the bracket representation, which is `BinaryTree::buildFromString` function. It creates a pointer-based `BinaryTree` class instance from the given string. It would be helpful to read the implementation details of `BinaryTree::buildFromString`

c. To sum up, you will need to use `BinaryTree` class for problems 2, 3 and 4. Please try to understand the code for `BinaryTree` class.

2. Traverse Binary Tree (2 pts)

a. Implement `BinaryTree::preOrder`, `BinaryTree::postOrder` and `BinaryTree:inOrder` function that can traverse a binary tree with given traverse mode

b. Input & Output

Input:

- String with bracket representation.
- String representing traverse mode. Either “`preorder`”, “`postorder`” or “`inorder`”

Output:

- A sequence of node values acquired from the tree traversal. The value is separated with a white space

c. Example input & output

Input	Output
“1(2)(3)” “preorder”	1 2 3
“1(2()(4))(3()(5))” “postorder”	4 2 5 3 1
“4(2(3)(1))(6(5))” “preorder”	4 2 3 1 6 5
“4(2(3)(1))(6(5))” “inorder”	3 2 1 4 5 6
“4(2(3)(1))(6(5))” “postorder”	3 1 2 5 6 4

d. Example execution

```
>> ./pa2.exe 2 “4(2(3)(1))(6(5))” “inorder”
[Task 2]
3 2 1 4 5 6
```

3. Depth/Height of Binary Tree (3 pts)

a. Implement `BinaryTree::getDepthHeight` function that can calculate the depth and height of a specific node in a given binary tree.

b. Input & Output

Input:

- A given binary tree represented by string with bracket representation.
- All node values in the tree are unique.
- A specific node represented by integer value.

Output:

- Depth and height of the specific node in a given binary tree.
- If the specific node doesn't exist in the binary tree, return "`error`".

c. Example input & output

Input	Output
`"1(2)(3)" 2`	`1 0`
`"1(2(3(4)))(5)" 3`	`2 1`
`"1(2(3(4)))(5)" 6`	`error`

d. Example execution

```
>> ./pa2.exe 3 "1(2(3(4)))(5)" 3
[Task 3]
2 1
```

4. Properness, Fullness, Completeness of Binary Tree (3 pts)

a. Implement `BinaryTree::isProper, BinaryTree::isFull, BinaryTree::isComplete` function that can check whether if the given binary tree is a proper, full, complete binary tree or not

b. Input & Output
Input:
- String with bracket representation
- Specify what you want to check. Either “proper”, “full”, “complete”

Output:
- String “True” if the giben binary tree is a binary tree that matches the property, “False” otherwise

c. Example input & output

Input	Output
“1(2)(3)”, “proper”	True
“1(2(4)(5))(3(6))”, “proper”	False
“1(2)(3)”, “full”	True
“1(2(4)(5))(3()(7))”, “full”	False
“1(2)(3)”, “complete”	True
“1(2(4)(5))(3(6))”, “complete”	True
“1(2()(4))(3(6))”, “complete”	False

d. Example execution

```
>> ./pa2.exe 4 “1(2(4)(5(6)(7))(3)”, “proper”
[Task 4]
True
```

5. Min-heap Insertion (2 pts)

Note: For solving problems 5 and 6, the similar utility functions provided in PA1 will be used to parse an input string. Therefore, you won't need to try implementing a string parser. Please read pa2.cpp, and find the lines where your code would be located.

a. Implement a function that **inserts** a new element to a binary min-heap. Your heap should maintain the min-heap property even after the insertion. Each test case will insert less than 100 values

b. Input & Output

Input: A sequence of commands

- `(‘insert’,integer)`: insert `integer` into the current min heap

Output:

- Values in a heap in a node number order, in a string separated with the white space (automatically printed with built-in function)
- Do not consider the exceptional cases such as overflow, underflow or empty heap. We will not use the test cases for those scenarios.

c. Example Input & Output

Input	Output
`[(‘insert’,5),(‘insert’,-3),(‘insert’,2)]`	`-3 5 2`
`[(‘insert’,4),(‘insert’,-2),(‘insert’,9),(‘insert’,10),(‘insert’,15),(‘insert’,-25)]`	`-25 4 -2 10 15 9`
`[(‘insert’,28),(‘insert’,9),(‘insert’,27),(‘insert’,10),(‘insert’,3),(‘insert’,45)]`	`3 9 27 28 10 45`

d. Example execution

```
>> ./pa2.exe 5 “[(‘insert’,5),(‘insert’,3),(‘insert’,2)]”
[Task 5]
2 5 3
```

6. Min-heap Deletion (3 pts)

a. Implement a function that **deletes** the minimum value or the indexed value from the binary min-heap. Your heap should maintain the min heap property even after the deletion.

b. Input & Output
Input: A sequence of commands, which is one of the following
- `(‘insert’,integer)`: insert `integer` into the current min heap
- `(‘delMin’,NULL)`: delete minimum value from current binary min heap and rearrange heap to maintain the min heap property.
- `(‘delete’,index)`: delete the value indexed by the given index from current binary min heap and rearrange heap to maintain the min heap property.

Output:
- Values in a heap in a node number order, in a string separated with the white space (automatically printed with built-in function)
- Do not consider the exceptional cases such as overflow, underflow or empty heap. We will not use the test cases for those scenarios.

c. Example Input & Output

Input	Output
`[(‘insert’,5),(‘insert’,-3),(‘insert’,22),(‘delMin’,NULL)]`	`5 22`
`[(‘insert’,28),(‘insert’,9),(‘insert’,27),(‘insert’,10),(‘insert’,3),(‘insert’,45),(‘delMin’,NULL)]`	`9 10 27 28 45`
`[(‘insert’,28),(‘insert’,9),(‘insert’,27),(‘insert’,10),(‘insert’,3),(‘insert’,45),(‘delMin’,NULL),(‘insert’,22)]`	`9 10 22 28 45 27`
`[(‘insert’,3),(‘insert’,7),(‘insert’,6),(‘insert’,25),(‘delMin’,NULL),(‘insert’,12),(‘delete’,2),(‘insert’,-1),(‘delMin’,NULL)]`	`6 7 12`

d. Example execution

```
>> ./pa2.exe 6 “[(‘insert’,4),(‘insert’,-2),(‘insert’,9),
(‘insert’,10),(‘insert’,15),(‘insert’,-25),(‘delMin’,NULL)]”
[Task 6]
-2 4 9 10 15
```