Name: ID: Exam Room:

INTERNATIONAL SCHOOL OF ENGINEERING

FACULTY OF ENGINEERING

CHULALONGKORN UNIVERSITY

2190221/2147105 Fundamental Data Structure and Algorithm / DATA STRUC ALG

Second Semester, Final Examination May 18, 2023. Time 08:30-11.30

Important

1. This exam paper has 3 questions. There are 5 pages in total including this page. The total mark is 30.
2. **All coding questions asked you to write Java code. JUnit is used to test your code. Your code must compile and run in order to get marked.**
3. When the exam finishes, students must stop and remain in their seats until the examiners allow students to leave the exam room.
4. A student must sit at his/her desk for at least 45 minutes.
5. A student who wants to leave the exam room early (must follow (5).) must raise his/her hand and wait for the examiner to check his/her machine. The student must do this in a quiet manner.
6. **Books, lecture notes, written notes, files are allowed.**
7. **The use of internet is not allowed.**
8. **All files must be loaded into the machine you use for exam.**
9. A student must not borrow any item from another student in the exam room. If you want to borrow an item, ask the examiner to do it for you.
10. Do not take any part of the exam file out of the exam room. All exam questions are properties of the government of Thailand. Violators of this rule will be prosecuted in a criminal court.
11. A student who violates the rules will be considered as a cheater and will be punished by the following rule:
    - **With implicit evidence or showing intention for cheating, student will receive an F in that subject and will receive an academic suspension for 1 semester.**
    - **With explicit evidence for cheating, student will receive an F in that subject and will receive an academic suspension for 1 year.**

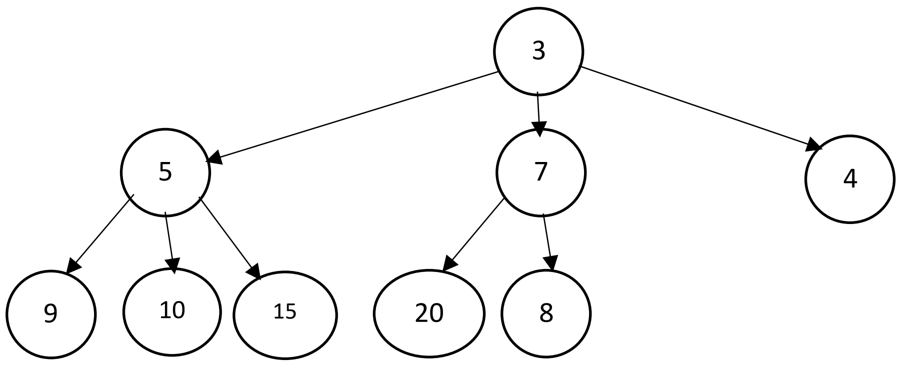
I acknowledge all instructions above. This exam represents **only my own work**. I did not give or receive help on this exam.

Student's signature(.......................................)

- READ AN ENTIRE QUESTION BEFORE STARTING TO WRITE ANYTHING FOR THAT QUESTION!!!
- In Eclipse, create a workspace called "2022_Datas_{seat no.}_{studentID}" where {studentID} is your student id number. For example, 2022_Datas_07_6532112221
- For each question, create a new Eclipse Project (project name is Q1,Q2,Q3,etc. according to the question) in the workspace.
- Download 2022_DataS_Final.zip from assignment in MyCourseville.
- YOU MUST Unzip the file into a folder.!!!!!!!!!! Use 7zip.
- For each question, copy that question's files into your project src folder.
- Follow the instruction of each question.
- At the end of the exam, leave Eclipse on (minimize the screen).

Name: ID: Exam Room:

1. (11 marks) A ternary min heap is a min heap where each node has at most 3 children. All properties are the same as a normal min heap. Its complete tree is filled from left to right at each level. It also has an array representation. An example is shown below:



The array representation is:

3	5	7	4	9	10	15	20	8

You are given the code for class Heap, a binary min heap (implemented using array). **Write a class TernaryHeap**:

- The class TernaryHeap has all the variables and methods from class Heap.
- Make modifications to all necessary places so that Ternary heap works.
- Test cases are in TestHeap.java. The score for each test is in the code's comment.
- Hint: For position i
    - the position of its parent node is (i-1)/3
    - the position of its child nodes are 3*i+1, 3*i+2, 3*i+3

2. (11 marks) A Treap is a type of binary search tree. Each of its node is defined as follows:

```
public class TreapNode {
    int bstValue; // value stored in the node.
    int heapValue; //priority value as in a min heap

    TreapNode left; //pointer to lower left node.
    TreapNode right; //pointer to lower right node.
    TreapNode parent; //pointer to the node above.
```

Name: ID: Exam Room:

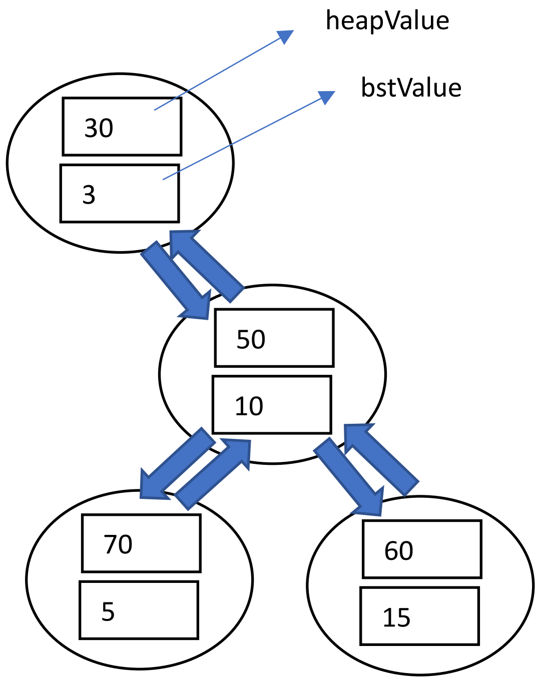
Assume that

- All (bstValue, heapValue) pair are unique.
- There can only be one heapValue for one bstValue.
- Treap does not store duplicated (bstValue,heapValue) pair.

Treap has the following properties:

- A Treap must be a binary search tree according to the bstValue in every node.
- From top (root) to bottom, the node is arranged according to heapValue (smaller value is nearer to the root).
- Treap is not a complete binary tree.

An example Treap looks like:



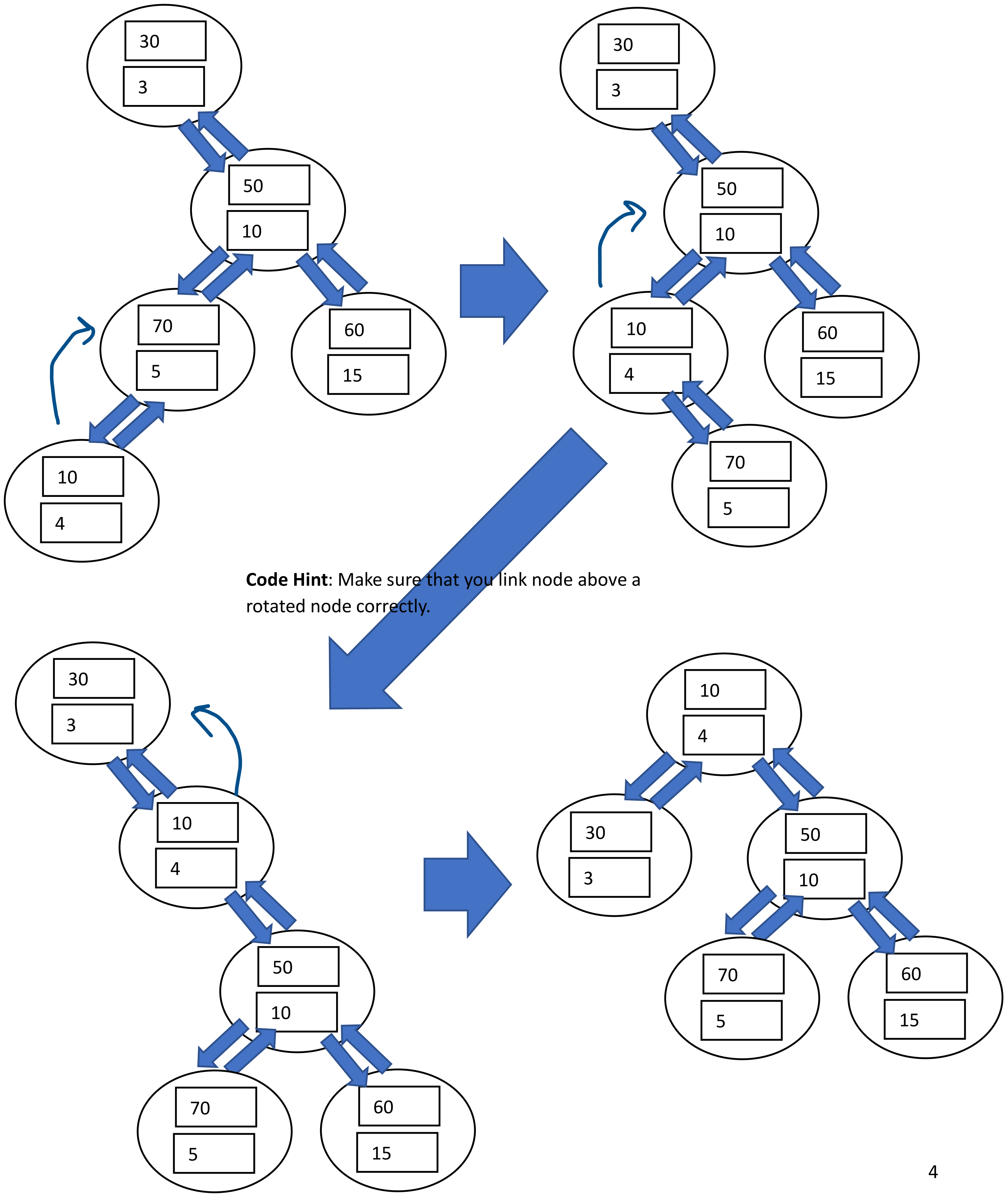
You are given codes for Binary Search Tree and AVL Tree (just 1 class for AVL tree), TreapNode, and a test file, TestTreap (scores for test are shown in code's comment).

Write class Treap (Copy and Modify codes from given classes as necessary). For methods, you only need to write these methods:

- Constructor (no parameter).
- `public TreapNode insert(int v, int h) //v is bstValue, h is healValue`
    - if v,h is stored inside the tree, do nothing and return **null** from the method.
    - if v,h is not stored inside the tree, insert a new node that contains these values.
    - Use heapValue of the newly added node to **rotate it up the tree**, such that heapValues from root to any leaf are sorted from small to large.
    - Then return the node that we added. (It may no longer be a leaf node)

Name: ID: Exam Room:

For example, adding node with (bstValue,heapValue) = (4,10) to the above tree will get us:



Code Hint: Make sure that you link node above a rotated node correctly.

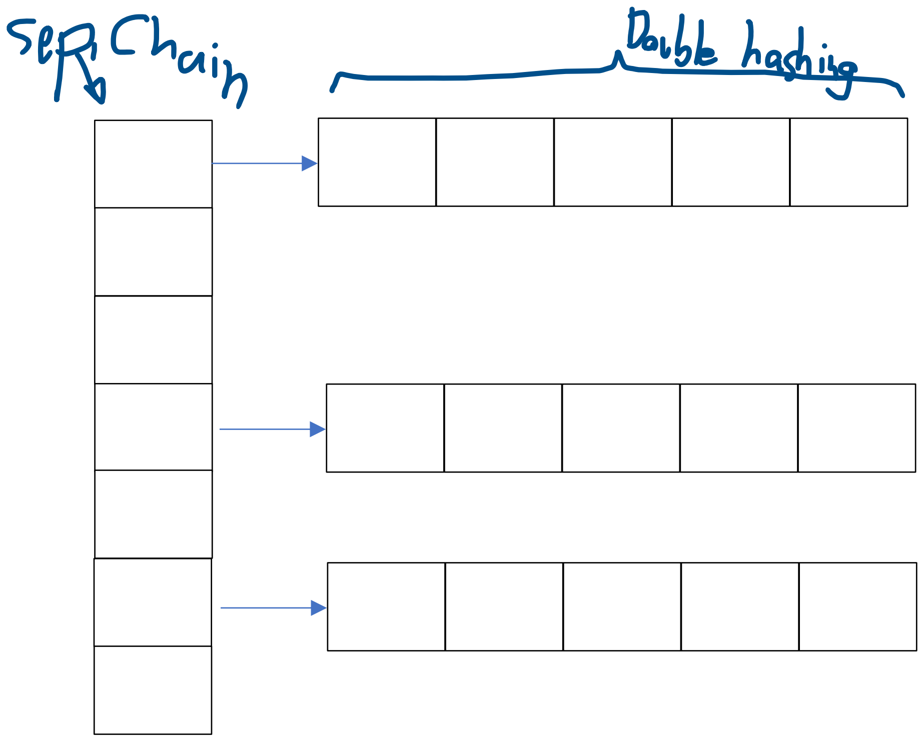
Name: ID: Exam Room:

3. (8 marks)

You have to fill in the table in this question! Then put this file in project folder Q3 in your workspace.

Alternatively, you can draw in a given picture (you can use Paint) and put the picture file in project folder Q3 in your workspace.

- A separate chaining hash table for integer (hash function hash(x) = x%tableSize) has 7 slots.
- In each slot, it stores a double hashing hash table (with 5 slots) (hash function h(x) = x % tableSize(), f(i) = i* h2(x), where h2(x) = 3 – (x%3)).
- For simplicity, the hash tables in this question are never rehashed.
- For data addition, no duplicated data is allowed.
- For deletion, use lazy deletion, and the Deleted slot can be reused in future addition.
- Fill in what the table look like eventually, if the following takes place in sequence:
    - Add 5, 26, 12, 40
    - Delete 12, delete 5
    - Add 47, 12


Sep Chain
Double hashing