National University of Singapore

School of Computing

Semester 1 (2011/2012)

CS2010 - Data Structures and Algorithms II

Quiz 1 (15%)

Saturday, September 17, 2011, 10.00-12.00pm (2 hours)

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this question paper until you are told to do so.
2. Quiz 1 is run on two large rooms in NUS, at the same time:
   Students with full name starting with 'A' until 'L' will go to LT19 (55 students)
   Students with full name starting with 'M' until 'Z' will go to COM1/SR1 (51 students)
3. This question paper contains FIVE (5) sections with sub-questions.
   It comprises TWELVE (12) printed pages, including this page.
4. Write all your answers in this question paper, **but only in the space provided**.
   You can use either pen or pencil. Just make sure that you write **legibly**!
   Important tips: Pace yourself! Do **not** spend too much time on one (hard) question.
5. This is an **Open Book Examination**. You can check the lecture notes, tutorial files, problem set files (**you have to bring printed version of ps1.pdf and ps2.pdf**), Steven's 'Competitive Programming 2' book, or any other books that you think will be useful. But remember that the more time that you spend flipping through your files implies that you have less time to actually answering the questions.
6. When this Quiz 1 starts, **please immediately write your Matriculation Number here:** ______________________ **(do not forget the last letter) and your Tutorial Group:** ______
   (if you forget your tutorial group number, just write down your Tutorial TA name).
   But *do not* write your name in order to facilitate unbiased grading.
7. All the best :).
   After Quiz 1, this question paper will be collected, graded manually in 2 weeks over recess week, and likely returned to you via your Tutorial TA on Week07.

–This page is intentionally left blank. You can use it as ‘rough paper’–

1 'ADT Table': (Questions → Answers) (20 marks)

The answers for this set of questions can be found in the lecture notes, tutorial files, or PS files. Can you find them *as fast as you can*? It is $O(1)$ if you already have the answer in your memory. Please fill in your answers on the space provided, 2 marks per question.
Grading scheme: 0 (zero correct answer), 1 (one correct answer), 2 (two correct answers).

1. There are several *linear* data structures. One of them is Array (or Vector/resizeable Array). Please mention at least two more: __________ and __________.
2. There are several *non-linear* data structures. Please mention at least two: __________ and __________.
3. There are four basic problem solving paradigms mentioned in Lecture 1. They are: Complete Search, ______________, Greedy, and ______________.
4. A tree is called a Binary Search Tree (BST) if each node has at most ________ children and each node satisfies the BST property, which is: ______________________.
5. We can get the *smallest* element in a BST (with ≥ 2 unique elements) by calling: __________, and we can get the *second smallest* element in the same BST by calling: ______________.
6. AVL stands for __________________ and it is an instance of a ________ BST.
7. In AVL tree, a *node* is said to be *height-balanced* if ____________________________ and a *BST* is said to be *height-balanced* if: ____________________________.
8. To be able to do `rotateRight` on node `X`, you need to ensure that `X` has ___________ and such rotation costs $O(\_\_\_\_)$.
9. What is the *Min*-heap[1] property? Answer: ____________________________. Because of this property, a *maximum* element in a min heap is located at: __________.
10. A complete binary tree can be stored in a *1-based* compact array. To navigate this 'tree', we can use arithmetic computation of array indices. To get the left children of index `i`, we compute index `2 × i`. To get the right children of index `i`, we compute index __________. To get the parent of index `i`, we compute index __________.

[1] Min heap has the same operations as max heap, just that all min heap operations enforce min heap property rather than max heap property.

2 Basic BST/AVL/Min (or Max) Heap Operations

Grading scheme: 0 (no answer), 1 (the final answer is totally wrong), 2 (the final answer has minor mistake(s)), 5 (the final answer is correct).

Q1. Binary Search Tree - Insertion (5 marks)

You already have the following BST:

```
   7
  / \
 3   10
       \
        15
```

You want to insert *five more* integers: {5, 4, 25, 20, 21} into the BST above one by one, in that order. Please draw the *final* resulting BST below:

Q2. AVL Tree - Insertion (5 marks)

You already have the following AVL tree (this is similar as Q1 above):

```
   7
  / \
 3   10
       \
        15
```

Now, insert the same *five* integers: {5, 4, 25, 20, 21} into the AVL tree above one by one, in that order. Please draw the *final* resulting AVL tree below:

Q3. Binary Search Tree - Deletion (5 marks)

You already have the following BST:

```
        10
      /    \
     4      11
    / \       \
   1   5       15
    \          /
     2        13
             /  \
            12  14
```

Delete 2, delete 11, delete 13, delete 14, and then delete 10, in that order. If we delete a node `X` with two children, we will select its *successor* to replace `X`. Please draw the *final* resulting BST below:

Q4. Min Heap - Extract Minimum Element (5 marks)

You already have the following *min* heap:

```
        3
      /   \
     4     6
    / \   / \
   8   5 7   10
  / \
 9   15
```

You want to know what are the three smallest elements in this min heap (obviously {3, 4, 5}) by calling `heapExtractMin()` *three times*. Please draw the *final* resulting min heap below:

Q5. Min Heap - Insertion (5 marks)

You already have the following *min* heap (this is similar as Q4 above):

```
       3
     /   \
    4     6
   / \   / \
  8   5 7   10
 / \
9   15
```

You want to insert *three more* integers: {-11, 1, -2} into the min heap above one by one, in that order. Please draw the *final* resulting min heap below:

Q6. Min Heap - Building Min Heap version 2 (5 marks)

Build a *min* heap from this set of *ten* integers: {13, 2, -3, 1, 5, 7, -8, 9, 20, -15} using the $O(n)$ algorithm (version 2) as mentioned in Lecture 4. Please write the *final* resulting min heap *1-based compact array* below (as suggested in Lecture 4, you can leave index 0 blank):

Index	0	1	2	3	4	5	6	7	8	9	10
Value	N/A										

3 PS++ (28 marks)

PSes are the core of CS2010 as students will probably spend more than the three recommended hours per week to deal with these problems. These group of questions are variants of the existing PSes. Can you use the knowledge that you have gained when you solved these PSes to solve the new variants?

3.1 PS1++, please refer to `ps1.pdf` that you have printed (18 marks)

Suppose that the meaning of method `query` in PS1 (see below):
`int Query(String START, String END, int genderPreference)`
is *altered* by Steven so that you now have to query your data structure and *display* (on top of reporting the number of) baby names that start with a prefix that is inside the query interval `[START..END)` *in sorted order*. The meaning of `genderPreference` is unchanged.

So, using the same examples as in `ps1.pdf`, `Query(``PET'', ``STE'', 1)` will still return 2 and *also* prints "PETER" and then "ROBERT", in that sorted order, to standard output (usually computer screen). `Query(``PET'', ``STE'', 2)` will return 0 but does not print anything to standard output. And so on...

Please come up with the *best possible* solution to handle this new requirement and analyze its time complexity per query. You just need to outline your ideas (use pseudo codes) to answer this question. Note: You are only given a small amount of space below (i.e. do **not** write too long-winded answer)!

3.2 PS2++, please refer to `ps2.pdf` that you have printed (10 marks)

Suppose Steven *forbids*[2] the usage of *any form* of heap data structure (including `Java PriorityQueue`) to solve PS2, can you use *another way* to implement the four requested operations: `ArriveAtHospital`, `UpdateStatus`, `GiveBirth`, and `Query` that still achieve at worst $O(\log n)$ time complexity[3] for each of the four operations?

Please come up with the *best possible* solution to handle this new requirement yet all operations are $O(\log n)$. You just need to outline your ideas (use pseudo codes) to answer this question.
Note 1: You are only given a small amount of space below (i.e. do **not** write too long-winded answer)!
Note 2: If you have spent more than 10 minutes in this question but still clueless, please move on...

[2] Students: "A PS on heap data structure and you forbid us to use it? :O" Steven: "Yes, can you solve this variant?"
[3] As in `ps2.pdf`, n is the maximum number of women that can be in the delivery suite at the same time.

4 Analysis (15 marks)

Prove (the statement is correct) or disprove (the statement is wrong) the following statements below.
If you want to prove it, provide the proof (preferred) or at least a convincing argument.
If you want to disprove it, provide at least one counter example.
Three marks per each statement below (1 mark for saying correct/wrong, 2 marks for explanation):
Note: You are only given a small amount of space below (i.e. do **not** write too long-winded answer)!

1. Given n numbers, we can sort them in $O(n)$ by inserting them into a BST (either balanced or not) and just call $O(n)$ `inorder()` traversal.

2. The `inorder()` traversal of a BST (either balanced or not) has been shown to produces sorted output and runs in $O(n)$. Steven claims that "calling `findMin()` followed by $n-1$ calls of `successor()`" on a BST (either balanced or not) also produces sorted output and *can be made* to run in total time of $O(n)$ instead of $O(n \log n)$.

3. We can always search for an item in any BST variant in $O(\log n)$ thanks to BST property.

4. The *second* largest element in a max heap with more than two unique elements is always one of the children of the root.

5. We can delete an element of a max heap located at index `i` in $O(\log n)$ time.
   The max heap is stored in a 1-based compact array of size n and $1 \leq i \leq n$.

5 A New Calculator for 7-years old Jane (7 + 3 more bonus marks)

Imagine that today is year 2018. Jane is ‘now’ a 7-years old girl. Recently, Steven gave her a calculator that can display n digits as a birthday present for her.

Jane likes the calculator as it helps her to count faster. One day, she entered a number k (where $0 \le k < 10^n$) and then repeatedly *squares* it (e.g. $k = k^2$ or $k = k \times k$) until the result overflows (remember that the calculator can only display n digits). When the result overflows, only the n most *significant* digits are displayed on the screen and an error flag appears. There is a button in the calculator that Jane can use to clear the error. She can then continue squaring the displayed number, observing various different numbers, until she realizes something...

Being a clever girl with a good memory, she realizes that there is no point continuing her button mashing process if a certain number that she has seen before *reappears* (do you understand why?). As a daughter of a computer scientist, she wonders: “Given n and k, what is *the largest number* that I can see using this button mashing process?”.

Example 1: If the calculator can only display $n = 1$ digit and $k = 6$, then Jane see:

Actual Value of k	Seen On Calculator Screen	Largest Number Seen by Jane So Far
6	6	6
$6 \times 6 = \underline{3}6$	3	6
$3 \times 3 = 9$	9	$\underline{9}$ (the answer)
$9 \times 9 = \underline{8}1$	8	9
$8 \times 8 = \underline{6}4$	6 (repeated)	9

At this point, Jane realizes that number 6 reappears and there is no point continuing.
The largest number seen by Jane in her calculator so far is 9. This is the answer.

Example 2: If the calculator can only display $n = 2$ digits and $k = 2$, then Jane see:

Actual Value of k	Seen On Calculator Screen	Largest Number Seen by Jane So Far
2	2	2
$2 \times 2 = 4$	4	4
$4 \times 4 = 16$	16	16
$16 \times 16 = \underline{25}6$	25	25
$25 \times 25 = \underline{62}5$	62	$\underline{62}$ (the answer)
$62 \times 62 = \underline{38}44$	38	62
$38 \times 38 = \underline{14}44$	14	62
$14 \times 14 = \underline{19}6$	19	62
$19 \times 19 = \underline{36}1$	36	62
$36 \times 36 = \underline{12}96$	12	62
$12 \times 12 = \underline{14}4$	14 (repeated)	62

At this point, Jane realizes that number 14 reappears and there is no point continuing.
The largest number seen by Jane in her calculator so far is 62. This is the answer.

There are different possible solutions for this non-original problem (the problem source will be revealed after Quiz 1). All possible solutions just require CS2010 knowledge taught so far (except the bonus marks). Your solution will be awarded different marks based on the criteria below:

Maximum Marks	Requirement (on Steven's machine, 10 MB of memory, 1 second runtime)
3	Able to solve test cases with $1 \le n \le 3$
5	Able to solve test cases with $1 \le n \le 6$
7	Able to solve test cases with $1 \le n \le 9$
$7 + 3 = 10$	Able to solve test cases with $1 \le n \le 9$ *without* using BST/AVL/Heap

Please come up with the *best possible* solution and analyze its time complexity.
You just need to outline your ideas (use pseudo codes) to answer this question.
Note 1: You are only given a small amount of space below (i.e. do **not** write too long-winded answer)!
Note 2: Do **not** attempt this question unless you have completed the other questions.

– End of this Paper –

Candidates, please do not touch this table!

Question	Maximum Marks	Student's Marks
1	20	
2	30	
3	28	
4	15	
5	$7 + 3 = 10$	
Total	103	