THE HONG KONG UNIVERSITY OF SCIENCE & TECHNOLOGY
Department of Computer Science

COMP 171 Data Structures and Algorithms
Spring 2005

Midterm Examination

Date: Saturday, April 2, 2005 Time: 14:30–16:30

- **You must attempt all questions. Your answers will be graded according to correctness, efficiency, precision and clarity.**
- **This is a closed book, closed notes examination. You should put aside your calculators, cellphones, PDAs, Palms, etc.**
- **This document has 12 single-sided pages, please check it.**
- **You may use the reverse side of the pages for your rough work.**
- **You cannot write your answers using a pencil, although you may use a pencil for rough work. Rough work will not be marked.**

Student name: _____ **Student ID:** _____

Email address: _____ **Lecture section:** _____

Question	Score	Maximum score
1		8
2		10
3		11
4		6
5		15
TOTAL		50

1. (8 marks) **Heaps and binary search tree.**

   (a) (1 mark) Does the following array represent a **min-heap** (Yes/No)? _____

0	1	2	3	4	5	6	7	8
7	9	8	20	16	14	10	17	18

   (b) (3 marks) Given the following **max-heap** of size 10:

0	1	2	3	4	5	6	7	8	9
29	11	17	10	11	16	15	9	7	10

   Delete the maximum from the max-heap and give the resulting max heap:

0	1	2	3	4	5	6	7	8

   (c) (4 marks)
   The following two sequences of keys (separated by blanks) are the output of the inorder and postorder traversals of a binary search tree. Draw this tree.

   Inorder: `B C D E G H`
   Postorder: `D E C H G B`

2. (10 marks) **Sorting**

Consider sorting an unordered array $A[0 \ldots n-1]$ of distinct integers using mergesort. The following algorithm is the merging routine used in mergesort.

Algorithm *merge*(A, p, q, r)
Input: Subarrays $A[p..l]$ and $A[q..r]$ s.t. $p \leq l = q - 1 < r$ and both subarrays are sorted in increasing order.
Output: $A[p..r]$ is sorted in increasing order.
($* \, T$ is a temporary array. $*$)

```
1.  k = p; i = 0; l = q − 1;
2.  while p ≤ l and q ≤ r
3.      do if A[p] ≤ A[q]
4.             then T[i] = A[p]; i = i + 1; p = p + 1;
5.             else  T[i] = A[q]; i = i + 1; q = q + 1;
6.  while p ≤ l
7.      do T[i] = A[p]; i = i + 1; p = p + 1;
8.  while q ≤ r
9.      do T[i] = A[q]; i = i + 1; q = q + 1;
10. for i = k to r
11.     do A[i] = T[i − k];
```

(a) (3 marks) Analyze the worst-case running time of the algorithm *merge* in terms of the lengths of $A[p..l]$ and $A[q..r]$.

(b) (1 mark) Explain why *mergesort* may run slower than *quicksort* when the input array size is large.

(c) (6 marks) Write down the pseudocode for a non-recursive version of *mergesort*. You may assume that the input size is a power of 2. Analyze the worst-case running time of your non-recursive mergesort using big Oh. (Show all steps.)

(cont’d)

3. (11 marks) **Recursion**
   Consider the following recursive program:

```
// x is an integer and n is any non-negative integer
int foo( int x, int n ){

  if ( n == 0 ) return 1;

  if ( n%2 == 1 ) {
     y = foo(x,(n-1)/2);
     return x*y*y;
   }
  else{
     y = foo(x,n/2);
     return y*y;
  }

}
```

   (a) (3 marks) What is the output for `foo(2, 7)`?

   (b) (3 marks) What does the program do?

(c) (5 marks) Analyze the worst-case running time of the program using big Oh, assuming that n is a power of 2. (Show all steps.)

4. (6 marks) **Stacks**
   You are given an input line of left parentheses '`(`' and right parentheses '`)`'. Your task is to check whether the parentheses are balanced, i.e., all left and right parenthesis can be matched and they nest properly. Write a pseudo-code for performing this check with the help of a stack. You can assume that the input is non-empty and that you can detect the end of line. For example, `(()())` and `())((` are valid and invalid inputs respectively.

5. (15 marks) **Binary search tree**
   This question concerns a non-empty binary search tree T. Each node in T stores an integer key and left and right child pointers. A node does NOT store a parent pointer or any other information. For each internal node v, all the keys in the left (resp. right) subtree of v is less than (resp. greater than) the key stored at v.

   (a) (5 marks) Consider the deletion of a key x from T. You can assume x exists in T. Describe the three cases that may arise and the deletion strategy in each case. It is not necessary to provide the algorithmic details.

(b) (10 marks) Describe an algorithm to output the median key in T. That is, if there are n keys in T, you are to output the $\lceil n/2 \rceil$th smallest key. Your algorithm must satisfy the following requirements:

- The input provides a pointer to the root of T and the input contains nothing else.
- The tree T and its nodes cannot be modified. Refer to the preamble of the question for what information is stored at the nodes.
- Local variables are allowed, but you cannot use any array, vector, linked list, or other large pointer structures as intermediate storage.
- If you design some functions and call them, all parameters must be passed by value. Passing parameters by reference is not allowed. This constraint also applies when some functions are designed and called implicitly in the description of your algorithm.
- You are allowed to use functions that return values.

(cont’d)