# BU CAS CS112 B1: Introduction to CS II
# Midterm Examination II

Instructor: Hongwei Xi

Time: 9:00-11:00AM  Room: MCS B29
Date: Tuesday, the 8th of May, 2007

Name:_____ Score:_____

| No. | Points | Score |
|---|---|---|
| 01. | 05 | |
| 02. | 15 | |
| 03. | 25 | |
| 04. | 20 | |
| 05. | 15 | |
| 06. | 10 | |
| 07. | 10 | |
| 08. | 10 | |
| 09. | 15 | |
| Total | 125 | |

# No computer is allowed!

- The exam is *closed* book and notes!
- Please read each question carefully before answering it.
- Your score for this exam is computed as follows:

  $$(\langle\text{your total points}\rangle/125) * 1.25$$

  So it is possible for you to score more than 100!

GOOD LUCK!

**Question 1** *(5 points) Please draw a binary search tree of height* 3 *that contains the following list of numbers:*

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9$$

**Question 2** *(15 points) Please implement the root insertion operation on (integer) binary search trees.*

```
class Node {
  int item; Node l; Node r;
  Node (int x, Node nl, Node nr) { item = x; l = nl; r = nr; }
};

static Node insertT (Node h, int i) {
  // you may assume the availability of rotL (left rotation) and rotR (right rotation)
  // your implementation:









}
```

**Question 3** *(25 points)*

- *(5 points) Linear-probing is an approach used in implementing hash tables. Please state clearly and concisely what linear-probing is about.*

- *(20 points) Please implement the removal operation for a linear-probing hash table.*

```
public void remove (Item x) {
  int i = hash (x.key (), M);  // M is the size of the hash table
  // you may assume the availability of the insertion method
  // your implementation is here:









}
```

**Question 4** *(20 points) Please complete the following implementation of the quicksort algorithm (on integer arrays):*

```
static void exch (int A[], int i, int j) {
  int t = A[i]; A[i] = A[j]; A[j] = t;
}

static void quicksort(int A[], int l, int r) {
  if (r <= l) return;
  int i = partition(a, l, r);
  quicksort(a, l, i-1);
  quicksort(a, i+1, r);
}

static int partition (int A[], int l, int r) {
   // please use A[r] as the pivot for partitioning
   // your implemenation:









}
```

**Question 5** *(15 points) Please describe and implement insertion sort.*

- *(5 points) Please state clearly and concisely what insertion sort is. Is it stable?*

- *(10 points)*

```
static void compExch (int A[], int i, int j) {
  int t = A[i];
  if (A[j] < t) { A[i] = A[j]; A[j] = t; }
  return;
}

static void insert_sort(int A[], int l, int r) {
  int i;
  for (i = r; i > l; i--) compExch (a, i-1, i);
  // your implementation:









}
```

**Question 6** *(10 points) Suppose that the edges of a given undirected graph $G$ are ordered as follows according to the weights attached to these edges:*

`3-7 1-4 3-9 1-9 7-8 0-5 5-2 3-8 2-9 0-6 4-9 2-6 6-4`

*where* `3-7` *and* `6-4` *have the minimal and the maximal weights, respectively. Please briefly describe how Prim's algorithm works and then use it to find all the edges in a minimal spanning tree of $G$ (by assuming that the vertex 3 is the source).*

**Question 7** *(10 points) Suppose that the edges of a given undirected graph $G$ are ordered as follows according to the weights attached to these edges:*

`3-7 1-4 3-9 1-9 7-8 0-5 5-2 3-8 2-9 0-6 4-9 2-6 6-4`

*where* `3-7` *and* `6-4` *have the maximal and the minimal weights, respectively. Please briefly describe how Kruskal's algorithm works and then use it to find all the edges in a minimal spanning tree of $G$.*

**Question 8** *(10 points) A data type for ternary trees can be given as follows:*

```
class Tree3 {
  int item; Tree3 left; Tree3 middle; Tree3 right;
  Tree3 (int i, Tree3 l, Tree3 m, Tree3 r) {
    item = i; left = l; middle = m; right = r;
  }
} ;
```

*Please implement the level-order traversal on ternary trees. That is all the nodes at one level should be visited before any nodes at the next level can be visited. You can assume the availability of a queue (for storing nodes).*

```
static void traverse (Tree3 t) { // level-order traversal
   Queue Q = new Queue();
   Q.put (t);
   while (!(Q.isEmpty())) {
   // your implementation:









  }
}
```

**Question 9** *(15 points) Suppose that a and b are two binary search trees. Please implement a function joinR that joins the two binary search trees a and b into a single binary search tree.*

```
class Node {
  int item; Node l, r;
  Node(int x) { item = x; l = null; r = null; }
} ;

static Node joinR (Node a, Node b) {
  // you can assume the availability of root insertion
  // your implementation









}
```