National University of Computer and Emerging Sciences, Lahore

CS201 **Data Structures**

Final Exam, Thu, 20 Dec 2012, Time: 3 hours, Maximum Marks: 50

Handwriting must be legible. Clearly label figures. Clearly state assumptions made, if any.

Write line numbers next to every line of code and briefly explain it using those line numbers.

1. A binary search tree has 3000 nodes.
   (a) What is the maximum possible height of this tree?[2]
   (b) What is the minimum possible height of this tree?[3]
   (c) What is the maximum possible number of leaf nodes in this tree?[5]

2. A linked list is a linear structure. The `next` field of every cell points to the cell that comes after it. If the `next` field of a cell starts pointing to a previous cell, the linked list is said to have a loop. Because of the presence of a loop, a function traversing the list goes in an infinite loop, and never terminates.
   (a) Describe an algorithm to detect loop in a linked list using LIST data structure.[5]
   (b) Write function `detect_loop()` implementing the above algorithm. It returns 0 if no loop is detected and 1 if a loop is detected. The function prototype and cell structure definition is given below.[5]

```
struct cell {
   int data;
   struct cell *next;
};

int detect_loop (struct cell *head);
```

3. Write a function to merge two sorted linked-lists. Assume that the lists are already sorted in a non-decreasing order. Function prototype is given below.

```
struct cell *merge_lists (struct cell *l1, struct cell *l2);
```

   where the structure of each cell of both linked lists is defined as follows.

```
struct cell {
  int data;
  struct cell *next;
};
```

   The merged list must also be in non-decreasing order. Your code must not traverse each of the two lists more than once, must not allocate any additional memory, and must not modify `data` field of any cell.[10]

4. Starting with an empty AVL tree insert the following keys in the order shown:

   GRUMPY, SLEEPY, HAPPY, DOC, DOPEY, BASHFUL, SNEEZY

   Show the complete state of the tree after adding each key. Describe what kind of rotation, if any, had to be performed to keep the tree balanced.[10]

5. Write a recursive function to find the height of a binary tree. Cell structure definition and function prototype are given below.[10]

```
struct cell {
   int data;
   struct cell *lst, *rst;
};

int height (struct cell *tree);
```

Good Luck!