**CS 201 DATA STRUCTURES**

**[**Friday, 16th November, 2012.] [FAST, NU LAHORE CAMPUS] **[Time**: 1.5 Hours ]

[Midterm exam 2] **SOLUTIONS** [**Total Marks**: 50]

**QUESTION 1 (Marks 15)**

Suppose we have singly, sorted, linear, linked lists with a head pointer (class given below). Given two such sorted linked lists, construct a third linked list which is also sorted and contains all the nodes of the original two lists. Do not create/allocate any new nodes for third list, use the nodes of the two given lists.

(Note: If you use any other method, then provide its implementation too.)

The following declarations are provided:

You do not have to write the code for the implemented methods.

|  |  |
| --- | --- |
| class Node  {  Node \*next;  int Data;  friend class LinkedList;  }; | class LinkedList  {  public:  LinkedList(); //implemented  ~LinkedList(); //implemented  LinkedList(const LinkedList &L); //implemented  const LinkedList & operator = (const LinkedList &rhs); //implemented  **bool Merge(LinkedList &L1, LinkedList &L2); //to be implemented**  **//add any other method you like but implement it too**  private:  Node \*head;  }; |
|  |  |

You have to implement the merge function. An example is given below:

|  |  |
| --- | --- |
| L1 is the following list: | L2 is the following list: |

L3.merge(L1, L2) should construct L3 as:

After merging roots of L1 & L2 will be pointing towards NULL.

**QUESTION 2 (Marks 15)**

Write a recursive function that determines whether the binary tree is BST or not.

**QUESTION 3 (Marks 5+5)**

Given the following binary tree

Perform the following deletion using successors where required.

|  |  |
| --- | --- |
| **PART A:** Delete the key 40 from the original tree shown above and draw it | **PART B:** Delete the key 10 from the original tree shown above and draw it |

**QUESTION 4 (Marks 10)**

Given the following 2-3 tree

Insert 45.