Indian Statistical Institute
Semestral Examination 2009-2010
B.Math II Year I Semester
Computer Science I (Programming in C)
Part C (**Theory**)

Date: **November 27, 2009** Time: **2 Hours** Total Marks: **30**

Answer any five questions. All questions carry equal marks. Marks are shown to the left of each question.

Q1:

[1+1] (a) Convert the expression ((A + B) * C – (D – E) / (F + G)) to equivalent **prefix** and **postfix** notations.
[4] (b) Construct the binary tree for the following **preorder** and **inorder** traversals.

Preorder : A B C D E F G H I J K L M N O P Q
Inorder : D E C F B H G A K M L J I O N Q P

Q2:

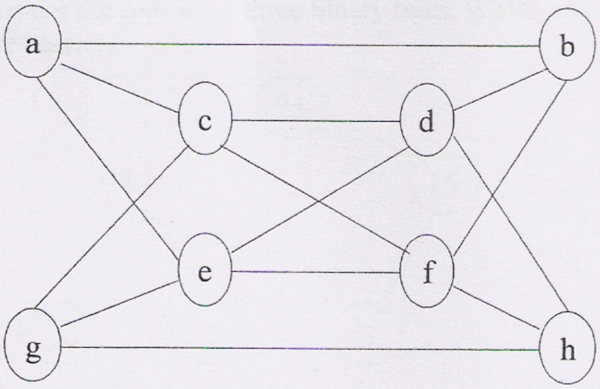
[1] (a) What is B-tree?
[3] (b) Insert the keys bellow, in the order stated, into an initially empty B-tree of order 5.

1, 7, 6, 2, 11, 4, 8, 13, 10, 5, 19, 9, 18, 24, 3, 12, 14, 20, 21, 16

[2] (c) Show the B-tree after deleting key **4** from the B-tree you obtained in Q2. (b) above.

Q3:

[2] (a) Explain the difference between **depth-first** and **breadth-first** traversal of a graph.
[2+2] (b) List the order in which the nodes of the undirected graph shown bellow are visited by a i) **breadth-first traversal** that starts from vertex **a**, and ii) **depth-first traversal** that starts from vertex **d**.



Q4:

[2+1] (a) What is **AVL tree**? How is an **AVL tree** different from normal **binary tree**?
[2+1] (b) Derive a recursive function of N(**h**), where N(**h**) denotes the minimum number of nodes of an **AVL tree** of height **h**. What is the minimum number of nodes in an AVL tree of hight **5**?

Q5:

[1+2] (a) What is **hashing** in data structure? Describe the method of **chaining** to resolve collisions in hash table.

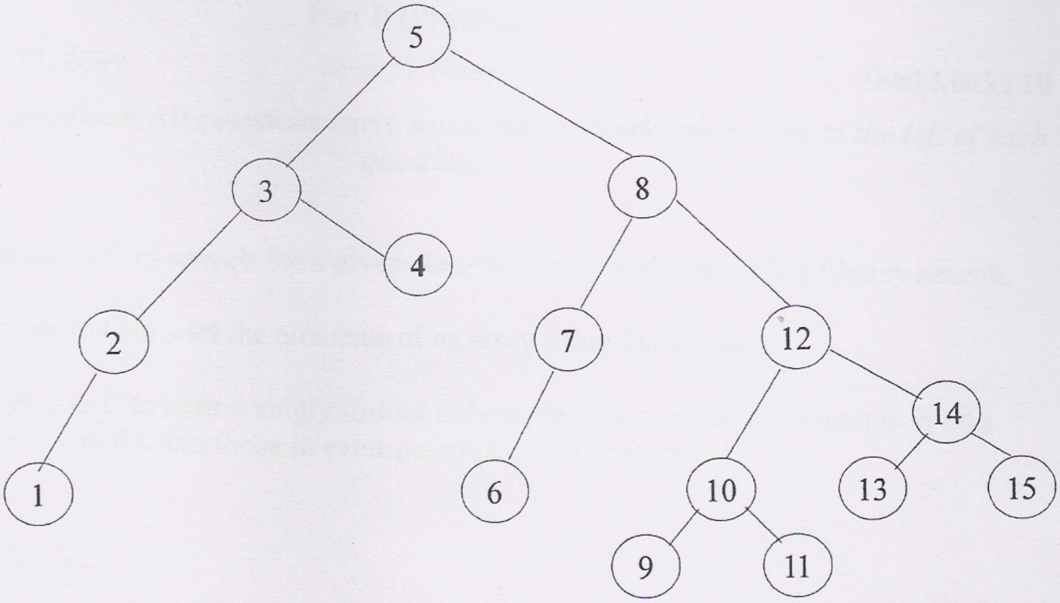
[2+1] (b) Suppose that a hash table contains HASHSIZE = 13 entries indexed from 0 through 12 and that the following keys are to be mapped into the hash table: 10, 100, 32, 45, 58, 126, 3, 29, 200, 400, 0. Determine the number of collisions occur when these keys are mapped into the hash table by using hash function H(k) = k % HASHSIZE. Apply the method of chaining to resolve these collisions.

Q6:

[3] (a) Insert the keys bellow, in the order stated, into an initially empty **AVL tree**.

11, 13, 21, 20, 22, 16, 24

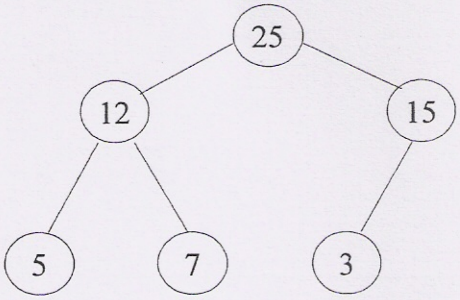
[3] (b) Show the AVL tree after deleting key **4** from the following **AVL tree**.



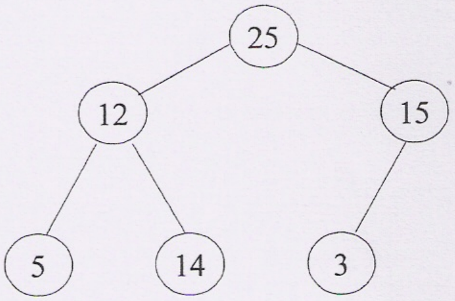
Q7:

[1+2] (a) What is a **heap**? Determine the minimum and maximum number of nodes that a heap of hight **h** can have?

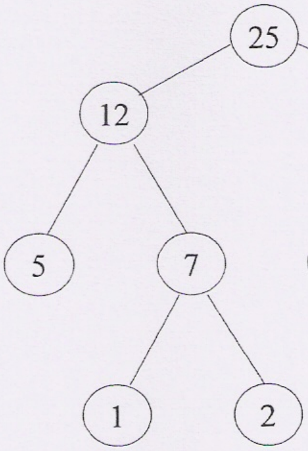
[3] (b) Consider the following three binary trees. Which one of these trees is a heap? For those that are not, give reasons.



(i)



(ii)



(iii)

Indian Statistical Institute
Semestral Examination 2009-2010
B.Math II Year I Semester
Computer Science I (Programming in C)

Part B (**Practical**)

Date : **November 27, 2009** Time: **1 Hour** Total Marks:**10**

Answer any two questions. All questions carry equal marks. Marks are shown to the left of each question.

[5] 1. Write a program in C to **search** for a given element in a **sorted** array using **binary search**.

[5] 2. Write a program in C to **sort** the elements of an array using **Quick Sort**.

[5] 3. Write a program in C to **split** a singly **linked list** into two lists so that all elements in **odd position** are in one list and those in **even position** are in another list.

Indian Statistical Institute
Semestral Examination 2009-2010
B.Math II Year I Semester
Computer Science I (Programming in C)

Part A (**Home Assignment**)

Date : November 13, 2009

Total Marks:**10**

All questions are compulsory and carries equal marks.

1. Write a program in C to convert an **infix** notation expresion to its equivalent **RPN** (Reverse Polish Notation) expresion using **stack**.

2. Write a program in C to implement the **insertion** and **deletion** operations in a **circular queue**.

3. Write a program in C to calculate the **factorial** of a given **large number** N (100 <= N <= 200) using **linked list**.

4. Write a program in C to insert a node into an AVL tree.

5. Write a program in C to combine **two** arrays that are already sorted into **one** sorted array.