Department of Computer Science

CS 201 – Data Structures

Final Term (Summer 2014)

**Instructors:** Adnan Waheed

*August 9, 2014*

|  |  |
| --- | --- |
| **Total Marks: 40** | **Time Allowed: 180 minutes** |

**Instructions:**

1. Understanding the question is part of exam. NO QUERIES WILL BE ENTERTAINED.
2. Use answer sheet for rough work and provide solutions in the given space.
3. Write neat & clean.
4. Use permanent ink pens only.
5. Poor programming approaches will decrease your marks.
6. Think about the boundary conditions.

**Roll No. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Section: \_\_\_\_\_\_\_\_\_**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Question No.** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **Total** |
| **Marks** | ***5*** | ***6*** | ***6*** | ***6*** | ***4*** | ***3*** | ***6*** | ***13*** | ***8*** | ***10*** | ***6*** | ***6*** | ***6*** | ***3*** | ***88*** |

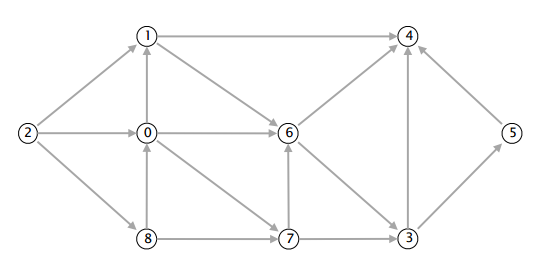
**GOOD LUCK ☺**

|  |  |
| --- | --- |
| **Question 1:** | **Marks 2.5+2.5** |

Consider the following acyclic digraph. Assume the adjacency lists are in sorted order: for

example, when iterating through the edges pointing from 0, consider the edge 0 → 1 before

0 → 6 or 0 → 7.



Compute the topological order by running the DFS-based algorithm and listing the vertices

**2 8 0 7 1 6 3 5 4**

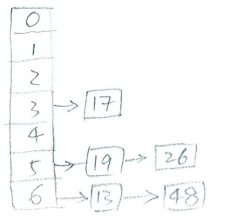
Run breadth-first search on the digraph, starting from vertex 2. List the vertices in the order in which they are dequeued from the FIFO queue.

**2 0 1 8 6 7 4 3 5**

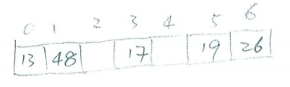
|  |  |
| --- | --- |
| **Question 2:** | **Marks 2+2+2** |

Consider a hash table of size 7 starting at 0 with hash function h(k)=k mod 7. Draw the table that results after inserting, in the given order, the following values: 19, 26, 13, 48, 17 for each of the three scenarios below:

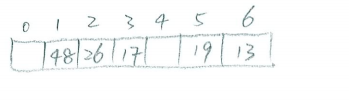
1. When collisions are handled by chaining



1. When collisions are handled by linear probing



1. When collisions are handled by re-hashing using a second hash function h'(k)=5-(k mod 5)



|  |  |
| --- | --- |
| **Question 3:** | **Marks 3+3** |

Consider the following AVL tree and solve following very neatly. Rough work on back side of the sheet.

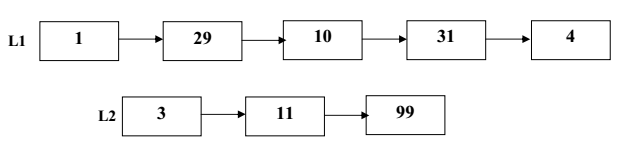
1. Insert the key 25 into the tree and re-balance if needed. Draw the final tree and all intermediate trees that you need. You must use the algorithms studied in class for inserting and re-balancing.

1. Remove the value 40 from the original tree and re-balance if needed. Draw the final tree and all intermediate trees that you need. You must use the algorithms studies in class for removing and re-balancing

sad

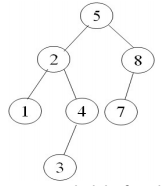
|  |  |
| --- | --- |
| **Question 4:** | **Marks 3+3** |

Suppose we have two list of integer as shown in figure below.



1. Write C++ code to merge the two lists together, however while merging L2 into L1 you have to follow the following rules. The first node of L2 should be inserted at head, the second node is inserted at the last of L1 and the third node of the list should be inserted exactly in the middle of the L1 and so on.
2. Write C++ code to delete the middle node of the link list, if the number of nodes in link list are even, you have to delete both the middle nodes and in case of odd number of nodes you have to delete the middle node.

|  |  |
| --- | --- |
| **Question 5:** | **Marks 4** |

Write C++ code to find the height of Binary Search Tree. The structure of the Tree is defined below for your reference.

struct BSTree

{

int data;

BSTree \*Left,

BSTree \*Right;

};

BSTree \*Root;

Example of Binary Search Tree is shown in figure above, your height function should return three for the above example. You have to write generic height function which works for all Binary Search Trees.

|  |  |
| --- | --- |
| **Question 6:** | **Marks 3** |

A hash function f is defined as f(key)= key mod 7 where linear probing is used to insert the keys 37,38,72,48,98,11,56 into a table indexed from 0 to 6. Show visually (no code required) the final hash table with above values inserted

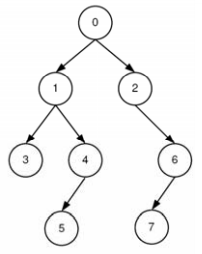
|  |  |
| --- | --- |
| **Question 7:** | **Marks 6** |

Create an AVL tree with following keys, clearly show visually each step. each step will contain a mark

12, 33, 44, 43, 99, 0, 88, 13, 43, 2, 222

|  |  |
| --- | --- |
| **Question 8:** | **Marks 3+3+3+4** |

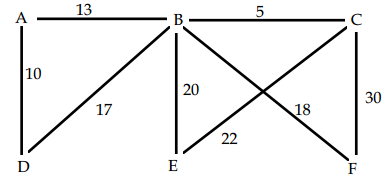
Following figure illustrates the structure of a company. Each node represents an employee in the company and stores information regarding his/her salary. Every arrow in the figure corresponds to a employee-manager relationship. You can assume that every node has a unique number(id), each manager can supervise at most two people, and each employee has only one manager.



1. Define a structure CP capable of representing a node of structure shown in figure above? You representation should cover all the attributes in statement otherwise you will get zero.
2. Write a recursive function *MaxSalary* that takes as input the pointer to node 0, and return the maximum salary in company
3. Write a recursive function that returns the sum of all salaries in the company. The function should take as input the pointer to node 0
4. Write a recursive function that checks if any of the employees in the company has higher salary from any of his managers. The function should return the value of 1 if there is such a case otherwise should return o. The function should take as input the pointer to node 0

|  |  |
| --- | --- |
| **Question 9:** | **Marks 4+4** |

Consider following graph



1. Perform a depth-first traversal of the graph shown above, starting with vertex C. Select the smallest edge first when appropriate. In the space below, list the vertices in the order in which they are visited.
2. Perform a breadth-first traversal of the graph shown above, starting with vertex C. Select the smallest edge first when appropriate. In the space below, list the vertices in the order in which they are visited.

|  |  |
| --- | --- |
| **Question 10:** | **Marks 10** |

Write a C++ method named mirror() that takes a reference to the root node of a binary tree and creates a new tree (with its own nodes) that is the mirror image of the original tree. For example: if root is a reference to the root of the tree on the left below, then the return value of mirror(root) would be a reference to the root of the tree on the right below.

***Hint***: This method is much easier to write if you use recursion.

Node mirror(Node root) {

}

|  |  |
| --- | --- |
| **Question 11:** | **Marks 6** |

List the output of the following program segment. If an error is contained in the program, please mark and explain the error. For the rest of the program, assume that all lines with an error are removed from the program.

#include <iostream>

using namespace std;

class Node

{

public: double data;

public: Node \*next;

};

int main(void)

{

double \*px, \*py;

double x, y;

Node \*p, \*q, \*r;

x = 9.5;

y = -2.0;

// px = 8.6;//error

//\*px = 10.0;//error

// \*x = \*px;//error

// x = \*px;//error

py = &y;

p = new Node(); q = new Node(); r = new Node();

p->data = 1.0;

q->data = 2.0;

r->data = 3.0;

p->next = q;

q->next = r;

r->next = p;

// cout << \*px << endl;// error

cout << x << endl;// 9.5

cout << \*py << endl;//-2

cout << q->next<<endl;//address

cout << p->next->next->next->next->data << endl;//2

delete q;

// cout << p->next->next->next->next->data << endl;// error

system("pause");

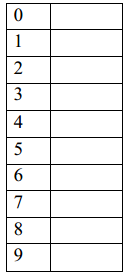
return 0;

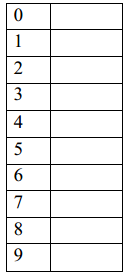
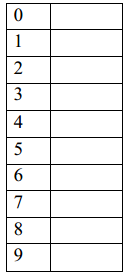
}

|  |  |
| --- | --- |
| **Question 12:** | **Marks 2+2+2** |

For each of the following versions of hash tables, insert the following elements in this order: 55, 86, 16, 25, 6, 7. For each table, TableSize = 10, and you should use the primary hash function h(k) = h%10. For each table the first column holds the indices and the second can hold values or pointers, depending on the hash-table type. If an item cannot be inserted into the table, please indicate this and continue inserting the remaining values.

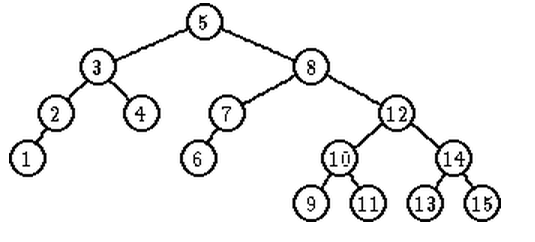
1. Chaining hash table – use a linked list for each bucket, insert at front



1. Linear probing hash table: c) Quadratic probing hash table 

|  |  |
| --- | --- |
| **Question 13:** | **Marks 3+3** |

Consider the following AVL tree.



Show the modified tree under each of the following operations. (Note: The two operations are independent. Each of them starts from the above tree.)

1. Deletion of the key 4
2. Insertion of the key 16.

|  |  |
| --- | --- |
| **Question 14:** | **Marks 3** |

Write the following infix expression as a prefix expression:

(((a-g) + (e / b)) \* c)