

Algorithms and Data Structures

Midterm 2

Notes

1. Tear off this sheet and use it to **keep your answers covered at all times**.
2. Turn the exam over and write your name next to the staple. Do **not** put your name anywhere else on the exam.
3. To receive full credit, show **all** your work.
4. You are graded not only on the correct answer, but also the most efficient answer.
5. Read all the questions before you begin.
6. The use of calculators is **not** allowed.
7. If you have questions about the test, please come and ask me.
8. No C++ code is meant to have syntax errors.
9. All code you write will be written in C++.
10. Use the following code for the appropriate question.

```

class EdgeNode
{ public:
  int fromNode;
  int toNode;
  int weight ;
  EdgeNode * next ;
  EdgeNode(int from, int to, int wt){
    toNode = to; fromNode = from;
    weight = wt; next = NULL;}
};

class GraphNode
{ public:
  int Name; // Node name
  EdgeNode * succs; //successor list
  GraphNode( int n){
    Name = n; succs = NULL;}
};

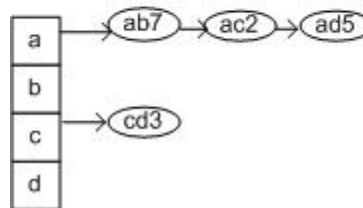
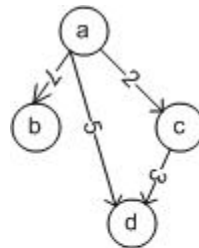
```

```

const int MAX = 26;
class Graph
{ protected:
  GraphNode G[MAX]; // Array of nodes
  int nodeCt; // Number of actual nodes
used
public:
  Graph();
};

```

For Example, for the following graph:



Multiple choice – 4 points each.

1. In the code below for all pairs shortest path, what is the purpose of the “if” statement?

```
for (k = 0; k < N; k++)  
  for (i=0; i < N; i++)  
    if path[i,k] != INF  
      for (j=0; j < N; j++)  
        path[i,j] = min(path[i,j], path[i,k]+path[k,j])
```

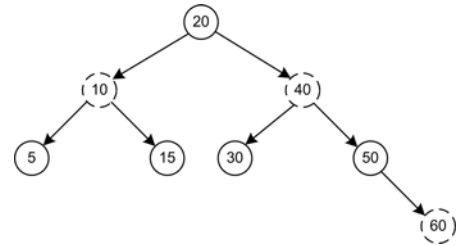
- a. If the if statement were removed, the algorithm would be incorrect.
 - b. The if statement allows you to ignore redundant paths.
 - c. The if statement reduces the run time, but does not effect the correctness.
 - d. None of the above.
2. Why is lazy deletion used to delete from a hash table?
- a. To ensure efficient insertion
 - b. To prevent the search routine from assuming an item is not there when it really is there.
 - c. To reduce the load factor
 - d. All of the above
 - e. None of the above
3. In double hashing, each key has a personalized step value it uses to find the next probe location. What makes us think that by adding a value repeatedly (mod table size) we will eventually try all positions in the hash table?
- a. All positions will not be hit, but that isn't important
 - b. The fact that the table size is prime makes this more likely
 - c. The personalized step value is chosen to guarantee this
 - d. If all positions aren't tried, the method is inappropriate
 - e. None of the above
4. You have a hash table of size N with less than N elements in the table. In trying to place a new key, collision resolution claims it has made N probes yet it has not found a free location. Which best explains how this could happen?
- a. You have primary clustering
 - b. You have secondary clustering
 - c. Every unused position is marked as deleted
 - d. The collision resolution algorithm is in a cyclic state of retrying the same few locations repeatedly
 - e. None of the above
5. Finding a node, in an AVL tree, has what complexity?
- a. $O(n \log n)$
 - b. $O(n)$
 - c. $O(\log n)$
 - d. $O(1)$
 - e. None of the above

6. An AVL tree of height 5 (root is height 1) has at least how many nodes?

- a. 10
- b. 12
- c. 14
- d. 31
- e. None of the above

7. In the red-black tree (red nodes are dashes) to the right, what is true of the tree after 45 is inserted using top down insertion?

- a. 40 would be black.
- b. 60 would be a child of 40
- c. 45 would be a child of 40
- d. A left rotation at 20 is required.
- e. None of the above



8. Which is true regarding the relationship between an AA tree and a red-black tree?

- a. They have basically the same advantages and disadvantages
- b. The AA tree is simpler because it has fewer special cases that need to be coded
- c. An AA tree is better because its height is smaller
- d. All of the above
- e. None of the above

Short Answer

9. (15 points) Explain how to delete from an AA tree. Show examples of interesting cases.

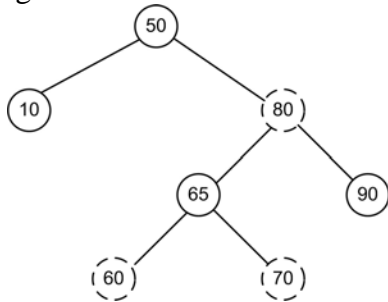
10. (20 points) Consider the following hash table of size 15. Use double hashing with
 $\text{step}(\text{key}) = 1 + \text{key} \% (\text{tablesize} - 2)$
 $\text{hash}(\text{key}) = \text{key} \% \text{tablesize};$

A -1 in the table indicates the cell is unused.

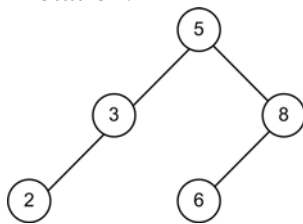
- a. Since the table size isn't prime, show a case that is problematic.
- b. Show the work needed to delete 9.
- c. Show the work needed to insert 21 (assuming 9 has been deleted)

	Key
0	15
1	46
2	-1
3	-1
4	34
5	20
6	96
7	-1
8	53
9	9
10	76
11	-1
12	87
13	28
14	-1

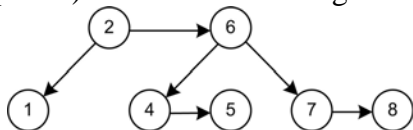
11. (7 points) In the red-black tree shown below (red nodes are dashes), insert 62. Draw the tree every time colors change or rotations are applied. Label each picture so it is clear what is going on.



12. (7 points) Given the following AVL tree, insert the value **1**, doing any necessary rotations to maintain the AVL property. Draw and label (with the name of the rotation) the tree after each rotation.



13. (7 points) Given the following AA-tree. Show the resulting AA tree when 10 is inserted.



20. (20 points) Using the graph structure from the cover page, write the code to print source and sink nodes of the graph. A source node in a graph has no predecessors. A sink node has no successors. Assume the graph has been read correctly into memory.