Name:

Email ID:

Section:

CSE 326, Data Structures

Sample Final Exam

Instructions:

- The exam is closed book, closed notes.
- Unless otherwise stated, *N* denotes the number of elements in the data structure under consideration.
- Answer each problem in the space provided.
- Show your work to ensure partial credit.

Time: 110 minutes

Problem	Max Points	Score
1	14 (2x7)	
2	18 (3x6)	
3	4	
4	7	
5	9	
6	16	
7	8	
8	4	
9	8	
10	4	
Total	92	

1) **[14 points total, 2 points each] True/False**. Circle True or False below. You do *not* need to justify your answers.

a	Linear probing is equivalent to double hashing with a secondary hash function of $h_2(k) = 1$.	**True**	**False**
b.	If $T_1(N) = O(f(n))$ and $T_2(N) = O(f(n))$, then $T_1(N) = O(T_2(N))$.	**True**	**False**
c.	Building a heap from an array of N items requires $\Omega(N \log N)$ time.	**True**	**False**
d.	Dijkstra's algorithm for shortest path and Prim's minimum spanning tree algorithm have the same big-Oh worst case running time.	**True**	**False**
e.	Both Prim's and Kruskal's minimum spanning tree algorithms work correctly if the graph contains negative edge weights.	**True**	**False**
f.	For large input sizes, mergesort will always run faster than insertion sort (on the same input).	**True**	**False**
g.	Merging heaps is faster with binary heaps than with leftist or skew heaps, because we only need to concatenate the heap arrays and then run BuildHeap on the result.	**True**	**False**

2) **[18 points total] Short Answer Problems:** *Be sure to answer all parts of the question!!*

a) [3 points] Which ADT do binomial queues implement? If we forget simplicity of implementation, are they better than binary heaps? Are they better than leftist heaps? Justify your answer.

b) [3 points] You could use an AVL tree to do a sort. Describe how you would do this. What is the worst-case running time for your sort?

c) [3 points] What is the main advantage that open addressing hashing techniques have over separate chaining?

d) [3 points] Suppose we choose the median of five items as the pivot in quicksort. If we have an N element array, then we find the median of the elements located at the following positions: left (= 0), right (= N – 1), center (the average of left and right, rounded down), leftOfCenter (the average of left and center, rounded down), and rightOfCenter (the average of right and center, rounded down). The median of these elements is the pivot.

What is the worst case running time of *this version* of quicksort?

e) [3 points] Kruskal's minimum spanning tree algorithm uses a heap to quickly find the lowest cost edge not already chosen. What would be the running time of the algorithm if instead of using a heap, the algorithm used a simple unsorted array to store the edges?

f) [3 points] Fill in the blank in the following text to make the statement true.

In the union/find data structure of N items, if we use union-by-size *without path compression*, then any combination of M union and/or find operations takes at most __________________ time.

3) **[4 points total] Minimum spanning tree (MST)**
Given a weighted, undirected graph with |V| nodes, answer the following as best as possible, with a brief explanation. Assume all weights are non-negative.

a) [2 points] If each edge has weight $\leq w$, what can you say about the cost of an MST?

b) [2 points] If the cost of an MST is c, what can you say about the shortest distances returned by Dijkstra's algorithm when run with an arbitrary vertex s as the source?

4) **[7 points total] Disjoint Sets:**

Consider the set of initially unrelated elements 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12.

a.) (5 pts) Draw the final forest of up-trees that results from the following sequence of operations using on union-by-size. Break ties by keeping the first argument as the root.

Union(0,2), Union(3,4), Union(9,7), Union(9,3), Union (6,8), Union(6,0), Union(12,6), Union(1,11), Union(9,6).

b.) (2 pts) Draw the new forest of up-trees that results from doing a Find(4) with path compression on your forest of up-trees from (a).

5) [9 points total] Heaps

a) [4 points] Draw the binary min heap that results from inserting: 77, 22, 9, 68, 16, 34, 13, 8 in that order into an initially empty binary min heap. You do not need to show the array representation of the heap. You are only required to show the final heap, although if you draw intermediate heaps, ***please circle your final result for ANY credit***.

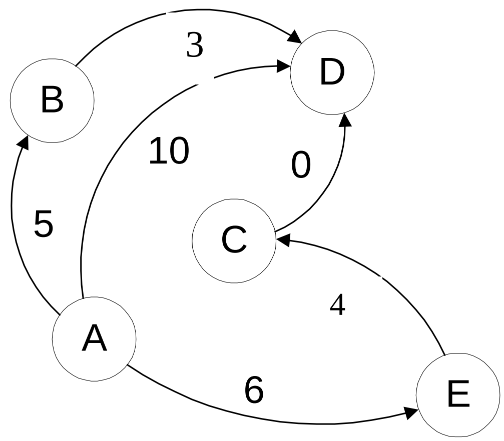
b) [2 points] Draw the binary min heap that results from doing 2 deletemins on the heap you created in part a). You are only required to show the final heap, although if you draw intermediate heaps **please circle your final result for ANY credit**.

c) [1 point] What is the null path length of the root node in the last heap you drew in part b) above?

d) [2 points] List 2 good reasons why you might choose to use a skew heap rather than a leftist heap.

6) **[16 points total] Graph Manipulations:**

Use the following graph for this problem. **Where needed and not determined by the algorithm, assume that any algorithm begins at node A.**



a) (4 pts) Draw both the adjacency matrix and adjacency list representations of this graph. Be sure to specify which is which.

b) (2 pts) Give two valid topological orderings of the nodes in the graph.

c) (4 pts) Step through Dijkstra's Algorithm to calculate the single source shortest path from A to every other vertex. You only need to show your final table, but you should show your steps in the table below for partial credit. Show your steps by crossing through values that are replaced by a new value. *Note that the next question asks you to recall what order vertices were declared known*.

Vertex	**Known**	**Distance**	**Path**
A			
B			
C			
D			
E			

d) (1 pts) In what order would Dijkstra's algorithm mark each node as *known***?**

e) (1 pt) What is the shortest (weighted) path from A to D?

f) (1 pt) What is the length (weighted cost) of the shortest path you listed in part (e)?

g) (3 pts) Imagine that the graph were undirected (i.e., ignore the directions of the edges). Write the edges considered by Kruskal's algorithm in the order they are considered. Assume the algorithm terminates as soon as the MST has been completed. Write an edge between vertices A and B as (A,B).

7) [8 points] Splay trees

Imagine that the following operations are performed on an initially empty splay tree:

Insert(1), Insert(10), Insert (5), Insert (3), Insert (7), Insert (13), Find (3).

Show the state of the splay tree after performing each of the above operations. Be sure to label each of your trees with what operations you have just completed.

8) **[4 points]** Draw the contents of the hash table in the boxes below given the following conditions:

The size of the hash table is 12.
Open addressing and quadratic probing is used to resolve collisions.
The hash function used is H(k) = k mod 12

What values will be in the hash table after the following sequence of insertions? Draw the values in the boxes below, and show your work for partial credit.

33, 10, 9, 13, 12, 45

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	

9) [8 points total] Memory

a) [2pts] Define spatial locality.

b) [2pts] Give an example of *spatial* locality in a program. Give an example and indicate what will have spatial locality and why.

c) [2pts] Define temporal locality.

d) [2pts] Give an example of *temporal* locality in a program. Give an example and indicate what will have temporal locality and why.

10) [4 points] B-trees

Given the following parameters:

Disk access time = 1milli-sec per byte

1 Page on disk = 1024 bytes

Key = 16 bytes

Pointer = 4 bytes

Data = 128 bytes per record (includes key)

(4 pts) What are the best values for:

M =

L =