NAME: **login ID:**
precept #:

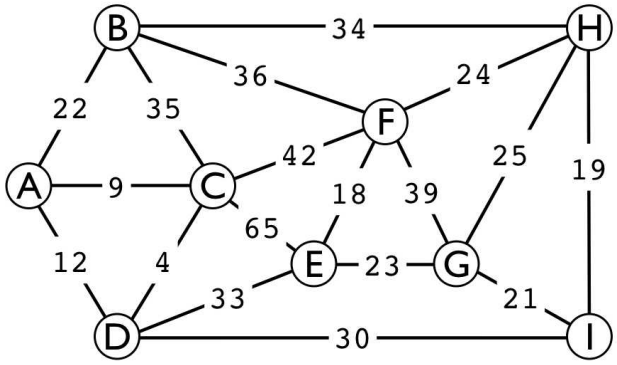
COS 226 Final Exam, Spring 2009

This test is 16 questions, weighted as indicated. The exam is closed book, except that you are allowed to use a one page cheatsheet. No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided. *Put your name, login ID, and precept number on this page* (*now*), and write out and sign the Honor Code pledge before turning in the test. You have *three hours* to complete the test.

"I pledge my honor that I have not violated the Honor Code during this examination."

1	/8		
2	/5		
3	/5		
4	/8		
5	/4		
		Subtotal	/30
6	/6		
7	/6		
8	/7		
9	/7		
10	/4		
		Subtotal	/30
11	/14		
12	/6		
13	/3		
14	/4		
15	/4		
16	/9		
		Subtotal	/40
TOTAL	/100		

1. **MST** (8 points). Consider the following undirected weighted graph.



a. Give the list of edges in the minimum spanning tree in the order that *Kruskal's algorithm* inserts them. For reference, the 18 edge weights are listed here:

4 9 12 18 19 21 22 23 24 25 30 33 34 35 36 39 42 65

b. Give the list of edges in the minimum spanning tree in the order that *Prim's algorithm* inserts them, assuming that it starts at vertex F.

2. **KMP** (5 points). The following is a KMP state-transition table for a 9-character string.
a. Write the characters in the string in the blanks below the table.

	0	1	2	3	4	5	6	7	8
A	1	1	3	1	X	Y	1	1	9
B	0	2	0	4	0	Z	0	8	0
C	W	0	0	0	0	6	7	0	0

___ ___ ___ ___ ___ ___ ___ ___ ___

b. Fill in the values of W, X, Y, and Z in the blanks provided below.

W:___ X:___ Y:___ Z ___

3. **Mystery code** (5 points). Circle the choice that describes a use of the following code

```
for (int i = 1; i <= G.V(); i++)
   for (int v = 0; v < G.V(); v++)
      for (Edge e : G.adj(v))
      {
         int w = e.to();
         if (dist[w] > dist[v] + e.weight())
         {
            dist[w] = dist[v] + e.weight())
            pred[w] = e;
         }
      }
```

a. To find the longest cycle in a weighted digraph

b. To compute the MST of a weighted graph

c. To topologically sort a digraph

d. To find shortest paths in a weighted digraph

e. To implement DFS in a weighted graph

4. **Acronyms** (8 points). Match each of the given acronyms (at right) with a description (at left). Write the letter corresponding to your answer for each acronym in the space provided. In some cases, more than one answer might be argued, but you must put only one letter per space. If you pick the best matches, you will use all the letters.

a.	Abstract machine, basis for KMP algorithm.	_____	SAT
b.	General graph-searching scheme.	_____	KMP
c.	Substring-search algorithm.	_____	NFA
d.	Fundamental recursive method.	_____	DFA
e.	For single-source shortest paths in unweighted graphs.	_____	DFS
f.	Set of all problems checkable in polynomial time.	_____	BFS
g.	A logic problem.	_____	PFS
h.	Abstract machine, basis for grep	_____	NP

5. **LZW compression** (4 points). Which of the following best describes the length of the code produced by the LZW compression algorithm for a string consisting of N characters that are all the same ? Circle your answer.

   a. $\log N$

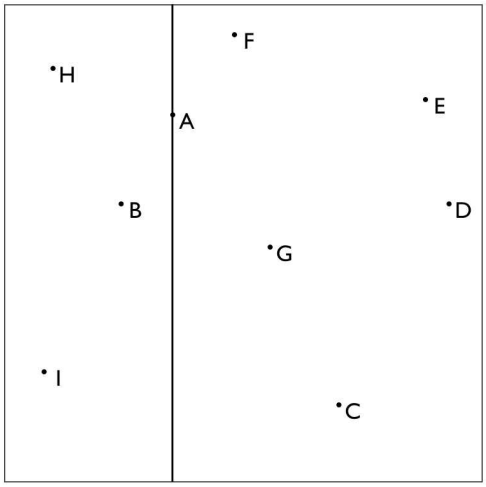
   b. $(\log N) * (\log N)$

   c. square root of N

   d. N

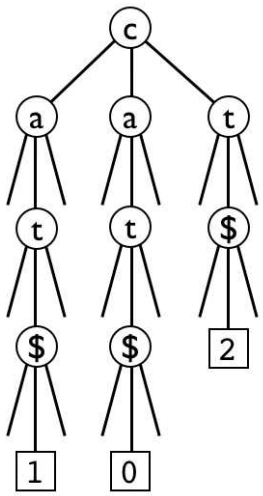
   e. $N \log N$

6. **2D trees** (6 points). Draw the partition of the plane that results from inserting the points drawn below `A B C D E F G H I` (in that order) into a 2D tree. The first partition (a vertical line) is drawn for you.



   Draw the tree here:

7. **Suffix TST** (6 points). The suffix TST corresponding to a string is constructed by building the TST corresponding to the string suffixes, including on each an end-of string character `$` that is larger than every string character. Thus, every path through the TST ends in a `$`, below which we put an external node corresponding to the starting point of the suffix. For example, the suffix TST corresponding to the string `cat` is drawn at left.



Draw the suffix TST corresponding to the string `ooops` .

8. **String symbol table** (7 points). Match each of the given search data structures with one or more of the given characteristics, where N is the number of keys, L is the average number of characters in a key, W is the length of the longest key, and M is the size of the alphabet. Write as many letters (in alphabetical order) as apply in the blank to the left of the name of the data structure. Assume that M is a constant.

   a. tree/trie shape is dependent on the order in which keys are inserted

   b. worst-case search time is proportional to W

   c. space usage (not including strings themselves) is not dependent on W

   d. worst-case search time is proportional to LN

   e. worst-case search time is proportional to WN

   _____ M-ary trie

   _____ BST

   _____ TST

   _____ red-black tree

9. **RE pattern matching I** (7 points). Draw an NFA (nondeterministic finite state automata) that recognizes the same language that the regular expression `((A*B | C )* | D )` describes. Use the notation and construction given in lecture. Circle your final answer.

10. **RE pattern matching II** (4 points). It is easy to build a NFA (nondeterministic finite state automata) corresponding to a regular expression, and a basic theorem from automata theory says that we can convert every NFA to a DFA (deterministic finite state automata). Why do we not do so to implement RE pattern matching? Circle one of the following choices.

    a. The DFA might have an exponential number of states.

    b. There might exist a string that the DFA recognizes but the NFA does not.

    c. There might exist a string that the NFA recognizes but the DFA does not.

    d. The NFA might loop.

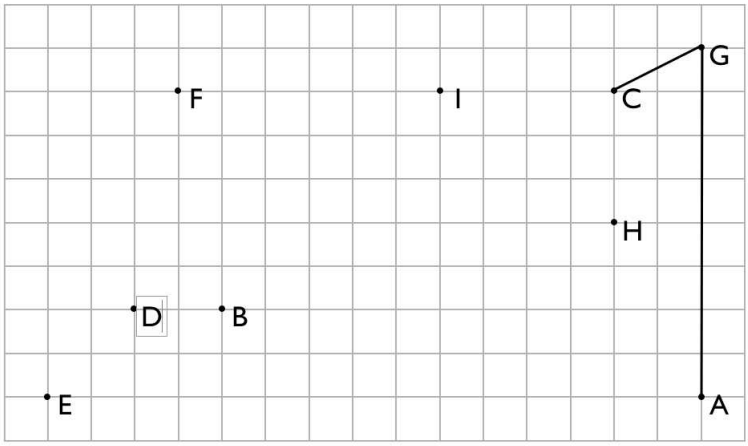
    e. The proof of the theorem is not constructive (does not tell us how to construct the DFA from the NFA).

11. **7 sorting algorithms** (14 points). The leftmost column is the original input of strings to be sorted, and the rightmost column is the sorted result. The other columns are the contents at some intermediate step during one of the 7 sorting algorithms listed below. Match up each algorithm by writing its letter under the corresponding column. Use each letter exactly once.

fuzz	benz	zeta	benz	cruz	cozy	cozy	benz	benz
fizz	czar	zinc	cozy	fizz	faze	fizz	cozy	cozy
cozy	cruz	maze	czar	cozy	faze	fuzz	czar	cruz
zinc	cozy	faze	cruz	fizz	czar	zinc	faze	czar
fizz	fizz	fuze	fuzz	fizz	benz	fizz	faze	faze
fuzz	faze	faze	fizz	faze	cruz	fizz	fizz	faze
fizz	fuzz	raze	fizz	fizz	faze	fizz	fizz	faze
fizz	fuze	fuze	fuzz	fizz	fizz	fuzz	fizz	fizz
maze	fuzz	haze	fizz	fuze	fizz	benz	fizz	fizz
faze	fuze	size	fizz	faze	fizz	czar	fuze	fizz
czar	fuze	fuze	faze	czar	fizz	faze	fuzz	fizz
benz	faze	faze	fuze	benz	fizz	maze	fuzz	fizz
fuze	fizz	zing	fuzz	fuze	fizz	faze	fuzz	fizz
fuzz	fuzz	zoom	faze	fuze	fuze	fizz	maze	fuze
faze	fizz	czar	fizz	faze	fuze	fuze	zinc	fuze
fizz	fizz	cozy	fuzz	fizz	fuze	fuzz	cruz	fuze
jazz	fuzz	lazy	fuze	fuzz	fuzz	cruz	fizz	fuzz
zoom	fizz	fuzz	fuze	fuzz	fuzz	jazz	fuzz	fuzz
cruz	fizz	fizz	faze	zoom	fuzz	zing	jazz	fuzz
zing	faze	fizz	fizz	zing	fuzz	zoom	raze	fuzz
fuzz	size	fuzz	haze	jazz	raze	fuze	zing	haze
raze	raze	fizz	jazz	raze	zing	fuzz	zoom	jazz
fuze	zing	fizz	lazy	fuzz	lazy	lazy	faze	lazy
lazy	zeta	benz	maze	lazy	haze	raze	fizz	maze
haze	zoom	fuzz	raze	haze	zeta	fuze	fuze	raze
zeta	jazz	fizz	size	zeta	size	haze	fuze	size
size	haze	jazz	zinc	size	zoom	size	haze	zeta
fuze	lazy	cruz	zoom	maze	jazz	zeta	lazy	zinc
faze	maze	fuzz	zing	fuzz	maze	faze	size	zing
fizz	zinc	fizz	zeta	zinc	zinc	fizz	zeta	zoom
	____	____	____	____	____	____	____	

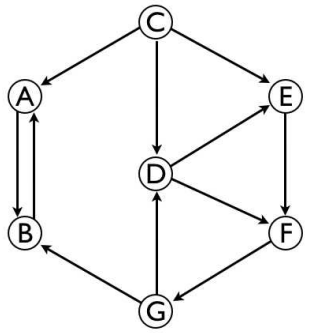
a. Mergesort
b. MSD radix sort
c. LSD radix sort
d. Quicksort (with no random shuffle)
e. Bottom-up mergesort
f. Quicksort with 3-way partitioning (with no random shuffle)
g. 3-way radix quicksort

12. **Convex hull** (6 points). Complete the trace given below for the Graham scan when used to find the convex hull of the following point set. For each point scanned, give the point followed by the current list of points on the trial hull after that point is scanned. The first three lines of the trace are filled in for you.



next point	*current hull*
A	A
G	A G
C	A G C
____	_____
____	_____
____	_____
____	_____
____	_____
____	_____

13. **Strong Components** (3 points). The following digraph has three strong components. List the vertices in each.



first component _____

second component _____

third component _____

14. **Reduction** (4 points). You are a manager at a large and successful internet company that was built on solving problem A in cubic time. One of your mathematicians has proven that problem A linear-reduces to problem B. Working independently, one programmer has implemented a linear-time solution to problem B and another programmer has implemented a quadratic-time solution to problem A. What should you do?

a. Fire the first programmer and promote the mathematician.

b. Fire the second programmer and promote the mathematician.

c. Promote the first programmer and fire the mathematician.

d. Promote them all.

e. Fire them all.

15. **Huffman encoding** (4 points). Draw a Huffman encoding trie for the message

```
S H E S E L L S S E A S H E L L S
```

(17 characters).

16. **Hard problem identification** (9 points). This question is in regard to your new job working for a software technology company. Your boss (having been told by you on the basis on your 126 knowledge that the company had better not bet its future on developing an application that finds an optimal tour connecting a set of cities) is still looking for a challenging project for you. Your boss is willing to invest in things that might be difficult, but not things that we know to be impossible or that we believe to be intractable. On the basis of your 226 knowledge, which of the following ideas can you tell your boss to forget about? Circle all that apply.

    A. A regular expression that describes palindromes

    B. An algorithm that guarantees to compress any given file by at least one percent

    C. A linear-time algorithm for finding the MST of a set of points in the plane that can only compare the distances between two points (not know their coordinate values)

    D. A linear-time algorithm for finding the MST of a graph with positive edge weights

    E. A linear-time algorithm for sorting an array of numbers that can only compare two numbers (not know their values)

    F. A fast poly-time algorithm for linear programming

    G. A fast poly-time algorithm for integer linear programming

    H. A sublinear expected-time (as a function of the total number of characters) algorithm for sorting random strings.

    I. A poly-time algorithm that finds the shortest path connecting two vertices in a digraph with edge weights that could be negative (but with no negative cycles).