COS 226	Algorithms and Data Structures	Fall 2011

Final

This test has 14 questions worth a total of 100 points. You have 180 minutes. The exam is closed book, except that you are allowed to use a one page cheatsheet (8.5-by-11, both sides, in your own handwriting). No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided. **Write out and sign the Honor Code pledge before turning in the test.**

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

Problem	Score
0	
1	
2	
3	
4	
5	
6	
Sub 1	

Problem	Score
7	
8	
9	
10	
11	
12	
13	
Sub 2	

Total	

Name:

Login ID:

Precept:

P01	11	Maia Ginsburg
P01A	11	Aman Dhesi
P02	12:30	Sasha Koruga
P02A	12:30	Joey Dodds
P03	1:30	Maia Ginsburg
P03A	1:30	Joey Dodds

0. **Miscellaneous. (1 point)**

   Write your name and Princeton NetID in the space provided on the front of the exam, and circle your precept number.

1. **Analysis of algorithms. (10 points)**

   (a) Suppose that you collect the following memory usage data for a program as a function of the input size N.

N	memory
1,000	10,000 bytes
8,000	320,000 bytes
64,000	10,240,000 bytes
512,000	327,680,000 bytes

   Estimate the memory usage of the program (in bytes) as a function of N and use tilde notation to simplify your answer.
   Hint: recall that $\log_b a = \lg a / \lg b$.

(b) For each function on the left, give the best matching order of growth of the *running time* on the right.

A. $\log N$

B. N

C. $N \log N$

D. N^2

E. 2^N

F. $N!$

__B__

```
public static int f1(int N) {
   int x = 0;
   for (int i = 0; i < N; i++)
      x++;
   return x;
}
```

_____

```
public static int f2(int N) {
   int x = 0;
   for (int i = 0; i < N; i++)
      for (int j = 0; j < i; j++)
         x++;
   return x;
}
```

_____

```
public static int f3(int N) {
   if (N == 0) return 1;
   int x = 0;
   for (int i = 0; i < N; i++)
      x += f3(N-1);
   return x;
}
```

_____

```
public static int f4(int N) {
   if (N == 0) return 0;
   return f4(N/2) + f1(N) + f4(N/2);
}
```

_____

```
public static int f5(int N) {
   int x = 0;
   for (int i = N; i > 0; i = i/2)
      x += f1(i);
   return x;
}
```

_____

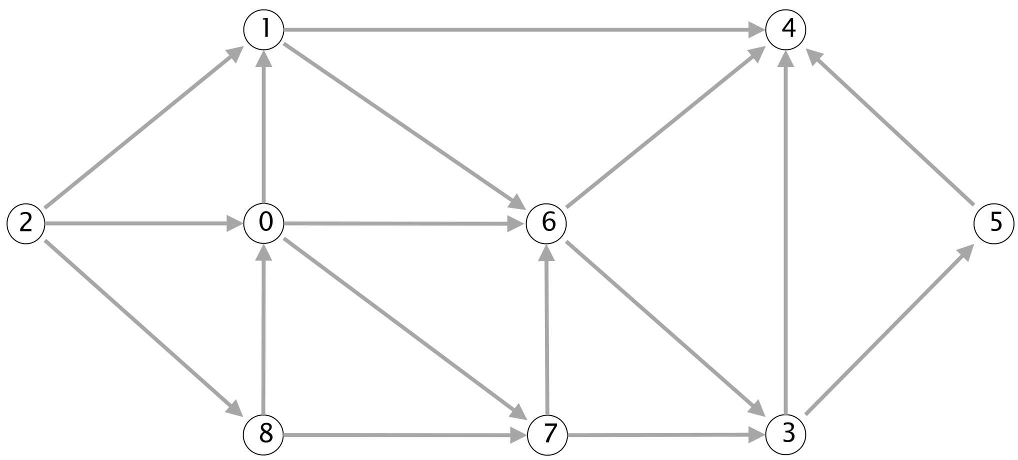
```
public static int f6(int N) {
   if (N == 0) return 1;
   return f6(N-1) + f6(N-1);
}
```

_____

```
public static int f7(int N) {
   if (N == 1) return 0;
   return 1 + f7(N/2);
}
```

2. **Graph search. (8 points)**

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 $0 \to 1$ before $0 \to 6$ or $0 \to 7$.



(a) Compute the topological order by running the DFS-based algorithm and listing the vertices in *reverse postorder*.

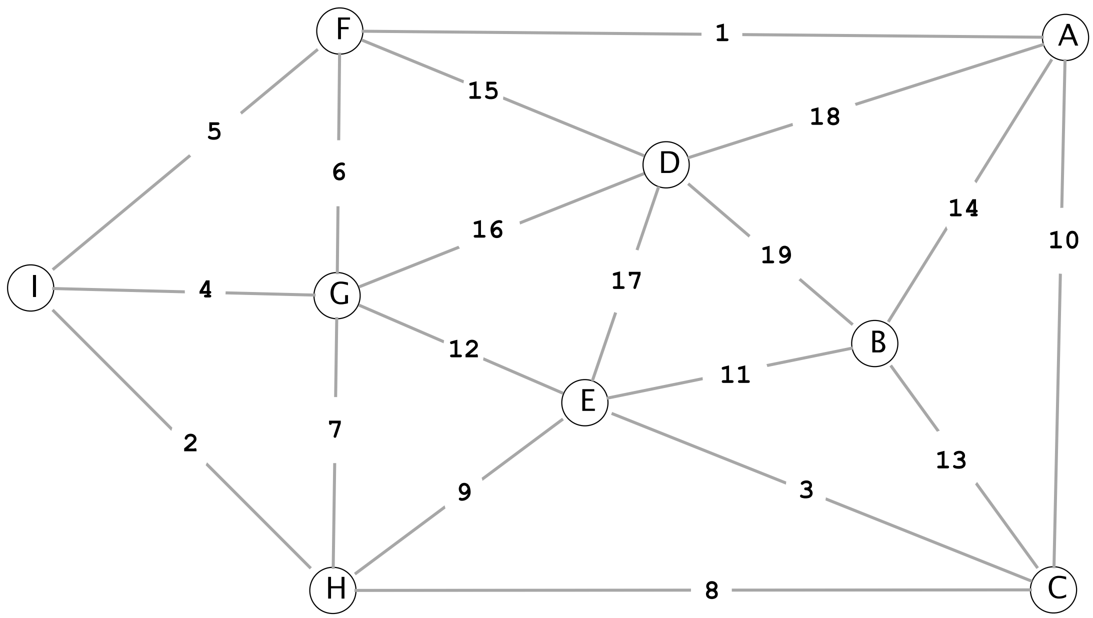
2
--- --- --- --- --- --- --- --- ---

(b) 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
--- --- --- --- --- --- --- --- ---

3. **Minimum spanning trees. (8 points)**

Consider the following edge-weighted graph with 9 vertices and 19 edges. Note that the edge weights are distinct integers between 1 and 19.



(a) Complete the sequence of edges in the MST in the order that *Kruskal's algorithm* includes them (by specifying their edge weights).

```
 1
----   ----   ----   ----   ----   ----   ----   ----
```

(b) Complete the sequence of edges in the MST in the order that *Prim's algorithm* includes them (by specifying their edge weights).

```
 1
----   ----   ----   ----   ----   ----   ----   ----
```

4. **Shortest paths. (8 points)**

Suppose that you are running Dijkstra's algorithm on the edge-weighted digraph (below left), starting from a source vertex s. The table (below right) gives the `edgeTo[]` and `distTo[]` values immediately after vertex 2 has been deleted from the priority queue and relaxed.

edge	*weight*	*edge*	*weight*
$0 \to 2$	6.0	$5 \to 1$	12.0
$0 \to 4$	6.0	$5 \to 2$	1.0
$0 \to 5$	17.0	$5 \to 4$	3.0
$1 \to 3$	17.0	$5 \to 7$	10.0
$2 \to 5$	11.0	$5 \to 8$	4.0
$2 \to 7$	6.0	$6 \to 0$	12.0
$3 \to 0$	1.0	$6 \to 1$	5.0
$3 \to 10$	3.0	$6 \to 2$	1.0
$3 \to 1$	25.0	$6 \to 4$	9.0
$3 \to 6$	13.0	$6 \to 9$	4.0
$3 \to 8$	9.0	$7 \to 1$	7.0
$4 \to 5$	3.0	$7 \to 5$	11.0
$4 \to 6$	4.0	$7 \to 9$	6.0
$4 \to 7$	3.0	$10 \to 1$	15.0
$4 \to 8$	1.0	$10 \to 5$	2.0
$4 \to 9$	15.0	$10 \to 8$	7.0

v	distTo[]	edgeTo[]
0	1.0	$3 \to 0$
1	17.0	$5 \to 1$
2	6.0	$5 \to 2$
3	0.0	*null*
4	7.0	$0 \to 4$
5	5.0	$10 \to 5$
6	13.0	$3 \to 6$
7	12.0	$2 \to 7$
8	9.0	$3 \to 8$
9	∞	*null*
10	3.0	$3 \to 10$

(a) Give the order in which the first 5 vertices were deleted from the priority queue and relaxed.

				2

(b) Modify the table (above right) to show the values of the `edgeTo[]` and `distTo[]` arrays immediately after the next vertex has been deleted from the priority queue and relaxed. Circle those values that changed.

5. **String sorting. (6 points)**

Consider the *first* call to key-indexed counting when running LSD string sort on the input array `a[]` of 20 strings. Recall that key-indexed counting is comprised of four loops. Give the contents of the integer array `count[]` after each of the first three loops (for indices between `'a'` and `'g'`); then, give the contents of the string array (for the indices 0–5 and 18–19) after the fourth loop.

i	a[i]
0	badge
1	freed
2	blurb
3	embed
4	basic
5	field
6	bluff
7	dwarf
8	fudge
9	climb
10	cycle
11	bleed
12	budge
13	crumb
14	cubic
15	cable
16	blend
17	cliff
18	bread
19	cache

c	count[] (*first*)	count[] (*second*)	count[] (*third*)
⋮	⋮	⋮	⋮
'a'			
'b'			
'c'			
'd'			
'e'			
'f'			
'g'			
⋮	⋮	⋮	⋮

i	a[i] (*fourth*)
0	
1	
2	
3	
4	
5	
6	*not required*
7	*not required*
8	*not required*
9	*not required*
10	*not required*
11	*not required*
12	*not required*
13	*not required*
14	*not required*
15	*not required*
16	*not required*
17	*not required*
18	
19	

6. **Substring search. (6 points)**

Below is a partially-completed Knuth-Morris-Pratt DFA for a string s of length 11 over the alphabet { A , B }. Reconstruct the DFA and s in the space below.

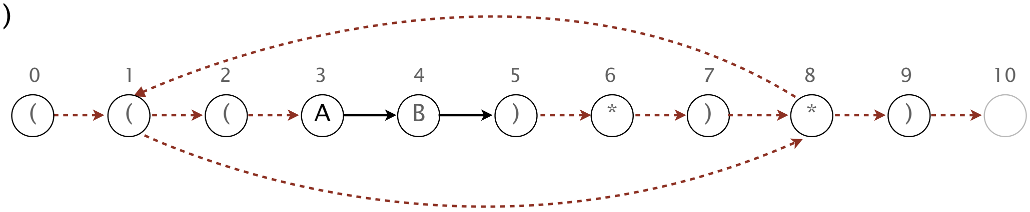
	0	1	2	3	4	5	6	7	8	9	10
A	0	0								10	11
B					5		2				4
s										A	

7. **Regular expressions. (5 points)**

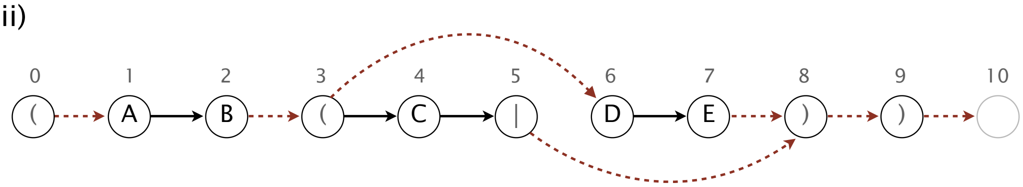
   You have been promoted to COS 226 grader. Circle each NFA below that could have been constructed by the RE-to-NFA algorithm from the textbook. Otherwise, explain one mistake in each invalid NFA.

   The match transitions are drawn with solid lines; the ϵ-transitions are drawn with dotted lines.

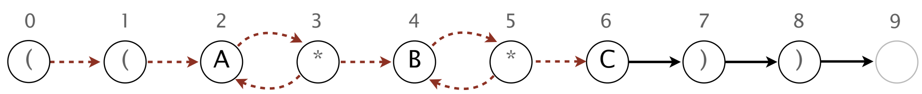
(i)



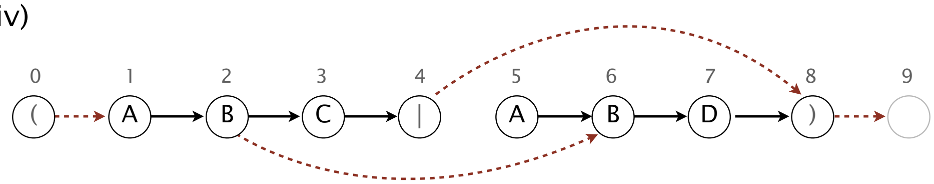
(ii)



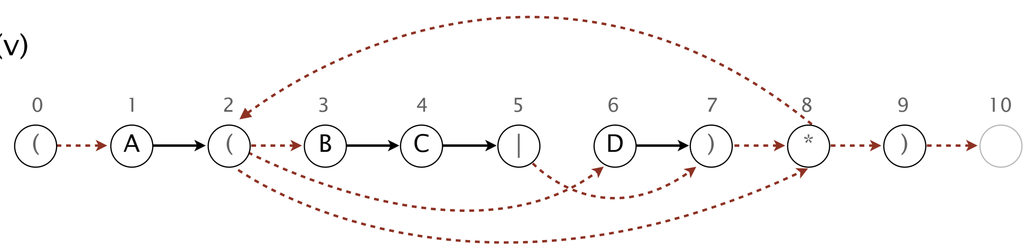
(iii)



(iv)

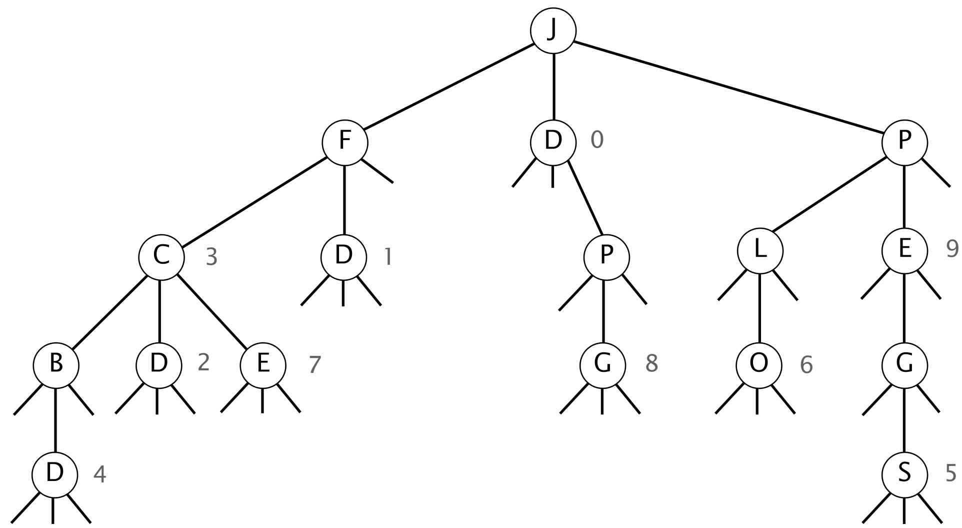


(v)



8. **Ternary search tries. (7 points)**

Consider the following ternary search trie, with string keys and integer values.



Circle which one or more of the following strings are keys in the TST.

`B    BD    C    CD    D    E    FD    JLO    JP    JPEG    JPEGS    JPG    PEG    PEGS`

9. **String symbol table implementation. (7 points)**

For each of the operations on the left, list which one or more of the symbol table implementations on the right can be used to *efficiently* implement it. By efficient, we mean $L \log N$ or better on typical ASCII strings (in random order) of average length L, where N is the number of keys in the data structure.

_____ Find the value associated with a given string key in the data structure.

_____ Associate a value with a string key.

_____ Delete a string key (and its associated value) from the data structure.

_____ Find the smallest string key in the data structure.

_____ Find the smallest string key in the data structure that is greater than or equal to a given string.

_____ Find the string key in the data structure that is the longest prefix of a given string.

_____ How many string keys in the data structure starts with a given prefix?

A. Unordered array.

B. Ordered array.

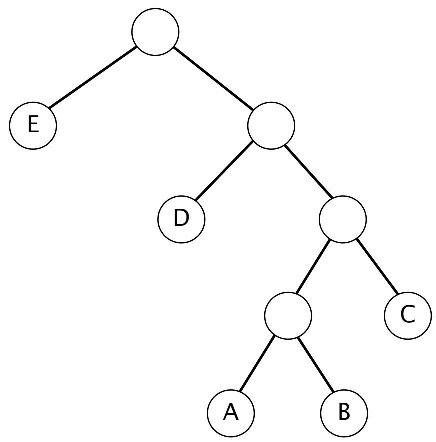
C. Red-black BST.

D. Separate-chaining hash table.

E. Ternary search trie.

10. **Data compression. (10 points)**

(a) Consider the following Huffman trie of a message over the 5-character alphabet $\{A, B, C, D, E\}$:



Identify each statement with the best matching description on the right.

____ The frequency of A is *strictly less than* the frequency of B.

____ The frequency of C is *greater than or equal to* the frequency of A.

____ The frequency of D is *strictly greater than* the frequency of A.

____ The frequency of D is *greater than or equal to* that of A, B, and C combined.

____ The frequency of E is *strictly less than* that of A, B, and C combined.

A. True for all messages.

B. False for all messages.

C. Depends on the message.

(b) Decode each of the following LZW-encoded messages or explain briefly why it is not a valid LZW-encoded message. (Recall that codeword 80 is reserved to signify end of file.)

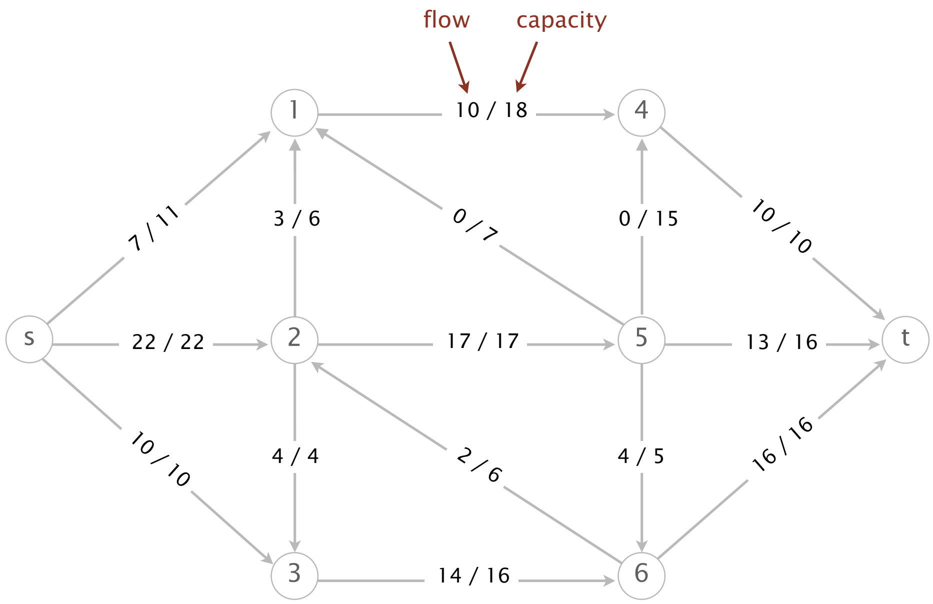
encoded message	*decoded message or brief explanation of why invalid*
41 42 43 44 80	A B C D
42 41 4E 82 41 80	
42 41 83 80	
41 42 81 82 80	
41 42 81 83 80	
42 41 4E 44 41 4E 41 80	

For reference, below is the hexademical-to-ASCII conversion table from the textbook:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	"	#	$	%	&	‘	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

11. **Maximum flow. (8 points)**

Consider the following st-flow network and feasible flow f.



(a) What is the value of the flow f?

(b) Perform one iteration of the Ford-Fulkerson algorithm, starting from the flow f. Give the sequence of vertices on the augmenting path.

(c) What is the value of the maximum flow?

(d) List the vertices on the s side of the minimum cut.

(e) What is the capacity of the minimum cut?

12. **Algorithm design. (8 points)**

Given an edge-weighted graph G and an edge e, design a linear-time algorithm to determine whether e appears in an MST of G. For simplicity, assume that G is connected and that all edge weights are distinct.

Note: Since your algorithm must take linear time in the worst case, you cannot afford to compute the MST itself.

(a) Describe your algorithm in the space below.

(b) What is the order of growth of the running time of your algorithm in the worst case as a function of the number of vertices V and the number of edges E? Circle the best answer.

1 $\quad V \quad$ $E + V$ $\quad E \log^* V \quad$ $E \log E$ $\quad EV \quad$ 2^V

13. **Reductions. (8 points)**

Consider the following two related problems:

- 3Sum. Given an integer array `a[]`, are there three indices `i`, `j`, and `k` (not necessarily distinct) such that `a[i] + a[j] + a[k] == 0` ?

- 3SumVariant. Given two integer arrays `b[]` and `c[]`, are there three indices `i`, `j`, and `k` (not necessarily distinct) such that `b[i] + b[j] == c[k]` ?

(a) Show that 3Sum linear-time reduces to 3SumVariant. To demonstrate your reduction, give the 3SumVariant instance that would be constructed to solve the following 3Sum instance:

`a[]`

-66	-30	70	99	-33	66	20	50

`b[]`

`c[]`

(b) Show that 3SumVariant linear-time reduces to 3Sum. To demonstrate your reduction, give the 3Sum instance that would be constructed to solve the following 3SumVariant instance:

`b[]`

299	700	10	14	-3	-1	20

`c[]`

999	19	-4	600	30	20

Hint: define M equal to 1 + maximum absolute value of any integer in `b[]` *or* `c[]`.

`a[]`