COS 126	General Computer Science	Fall 2008

Exam 2

This test has 9 questions worth a total of 50 points. You have 120 minutes. The exam is closed book, except that you are allowed to use a one-page cheatsheet, 8.5 by 11 inches, handwritten by you, on both sides. No calculators or other electronic devices are permitted. Partial credit will be given for partially correct answers. **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."

Signature

Problem	Score
0	
1	
2	
3	
4	
Sub 1	

Problem	Score
5	
6	
7	
8	
Sub 2	

Total	

Name:

NetID:

Preceptor:	David	Donna
	Jeff	Maia
	Mike	Sid
	Tom	Will
	Wyatt	

0. Miscellaneous. (2 points) (really)

(a) Write your name and Princeton NetID in the space provided on the front of the exam, and circle the name of your preceptor.

(b) *Write* and sign the honor code on the front of the exam.

1. Creating Data Types (6 points)

A very eager student decides to extend the Guitar Hero assignment and actually interface with a real Guitar Hero guitar. During the student's efforts, the need arises for a Button data type. In the space below, please implement the Button class with the following API:

```
public class Button
-------------------------------------------------------------------------
          Button()           constructor; initializes to released state
     void push()             sets the button to the pushed state
     void release()          sets the button to the released state
     void toggle()           toggles (reverses) the button's state
  boolean isPushed()         true if button is in pushed state, else false
```

2. Queue and Stack (5 points)

Here are API's for a queue and a stack of integers:

```
public class QueueOfInts
------------------------------------------------------------
            QueueOfInts()           creates empty queue
    boolean isEmpty()               true if queue is empty
       void enqueue(int item)       enqueues one int
        int dequeue()               dequeues one int

public class StackOfInts
------------------------------------------------------------
            StackOfInts()           creates empty stack
    boolean isEmpty()               true if stack is empty
       void push(int item)          push one int onto stack
        int pop()                   pop one int off stack
```

Fill in the blanks in the following block of client code that reverses the items in a `QueueOfInts` `q` by using an intermediate `StackOfInts s`.

```
    StackOfInts s = new StackOfInts();
    while(!q.isEmpty()) {

        int x = q.________________();

        __________.push(x);
    }

    while(________________________________) {

        int x = _________.pop();

        q._______________(x);
    }
```

3. ADT and Linked Structures (8 points)

As the holiday season approaches, we must remind ourselves of—or perhaps seek distraction from—our respective family trees. Below is a start on building a family tree in Java, using the partially-completed `Person` datatype. In the simplified world of Question 3, by the way, there is no death, divorce, and neither step-children nor half-siblings. Siblings and first cousins never marry each other.

```
public class Person {
    private String name;
    private Person mom;
    private Person dad;
    private Person spouse;
    private Children kids;

    public Person(String name, Person mom, Person dad) {
        this.name = name;
        this.mom = mom;
        this.dad = dad;
        this.spouse = null;
        this.kids = null;
    }

    private class Children {
        Person child;
        Children siblings;
    }

    public boolean isSibling(Person p) {
        if (this == p) return false;            //(you can't be your own sibling)
        return (this.mom == p.mom);
    }

    public void marry(Person fiance) {
        if (this.spouse != null || fiance.spouse != null)
           throw new RuntimeException("Bigamy attempted!");
        this.spouse = fiance;
        fiance.spouse = this;
    }


    ADDITIONAL METHODS GO HERE


}
```

3. (continued)

a. As you probably know, your *first cousins* are the children of your mom's or dad's siblings. Please complete the instance method `isFirstCousin` below. You will want to use the `isSibling` method on the opposite page.

```
public boolean isFirstCousin(Person p) {









}
```

b. Now complete the instance method `birth`, below, which adds a new `Person` to the family tree, and returns a pointer to him or her. Assume that `mom` and `dad` have the same `Children` list, and that this shared list is kept in no particular order.

```
public Person birth(String babyname, Person dad) {
    Person mom = this;
    if (mom.spouse != dad) throw new RuntimeException("Adultery detected!");

    Person baby =

















}
```

4. Regular Expressions and DFA's (8 points)

For each of the following sets of strings over the alphabet {**a, b, c**}, write a regular expression that matches all strings in the set, and choose from among the alternatives below the DFA that accepts that same set (just circle the correct letter):

a. strings that start and end with **a** (including just **a**)

- Regular expression:
- DFA: A B C D E

b. strings containing at most one **a**

- Regular expression:
- DFA: A B C D E

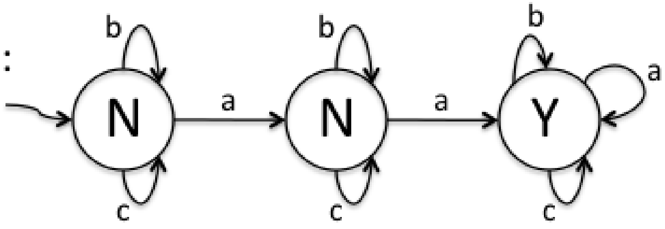
c. strings containing at least two **a**'s

- Regular expression:
- DFA: A B C D E

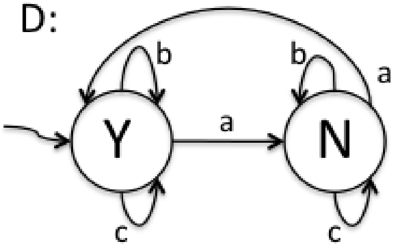
d. strings containing an even number of **a**'s

- Regular expression:
- DFA: A B C D E

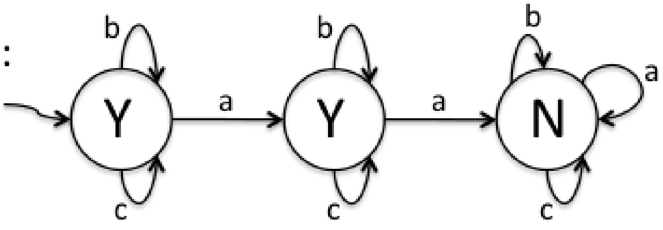
A:



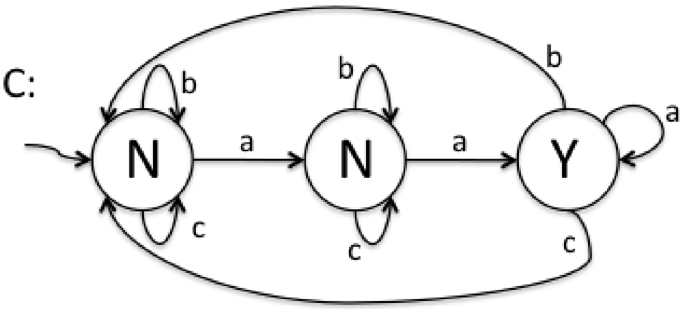
D:



B:



C:



E: none of these

5. Universality and Computability (5 points)

In this question, computing "power" means the ability to perform a computation at all, and not how fast.

a. Turing machines with multiple tapes are more powerful than Turing Machines with just one tape.

Circle one: TRUE FALSE

b. A Turing machine with a magic oracle that solves the Halting Problem is more powerful than the same Turing machine without the oracle.

Circle one: TRUE FALSE

c. Because of the Halting Problem, it is not possible to determine if a program you write actually terminates.

Circle one: TRUE FALSE

d. (Fill in the blank) The TOY machine you studied in class can solve any problem solvable by a Turing machine, provided you allow TOY to have ______________________________________.

e. The Church-Turing thesis says (circle the best answer):

i. Any computational problem can be solved by a Turing Machine.

ii. Any computational problem that can be solved by a physically harnessable process of this universe can be solved by a Turing Machine.

iii. A Universal Turing Machine can simulate any other Turing machine.

6. Intractability (5 points)

Suppose you are a boss of FedEx years from now, and one of your employees (a Harvard grad) comes to you and says that she has invented a new algorithm that solves the traveling salesperson problem (TSP) optimally, and whose running time has order of growth N^3. Quite pleased with herself, she plans to use it to find the absolute shortest possible path that drivers should take when delivering packages to a set of N homes. What should you say? For each possible response below, circle it if you agree, and provide a sentence or phrase in the space below it explaining your thinking, even if you disagree.

a. Big deal, I wrote a program that finds an optimal solution for TSP in COS 126, and its running time had order of growth only N^2.

b. If you indeed have such an algorithm, then P = NP. (Can we split the million dollar prize?)

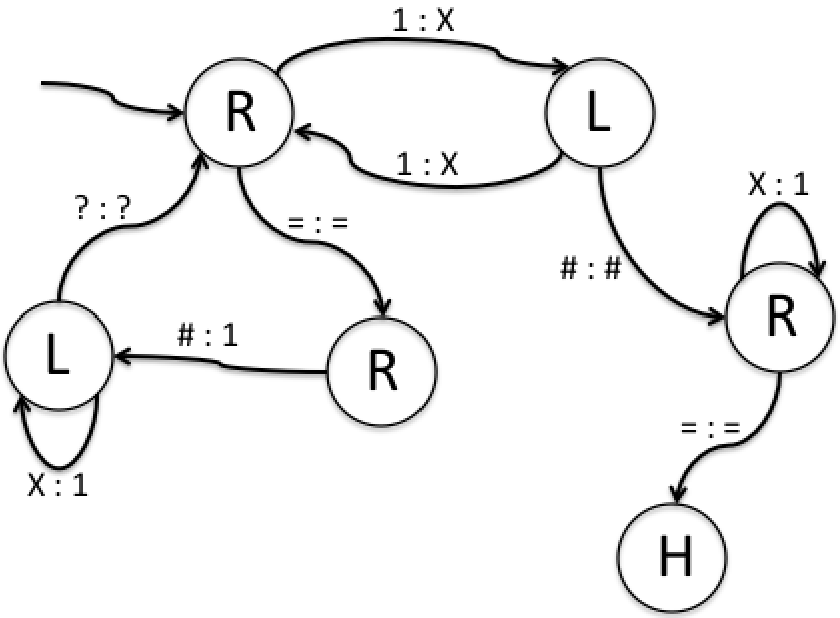
c. Wow, that algorithm could be used to solve the Halting Problem.

d. Cool, that algorithm could be used to break the RSA cryptosystem that encrypts credit card numbers transmitted on the Internet, since RSA depends on factoring and factoring is in NP.

e. That is impossible. Your algorithm runs in Polynomial time, TSP is in NP, and we know that P != NP. Sounds like a contradiction to me.

7. Turing Machines (5 points)

Here's a Turing machine, drawn in the style we've used in lecture and in the theory assignment. It performs some operation on two numbers `m` and `n` expressed in *unary* notation: for example, the number 3 is represented by 3 consecutive 1's.



a. If this Turing machine is given the tape below, what will be on the tape when it halts?

									m						n								
initial:	. . .	#	#	#	#	1	1	1	1	1	1	1	?	1	1	1	=	#	#	#	#	#	. . .
													↑										

final:	. . .	#	#	#	#								?				=						. . .

b. What function of `m` and `n` does this Turing Machine compute?

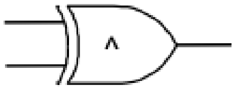
8. Circuits (6 points)

You enjoyed TOY so much, you volunteered for the Electrical Engineering Department student project to build an actual, hardware TOY. You are put to work on the Arithmetic Logic Unit, and the tyrannical grad student in charge decides that you are just the person to design the 16-bit adder. You vividly recall from COS 126 that an "Odd" cicuit will be required. Each Odd circuit (one per bit, you recall) takes three inputs: data input bits A and B, and a Carry bit. (Still fresh in your mind is the knowledge that an Odd circuit outputs 1 only when an odd number of its inputs are 1.)

a. Make a Truth Table for Odd.

b. Write a Sum-of-Products Boolean expression for Odd.

c. The tyrannical grad student forgot to tell you that all you have are 2-input XOR gates. **Rewrite** your Boolean expression using only XOR operations. Then **draw** an Odd circuit using only 2-input XOR gates. (Remember that XOR outputs 1 if exactly one of the two inputs is 1.) Here's what one looks like:



d. Although you were able to make an Odd circuit using only XOR's, it turns out that XOR by itself is not *universal*. What does this mean?