Print your name and netid neatly in the space provided below; print your netid in the upper right corner of every page.

Name: ________________________________
Netid: ________________________________

This is a closed book, closed notes examination. You may not use calculators or any other electronic devices. Any sort of cheating on the examination will result in a zero grade.

We cannot give any clarifications about the exam questions during the test. If you are unsure of the meaning of a specific question, write down your assumptions and proceed to answer the question on that basis.

Do all the problems in this booklet. Do your work inside this booklet, using the backs of pages if needed. The problems are of varying degrees of difficulty so please pace yourself carefully, and answer the questions in the order which best suits you. Answers to essay-type questions should be as brief as possible. If the grader cannot understand your handwriting you will get 0 points.

There are 10 questions on this exam and the maximum grade on this exam is 80 points.
1. eXtreme Programming (XP)

(a) One of the 12 pillars of XP is *Simple Design*. Practitioners of XP prefer simple design over *big design up front* (BDUF). One challenge, though, is how to maintain a simple design as the project evolves. Pick **two** other pillars of XP that you are familiar with and describe how those two pillars help drive a simple design.

(b) In the context of XP, how are acceptance tests similar to unit tests? How are they different?
2. Reverse Engineering

(a) Describe what reverse engineering is in one sentence.

(b) In lecture, Prof. Johnson mentioned several tools that developers could use to help them reverse engineer a system. Give two examples of such tools. For each tool, describe (i) what information they can provide and (ii) how you would use them to reverse engineer a system.

(c) Sometimes using software metrics can be useful in guiding the reverse engineering process. Pick one software metric that you are familiar with and describe what it measures. Then, describe how you would use that metric to help guide you through the reverse engineering process.
3. Software Architecture

3. (a) Eclipse is an example of a “plug-in” architecture. What is a plug-in, and what does it mean to have a plug-in architecture?

3. (b) Why is a plug-in architecture a good choice for building an integrated development environment (IDE)?

4. Software Quality Attributes

2. (a) What does the acronym FURPS stand for?

4. (b) Describe a situation in which a design decision that improves one quality attribute will make another one worse.
3. (c) The article *Capturing Architectural Requirements* from IBM says that providing a printing capability (in an application) is a functional requirement of particular significance to architecture. Explain.

5. Software Configuration Management

3. (a) Name three components/documents that are part of a software product that you might want to keep under Software Configuration Management.

2. (b) Name two goals of proper Software Configuration Management?
6. Software Metrics

(a) Bill Gates has been quoted as saying “Measuring software productivity by lines of code is like measuring progress on an airplane by how much it weighs”. And, yet, lines of code is the most popular software metric that we have today. In fact, look up any published paper or website and lines of code is the dominant (if not the only) metric reported. Besides being simple to collect, what are two other advantages of measuring lines of code? What is one problem of measuring lines of code?

(b) The object-oriented metrics we discussed in class have strong correlations with code smells. For instance, when a particular component is not within the acceptable range of a metric, it is usually a good indication that it is suffering from some code smell as well. For each metric below, list and describe one code smell that could contribute to it being out of the acceptable range. To prevent guessing, list only one code smell for each metric. If you list more than one, you get zero for that subpart.

i. Coupling Between Objects

ii. Number of Children

iii. Response for a Class
7. Code smells and refactorings

Name three code smells that are present in the code below. It is OK if you circle or underline the offending code. For each code smell that you identify, suggest how you would refactor it to remove the offensive code smell. You do not need to implement the actual refactoring. Use the space on the next page for your suggestions.

```java
public class Account {
    private int accNo;
    private double amount;
    private double interestRate;

    // Constructor omitted

    public int getAccountNumber() { return accNo; }
    public double getAmount() { return amount; }
    public double getInterestRate() { return interestRate; }
}

public class Customer {
    private String name;
    private List<Account> accounts;

    public String getStatement() {
        // Print statement header
        String statement = "Statement for " + name + "\n";

        // Print info for each account
        for (int m=0; m < accounts.size(); m++) {
            Account account = accounts.get(m);

            // Get info about the account
            int accNo = account.getAccountNumber();
            double amount = account.getAmount();
            double interestRate = account.getInterestRate();

            statement += "For account " + accNo + "\n";

            // Print account summary
            statement += "Amount available: " + amount;
            statement += "interest: " + amount * interestRate;
            statement += "\nTotal: " + amount + (amount * interestRate);
            statement += "\n\n";
        }

        return statement;
    }
}
```

Listing 1: First Iteration of a Banking System
8. Testing Techniques

(a) Describe two reasons why manual testing is better than automated testing. Similarly, describe two other reasons why automated testing is better than manual testing.

(b) For each of the following cases, justify why automated testing or manual testing would work better.

   i. Exploratory test cases

   ii. GUI tests

   iii. Regression testing
9. In the following Java code fragment, `getPeopleAsString` is a method that accepts a set of person objects and returns a string containing the first name and last name of each person. In the returned string, the information of each person is separated from the next person by ",".

Having this code snippet, answer the following questions as described in Code Complete 2:

```java
public String getPeopleAsString(Set<Person> people) {
    String firstName;
    String lastName;
    String peopleString = "";

    for (Person person : people) {
        if (person.firstName != null) {
            firstName = person.firstName;
        } else {
            firstName = "";
        }
        if (person.lastName != null) {
            lastName = person.lastName;
        } else {
            lastName = "";
        }
        peopleString += firstName + "," + lastName + ";";
    }
    return peopleString;
}
```

(a) What is the minimum number of test cases needed for structured basis testing? Explain your reasoning.

(b) What is the minimum number of test cases needed for data flow testing? Explain your reasoning.
10. White-Box and Black-box Testing

(a) What is the main difference between white-box and black-box testing?

(b) Is equivalence partitioning considered as a white-box or back-box testing techniques?

(c) Explain how you can use equivalence partitioning for testing the code example in Question 9.