IC470 Software Engineering
12 Week Problem Set

Due: As per the course webpage.

Collaboration: You may only collaborate on this problem set as a homework with other midshipmen currently enrolled in this course, and you must collaborate with at least one other person. You must turn in only a SINGLE solution with the names of all the midshipmen that collaborated on the solution in the top right corner. The graded work will be handed back in the same section that it was submitted. If your name appears on more than one solution, you will receive the lowest score of the work on which your name appears.

Project Planning. Consider a project that has three major components (WebUI, DB, and AI) and the following constraints:

i. Assuming that you could put all available resources toward one component, you estimate that it would take one week to gather requirements, one week to design, one week to implement, and one week to test. (So, four weeks total per component.)
ii. The WebUI and AI components can be implemented concurrently.
iii. The AI portion has a lot of customer uncertainty regarding what is needed.
iv. DB portion must be developed after everything else has finished and been tested.

1. (10 pts) Give a Gantt Chart for the above project using the *Agile* Model. Be sure to indicate when you would meet with the customer (put a symbol of some kind on the chart). State any assumptions that you make when making your Gantt Chart.

Chapter 7: (Coupling & Cohesion)

2. (5 pts) How should the classes of a software system be designed so as to be as reusable as possible?

3. (5 pts) What is the influence of cohesion on maintenance?

4. (5 pts) What is the influence of coupling on maintenance?

5. It has been suggested that C++ and Java support the implementation of abstract data types, but only at the cost of giving up information hiding.

   a. (5 pts) Define information hiding, and

   b. (5 pts) Clearly attack or defend this suggestion, and support your answer.

6. Consider the below (syntactically correct) source code:

```
public class main{
  public static void main(String[] args){
    Type1 t1 = new Type1(1);
    Type2 t2 = new Type2(t1);
    Type3 t3 = new Type3(t1);
    Type1[] all = {t1, t2, t3};
    System.out.println(t1.getContents(t1));
    System.out.println(t2.getContents(t1));
    System.out.println(t3.getContents(t1));
    System.out.println(t3.getContents(t3));
    System.out.println(getTotal(all));
  }
  private static int getTotal(Type1[] all){
    Type1 temp = all[0];
    int total=0;
    all[0]=all[1];
    all[1]=temp;
    for(int i=0; i<3; i++)
      total+=all[i].contents;
    return total;
 }
}
```

```
public class Type2 extends Type1{
  public Type2(Type1 in){
    value=in;
    contents=in.contents++;
  }
  public String getContents(Type2 t){
    return t.getContents()+"202";
  }
  public String getContents(){
    return Integer.toString(contents);
  }
  private Type1 value;
}
```

```
public class Type1{
  public Type1(){}
  public Type1(int in){
    contents=in;
  }
  public String getContents(Type1 t){
    return t.getContents();
  }
  public String getContents(){
    return Integer.toString(contents+1);
  }
  public int contents;
}
```

```
public class Type3 extends Type1{
  public Type3(Type1 in){
    value=in;
    contents=in.contents++;
  }
  public String getContents(Type1 t){
    return t.getContents()+"303";
  }
  public String getContents(){
    return Integer.toString(contents);
  }
  private Type1 value;
}
```

a. (5 pts) **Coupling:** What is the WORST level of coupling between the modules? Identify the level of coupling, which specific classes, methods, and/or variables are involved and justify your answer.

b. (5 pts) **Cohesion**: Identify the level of cohesion in the main class's getTotal function and justify your answer.

Chapter 14: (Object-Oriented Design)

Consider the below ATM system description (from howstuffworks.com, *note that it is the same* as the description in the 6 week homework set), and answer the following questions. Make sure that *all methods and classes* identified in your sequence diagram are made an appropriate part of your UML detailed class diagram.

An ATM is simply a data terminal with two input and three output devices. A normal data terminal has some sort of keyboard for input, some sort of screen for output, and a network connection that lets it talk to a server somewhere on the network. An ATM adds a card reader as an input device, along with a printer and an amazing money dispenser as output devices, to create a complete package.

Settlement Funds

Let's say you want to get some money from an ATM at a convenience store. Chances are that the merchant who owns the store either owns or rents the ATM. So the merchant fills the ATM with cash each day, and it is the merchant's cash that you receive when you get money from the ATM.

You walk up to the ATM, insert your card, and type your password. The card tells the machine your bank and account information. The ATM forwards this information to the host processor, which routes the transaction request to your bank.

If you're requesting cash, the host processor causes an electronic funds transfer to take place from your checking account to the host processor's account. Once the funds are transferred to the host processor's bank account, the processor sends an approval code to the ATM authorizing the machine to dispense the cash. The host processor then sends your funds into the merchant's bank account by automated clearing house, usually the next bank business day. So when you request cash, the money moves electronically from your account to the host's account to the merchant's account, and you get the merchant's cash.

Now you know what the virtual process is, but what's actually going on inside the machine?

Parts of the Machine

An ATM has two input devices, the card reader and the keypad. The card reader captures the account information stored on the magnetic stripe on the back of an ATM card. The keypad lets the cardholder tell the bank what kind of transaction is required (cash withdrawal, balance inquiry, or whatever) and for what amount. Also, the bank requires the cardholder's personal identification number (PIN) for verification.

The most important output device is the heart of an ATM—the safe and cash-dispensing mechanism. The entire bottom portion of most small ATMs is a safe that contains the cash. The cash is stored in a series of cassettes—a big ATM in a high-traffic area can hold up to $100,000. The bill count and all of the information pertaining to a particular transaction is recorded in a journal. The journal information is printed out periodically and the machine owner maintains a hard copy for two years.

Besides the electric eye that counts each bill, the cash-dispensing mechanism also has a sensor that evaluates the thickness of each bill. If two bills are stuck together, then instead of being dispensed to the cardholder they are diverted to a reject bin. The same thing happens with a bill that is excessively worn or torn, or is folded. So, while it's not likely you're going to get an extra 20-dollar bill with your next withdrawal, you'll be happy to know you won't get half of a bill either.

7. (5 pts) Give a UML Sequence Diagram for the ATM scenario of depositing $100, and then successfully withdrawing $20 from your account in the abnormal case in which the top two $20 bills in the dispenser are stuck together, but the third $20 bill is able to be successfully dispensed.
   - You may only use classes in your sequence diagram if they appear in your UML class diagram (revisit your UML Class Diagram from homework 1 and add classes to your UML class diagram as needed).
   - For any classes that you add, provide a 1 paragraph discussion of why you needed to add the class now, and why you were not able to detect the need for the class as you prepared your answer for homework 1 previously.

8. (5 pts) Give, as a separate diagram, your UML Class Diagram from homework 1 (modified as needed to correct any deficiencies) expanded into a UML *Detailed* Class Diagram for the entire ATM system. All method invocations shown in your UML Sequence Diagram must be present as member functions defined on the *appropriate* classes in your UML *Detailed* Class Diagram.

Chapter 15: (Testing)

9. (5 pts) Give the McCabe graph for the code below.

10. (5 pts) Compute McCabe's M value for the code below.

11. (5 pts) Determine how many paths there are through the code below. Hint: find how many paths w/o loop and then consider the loop.

12. (5 pts) Determine how long it would take to test the code for complete path coverage assuming it takes 30 seconds to run and evaluate the results of each test case. Give your answer in terms of years.

```
int i=0;
do {
   a = result(); // result() returns int between 1..5
   switch a {    // result() has McCabe value of 6
      case 1: x=3;
              break;
      case 2: if (b == 0)
                  x=2;
              else
                  x=4;
              break;
      case 3: process(c);
              break;
  } // end switch
 i++;
} while (i < 10)
```

13. (5 pts) Give one black box test case for the above code. If one cannot be determined, explain why.

14. (5 pts) Give one glass box test case for the above code that tests "case 1:" If a test case cannot be determined, explain why.

15. (5 pts) What is regression testing?

ACM/IEEE Software Engineering Code of Ethics

16. Which principle of the Software Engineering Code of Ethics is most applicable to your capstone project?
    a. (4 pts) Give the specific principle by name (not number)
    b. (4 pts) Explain why it is the most applicable to your capstone project
    c. (2 pts) Explain what your team can do to ensure that you adhere to the principle.