

CMPS115 Winter 2004 HW 2 (ilities) (80 points)

Problem 1:

(30 points) Here you are asked to determine some simple A_0 numbers. Show your work, else there is no possibility of partial credit!

There are two computers, each runs an identical software stack (operating system, Virtual Machine, application). Some quality facts:

- The computers are not subject to any hardware failures, as long as they have power.
- The OS has an MTTF of 168 hrs, with an MTTR (reboot) of 1 minute.
- The VM has MTTF of 1000 hours, and an MTTR of 1 minute.
- The application has an MTTF of 24 hrs and a MTTR of 1 minute.

(a) (6 pts) What is the A_0 of each of the three components of the stack, considered independently?

(b) (4 pts) What is the A_0 of each computer software stack?

(c) (5 pts) On average, how long does the software stack stay up between failures?

(d) (8 pts) Using a work-duplicator and response-combiner function of some kind, the two computers are placed in parallel to attempt to improve system availability. What is the resulting availability?

(e) (8 pts) Both computers are plugged into the same electrical socket. The building power has an MTBF of 2000 hours and an MTTR of 3 hrs. What is the A_0 of the parallel system configuration?

CMPS115 Winter 2004 HW 2 (ilities) (80 points)

Problem 2:

(10 points) You are given the code fragment below; rewrite it so that the function of the code does not change at all, but the Halstead Effort number is reduced. (The resulting code will be slightly shorter.) Don't make any changes that do not reduce the Halstead Effort number.

```
public class halTest{
    public int methodWithLongName (int x,
                                    int longNameVariable){
        System.out.println("x = " + x);
        System.out.println("the other param = " +
                            longNameVariable);
        // this is a long comment to explain
        // what I am thinking about

        int x1 = x + longNameVariable;
        System.out.println("x1 = " + x1);

        int x2 = x - longNameVariable;
        System.out.println("x1 = " + x2);

        return x1 + x2;
    }
}
```

Problem 3:

(15 points) Draw a UML structure diagram for an AbstractFactory design for getting an object which implements a "Connection" interface. The Connection interface has four methods on it:

- connect(String destination) which returns a boolean,
- sendMessage(String s) which transmits a message,
- getMessage() which returns a String, and
- disconnect() which has a void return.

There are two concrete Connections you might serve up: a TcpConnection, and a TelephoneConnection.

To get you started: the client will get the AbstractFactory by calling MyConnectionFactory.getInstance(). This method (which you do **not** have to code!) looks at a property value and returns either a TcpConnectionFactory or a TelephoneConnectionFactory.

CMPS115 Winter 2004 HW 2 (ilities) (80 points)

Problem 4:

(25 points) You are given a design as indicated in the UML structure chart below. Re-design this into a composition/delegation design that avoids inheritance. You will probably need to define a small number of interfaces or abstract classes, and a small number of implementing classes. Keep in mind that relationships can exist between interfaces, abstract classes, or concrete classes.

