Midterm

Note

This is an open-book exam. You may consult any books, papers, or notes, but discussion is strictly forbidden.

Problems

1. (10 %) Prove the following sequents using *Natural Deduction* (in the sequent form).

   (a) $\vdash (\neg B \rightarrow \neg A) \rightarrow (A \rightarrow B)$

   (b) $\vdash (\exists y P(y) \vee \exists z Q(z)) \rightarrow \exists x(P(x) \vee Q(x))$

2. (10 %) The first-order theory for *monoids* contains the following two axioms:

   - $\forall a \forall b \forall c(a \cdot (b \cdot c) = (a \cdot b) \cdot c)$ (Associativity)
   - $\forall a((a \cdot e = a) \wedge (e \cdot a = a))$ (Identity)

   Here e is a constant, called the identity, and $\cdot$ is the binary operation. Let M denote the set of the two axioms. Prove using *Natural Deduction* (with the =-introduction and =-elimination rules) the sequent $M \vdash \forall e'(\forall a((a \cdot e' = a) \wedge (e' \cdot a = a)) \rightarrow e' = e)$, which says that the identity element of a monoid is unique. (Hint: a typical proof in algebra books is the following: assuming e' is an identity, $e' = e' \cdot e = e$.)

3. (20 %) Prove the total correctness of the following program.

```
{$true$}
S1: $m, i := A[0], 1$;
S2: **while** $i < n$ **do**
    S3: **if** $A[i] > m$ **then** $m := A[i]$;
    S4: $i := i + 1$
**od**
{$\forall j(0 \leq j < n \rightarrow A[j] \leq m)$}
```

4. (20 %) You were assigned to design an information system for a university. Among other things, you have gathered the following requirements:

(a) This university has several colleges, each with several departments.

(b) Every college has a name, the year of establishment, a dean, and one to three associate deans.

(c) Every department has a name, the year of establishment, a chair, and less than two vice-chairs.

(d) Every faculty member belongs to (is appointed by) a department, and may jointly be appointed by some other departments.

(e) The university president, college deans and associate deans, and department chairs and vice chairs must also be faculty members.

(f) A staff member belongs to either the university, a college, or a department; no joint appointments are allowed.

(g) The president must have the highest salary among all university employees.

(h) For every department, the number of faculty members must be at least four times of the number of staff members.

Now the next step should be to make all these more precise by drawing a UML class diagram and adding OCL constraints (in the diagram). Please carry out this step for the requirements listed above; make assumptions wherever necessary.

5. Software engineers at SDM.com are implementing a multi-platform application. The following is their code for this application.

```
interface GUIFactory {
    public Button createButton();
    public CheckBox createCheckBox();
}

class WindowsFactory implements GUIFactory {
    private WindowsFactory() {};
    private static WindowsFactory instance;

    public static WindowsFactory getInstance() {
        if (instance == null)
            instance = new WindowsFactory();
        return instance;
    }
    public Button createButton() {
        return new WindowsButton();
    }
    public CheckBox createCheckBox() {
        return new WindowsCheckBox();
    }
}

class LinuxFactory implements GUIFactory {
    private LinuxFactory() {};
    private static LinuxFactory instance;

    public static LinuxFactory getInstance() {
        if (instance == null)
            instance = new LinuxFactory();
        return instance;
    }

    public Button createButton() {
        return new LinuxButton();
    }
    public CheckBox createCheckBox() {
        return new LinuxCheckBox();
    }
}

interface Button {
    public void draw();
}

class WindowsButton implements Button {
    public void draw() {
        System.out.println("I'm a WindowsButton");
    }
}
```

```
class LinuxButton implements Button {
        public void draw() {
                System.out.println("I'm an LinuxButton");
        }
}

interface CheckBox {
        public void draw();
}

class WindowsCheckBox implements CheckBox {
        public void draw() {
                System.out.println("I'm a WindowsCheckBox");
        }
}

class LinuxCheckBox implements CheckBox {
        public void draw() {
                System.out.println("I'm an LinuxCheckBox");
        }
}

class Application {
        public Application(GUIFactory factory) {
                Button button = factory.createButton();
                button.draw();
                CheckBox checkbox = factory.createCheckBox();
                checkbox.draw();
        }
}

public class PerformApplication {
        public static void main(String[] args) {
                Application app = null;

                if (args[0].equalsIgnoreCase("windows")) {
                        app = new Application(WindowsFactory.getInstance());
                }
                else if (args[0].equalsIgnoreCase("linux")) {
                        app = new Application(LinuxFactory.getInstance());
                }
        }
}
```

Please answer the following questions according to the above java code.

(a) (2 %) Which design pattern(s) do they used in their code?

(b) (5 %) Draw the class diagram of this code and specify the roles (participants) the classes play in the design pattern(s).

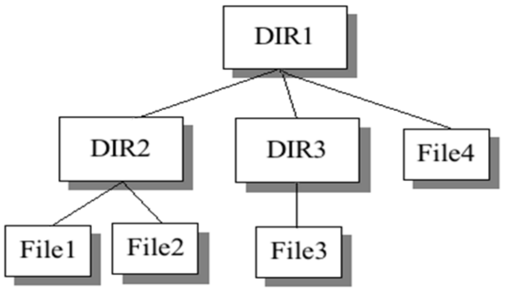
(c) i. (4 %) We have one more kind of OS, Mac, that needs to be added into this application. Please write the necessary code needed for this requirement.

ii. (4 %) We have one more kind of component, RadioBox, that needs to be added into each kind of OS supported in this application. Please describe what should be modified to fulfill this change. (No code needed)

6. Assume that we want to implement a "virtual file system" that supports various general file system operations. Instead of creating physical files/directories on the operating system, a "virtual file system" keeps everything in memory so that it can be manipulated like a set of objects. For example, you can create an object called "Directory" and associate sub-directories/files with it.

You can use the virtual file system to maintain a user's directory structure and serialize the structure to either a file or database.

(a) (5 %) Imagine that we want to model a directory hierarchy shown below. The Composite pattern is definitely a good candidate for implementing the "virtual file system". Please use class diagram to design such virtual file system with the Composite pattern, and provide Java or C++ code to build the object structure shown below.



(b) (5 %) We can foresee that the virtual file system will need to support more operations in the future. For example, we might want to calculate the total size of files under a specific folder. Please provide an extension in your previous design to support this flexibility, and give a concrete scenario to explain how it works. You can extend your previous class diagram or provide pseudo code.

7. Suppose you have the following abstraction: Customer and Store. Some Customers are interested in a particular best-selling product, but the product is not on the market or out of stock. Customers can inquire the Store if the product is available. Here are the classes that represent Customer and Store:

```
class Customer {
    public void waitForProduct() {
        while (true) {
            Store store = getToStore();
            if (store.isProductAvailble())
                break;
            else
                // we will go back to store and ask again tomorrow
                sleep (oneDay);
        }
    }

    private Store getToStore () {
        // returns the Store object
    }
}

class Store {
    public boolean isProductAvailable() {
        // returns true if the product has arrived
    }
}
```

Suppose Store and Customer both feel it troublesome to have periodical inquiries for the availability of the product, and you want to change the design: the Customers can register at the Store so that they can get notified when the product becomes available. A registered Customer may also unregister if he or she doesn't want to wait for the product anymore.

(a) (6 %) What design pattern will you use for this purpose? Please redesign the Store and Customer classes and to address this requirement.

(b) (4 %) Suppose the product is too popular and the Store wants to limit the number of Customers that can register for product arrival notifications. Please address this requirement in your design. You may assume there are collection (or aggregate) classes that are already available and don't have to implement them on your own.

Suppose now a Customer may tell the Store what action to perform when the product has arrived. Now the Customer has the Action object:

```
class Customer {
        // other member fields and methods

        // the action the store to perform when the product has arrived
        private Action action;
}
```

Here is the Action interface:

```
abstract class Action {
        // perform the action
        public abstract void perform();
}
```

(c) (5 %) Please modify the Customer notification mechanism by applying the Command Pattern so that the Store can perform the custom Action (for example, sending short message or email to the customer) the Customer would like the Store to perform. Please also provide at least one concrete Action class (but don't have to provide the details of how the action is performed).