Midterm

Note

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

Problems

1. (10 %) Prove the following sequents using *Natural Deduction* (in the sequent form). You may assume $\Gamma \vdash A \vee \neg A$ to be an axiom (the Law of Excluded Middle) if it makes the proof simpler and shorter.

   (a) $\neg A \wedge \neg B \vdash \neg(A \vee B)$

   (b) $\neg(A \wedge B) \vdash \neg A \vee \neg B$

2. (20 %) A majority of an array of n elements is an element that has more than $\frac{n}{2}$ occurrences in the array. Below is a program that finds the majority of an array X of n elements or determines its non-existence. (Hint: if $A[i] \neq A[j]$, then the majority of A remains a majority in a new array B obtained from A by removing $A[i]$ and $A[j]$.)

```
C,M := X[1],1;
i := 2;
while i<=n do
   if M=0 then C,M := X[i],1
          else if C=X[i] then M := M+1
                         else M := M-1
               fi
   fi;
   i := i+1
od;
if M=0 then Majority := -1
       else Count := 0;
            i := 1;
```

```
            while i<=n do
               if X[i]=C then Count := Count+1 fi;
               i := i+1
            od;
            if Count>n/2 then Majority := C
                         else Majority := -1
            fi
fi
```

(a) Define (mathematically, using recursion and logical formulas) a function *cnt* such that $cnt(a, i, j)$ gives the number of occurrences of a in $X[i..j]$ and a predicate *maj* such that $maj(a, i, j)$ is true if and only if a is a majority in $X[i..j]$.

(b) Devise a suitable loop invariant for the first while loop (that is strong enough for actually proving the partial correctness of the program).

3. (3 %) List three advantages of writing design documents.

4. Software engineers at SDM.com are implementing a restaurant game. In this game, different styles of restaurant will have different looks. The following is a code snippet for this application in C++:

```
class Restaurant {
private:
    FFSeat* ffseat;
    FFTable* fftable;
    LBSeat* lbseat;
    LBTable* lbtable;

public:
    Restaurant (char* inputStyle){
        style = inputStyle;
        if (strcasecmp(style, "FastFood") == 0) {
            ffseat = new FFSeat;
            fftable = new FFTable;
        }
        else if (strcasecmp(style, "LoungeBar") == 0) {
            lbseat = new LBSeat;
```

```
            lbtable = new LBTable;
        }
    }

    void draw() {
        if (strcasecmp(style, "FastFood") == 0) {
            ffseat->draw();
            fftable->draw();
        }
        else if (strcasecmp(style, "LoungeBar") == 0) {
            lbseat->draw();
            lbtable->draw();
        }
    }
}

// Fast Food Style
class FFSeat {
public:
    void draw() { cout << "Plastic Chair\n"; }
};

class FFTable {
public:
    void draw() { cout << "Plastic Table\n"; }
};

// Lounge Bar Style
class LBSeat {
public:
    void draw() { cout << "Sofa\n"; }
};

class LBTable {
public:
    void draw() { cout << "Glass Table\n"; }
};
```

```
int main(char* style) {
    Restaurant* restaurant = new Restaurant(style);
    restaurant->draw();
}
```

(a) (6 %) List creational design pattern(s) that could be applied to improve this code and explain why you choose them?

(b) (6 %) Draw the class diagram after applying the design pattern and specify the roles (participants) the classes play in the design patterns.
(You may add additional classes to apply design pattern(s).)

5. Assume we will develop a logging framework for all product teams within our company. The logging framework needs to be general and flexible enough to meet the requirements from different product teams. Therefore, we come up with an idea that uses the concept of wrapper classes to dynamically attach more functionality around our basic logger class. We should be flexible enough so that one wrapper can wrap another wrapper to achieve the desired effect. Most importantly, the user of logging classes should not be aware of what wrappers are being used underneath. They should just use the general logging interface. Here is the sample client code in Java that uses the logging framework.

```
public void methodA(){
    Logger logger = Logger.getInstance();
    logger = new SystemLoadLogWrapper(logger);

    // the following ling outputs "[CPU 87%][MEM 50%]: Message 1"
    logger.logMessage("Message 1");

    logger = new DateTimeLogWrapper(logger);

    // the following line outputs
    // "Wed Dec 09 20:39:07 EST 2009: [CPU 80%][MEM 52%] : Message 2"
    logger.logMessage("Message 2");
}
```

(a) (4 %) Which pattern might be a good candidate to provide such flexibility? Why would you use this pattern?

(b) (6 %) Please provide a UML class diagram that describes the class relationship (You should include the "Logger", the "SystemLoadLogWrapper", the "DateTimeLOgWrapper" and any classes that might be relevant).

6. Suppose we have a program that synchronizes the contents from and to various data sources. Currently the program supports databases, local files and TCP sockets as data sources. Here is a code snippet of the program in Java (for simplicity, we assume each data source uses the same representation of data as class DataSet):

```
public class DatabaseDataSource {
    // a reference to file data source
    private LocalFileDataSource fileDS;

    // a reference to TCP socket data source
    private TCPSocketDataSource tcpDS;

    // variables indicating whether the other data sources should be updated
    private boolean shouldUpdateLocalFile;
    private boolean shouldUpdateTCPSocket;

    public void dataChanged (DataSet dataSet) {
        if (shouldUpdateLocalFile)
            fileDS.modifyFile (dataSet);
        if (shouldUpdateTCPSocket)
            tcpDS.updateRemoteData (dataSet);
    }

    public void updateData (DataSet dataSet) {
        // logic to update dataSet to database
    }
}

public class LocalFileDataSource {
    // a reference to database data source
    private DatabaseDataSource databaseDS;

    // a reference to TCP socket data source
    private TCPSocketDataSource tcpDS;
```

```
    // variables indicating whether the other data sources should be updated
    private boolean shouldUpdateDatabase;
    private boolean shouldUpdateTCPSocket;

    public void fileModified (DataSet dataSet) {
        if (shouldUpdateDatabase)
            databaseDS.updateData (dataSet);
        if (shouldUpdateTCPSocket)
            tcpDS.updateRemoteData (dataSet);
    }

    public void modifyFile (DataSet dataSet) {
        // logic to update dataSet to local file
    }
}

public class TCPSocketDataSource {
    // a reference to database data source
    private DatabaseDataSource databaseDS;

    // a reference to file data source
    private LocalFileDataSource fileDS;

    // variables indicating whether the other data sources should be updated
    private boolean shouldUpdateDatabase;
    private boolean shouldUpdateLocalFile;

    public void remoteDataUpdated (DataSet dataSet) {
        if (shouldUpdateDatabase)
            databaseDS.updateData (dataSet);
        if (shouldUpdateLocalFile)
            fileDS.modifyFile (dataSet);
    }

    public void updateRemoteData (DataSet dataSet) {
        // logic to update dataSet to via TCP socket
```

```
    }
}
```

It can be seen that the program is not well designed. Each data source has to know the presence of other data sources to do updates. The data sources are highly coupled, which makes it difficult and error-prone to add new data sources or to change the interaction between existing data sources. It's time to refactor the design.

(a) (8 %) What design pattern will you use to decouple the 3 data sources? Please provide your refactored design.

(b) (7 %) Suppose we want to allow the user of our program to write custom Action to perform when some data source is to be updated. Please apply the command pattern for this requirement. Provided that we have the following Logger class, please provide a concrete logger action.

```
public class Logger {
    public void writeMessage (String message) {
        // logic to write the log message
    }
}
```

7. (30 %) You have been assigned to design an information system for managing professional conferences. The most important task in organizing a professional conference is to select the papers for presentation in the conference. This may be broken down into receiving paper submissions, reviewing the papers, and making the decisions (regarding which papers to accept and which to reject). All these are now typically carried out on-line. Among other things, you have managed to gather the following requirements:

(a) The system is envisioned to run for several years, possibly managing many conferences at a given time.

(b) A conference has a name, mostly also an acronym, and the dates and the place it will be held.

(c) A conference also has a program committee (PC) that is responsible for selecting the papers.

(d) A PC has at least one chair and several members; the chair is also a member.

(e) Every submission/paper should have a title and one or more authors.

(f) A submission should also identify the relevant subjects by giving appropriate keywords.

(g) Every PC member should identify his/her subjects of interest, also by giving appropriate keywords.

(h) The PC chair(s) must not submit a paper.

(i) For the reviewing, a paper should be assigned to at least three PC members, whenever possible with a subject of interest also identified by the paper.

(j) The PC members and the authors might be related by an adviser-advisee or co-authoring relationship in the past.

(k) A submission should not be assigned to a PC member who has an adviser-advisee or co-authoring relationship with one of the authors of the submission within the past five years.

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