Suggested Solutions to Midterm Problems

1. Prove the following sequents using Gentzen's System LK. You may take the set view of a sequent to shorten your proofs. (10 %)

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

   Solution. (Ming-Hsien Tsai)

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\;\text{Axiom} \qquad \dfrac{}{B, A \vdash B}\;\text{Axiom}}{A \to B, A \vdash B}\;\to\text{: Left}}{A \to B, \neg B, A \vdash}\;\neg\text{: Left}}{A \to B, \neg B \vdash \neg A}\;\neg\text{: Right}}{A \to B \vdash \neg B \to \neg A}\;\to\text{: Right}}{\vdash (A \to B) \to (\neg B \to \neg A)}\;\to\text{: Right}
$$

□

   (b) $\vdash \exists x(P(x) \land Q(x)) \to (\exists y P(y) \land \exists z Q(z))$

   Solution. (Ming-Hsien Tsai)

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P(w), Q(w) \vdash P(y)[w/y]}\;\text{Axiom}}{P(w), Q(w) \vdash \exists y P(y)}\;\exists\text{: Right} \qquad \dfrac{\dfrac{}{P(w), Q(w) \vdash Q(z)[w/z]}\;\text{Axiom}}{P(w), Q(w) \vdash \exists z Q(z)}\;\exists\text{: Right}}{P(w), Q(w) \vdash (\exists y P(y) \land \exists z Q(z))}\;\land\text{: Right}}{(P(x) \land Q(x))[w/x] \vdash (\exists y P(y) \land \exists z Q(z))}\;\land\text{: Left}}{\exists x(P(x) \land Q(x)) \vdash (\exists y P(y) \land \exists z Q(z))}\;\exists\text{: Left}}{\vdash \exists x(P(x) \land Q(x)) \to (\exists y P(y) \land \exists z Q(z))}\;\to\text{: Right}
$$

□

2. 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)
   - $\forall a ((a \cdot e = a) \land (e \cdot a = a))$. (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 Gentzen's System LK the sequent $M \vdash \forall e' (\forall a ((a \cdot e' = a) \land (e' \cdot a = a)) \to e' = e)$, 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, $e' = e' \cdot e = e$.) (15 %)

   Solution. (Ming-Hsien Tsai)

   Let M_1 denote $\forall a \forall b \forall c (a \cdot (b \cdot c) = (a \cdot b) \cdot c)$ and M_2 denote $\forall a ((a \cdot e = a) \land (e \cdot a = a))$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash e = e}\,\text{Axiom} \quad \dfrac{}{w \cdot e = w, w \cdot e = e, e = e \vdash w = e}\,\text{Axiom}^1}{w \cdot e = w, w \cdot e = e \vdash w = e}\,\text{Cut}}{M_1, w \cdot e = w, e \cdot w = w, e \cdot w = e, w \cdot e = e \vdash w = e}\,\text{Weakening: Left}}{M_1, ((a \cdot e = a) \wedge (e \cdot a = a))[w/a], e \cdot w = e, w \cdot e = e \vdash w = e}\,\wedge\text{: Left}}{M_1, M_2, e \cdot w = e, w \cdot e = e \vdash w = e}\,\forall\text{: Left}}{M_1, M_2, ((a \cdot w = a) \wedge (w \cdot a = a))[e/a] \vdash w = e}\,\wedge\text{: Left}}{M_1, M_2, \forall a((a \cdot w = a) \wedge (w \cdot a = a)) \vdash w = e}\,\forall\text{: Left}}{M_1, M_2 \vdash (\forall a((a \cdot e' = a) \wedge (e' \cdot a = a)) \rightarrow e' = e)[w/e']}\,\rightarrow\text{: Right}}{M_1, M_2 \vdash \forall e'(\forall a((a \cdot e' = a) \wedge (e' \cdot a = a)) \rightarrow e' = e)}\,\forall\text{: Right}$$

Note 1: For binary predicate $P(x, y)$, we have the following axiom.

$$\dfrac{}{s_1 = t_1, s_2 = t_2, P(s_1, s_2) \vdash P(t_1, t_2)}\,\text{Axiom}$$

By choosing $P(x, y)$ as $x = y$, s_1 as $w \cdot e$, s_2 as e, t_1 as w, and t_2 as e. The following is also an axiom,

$$\dfrac{}{w \cdot e = w, e = e, w \cdot e = e \vdash w = e}\,\text{Axiom}$$

, which is the same as

$$\dfrac{}{w \cdot e = w, w \cdot e = e, e = e \vdash w = e}\,\text{Axiom}$$

□

3. For each of the following informal requirement descriptions, write a formal specification in the form of "$\{pre\}\ y := ?\ \{post\}$" where y is a single variable or a list of variables that may be changed by the program. (10 %)

   (a) The input is an array A of size n. The output d is 1 if all elements of A are distinct; otherwise, the output d is 0.

   Solution. (Ming-Hsien Tsai)

   Assume *size* is a function which counts the size of a given array. We can define the predicate *distinct*.

   $$distinct \triangleq \forall i \forall j (1 \leq i \leq n \wedge 1 \leq j \leq n \wedge i \neq j \rightarrow A[i] \neq A[j])$$

   The specification would be:

   $\{size(A) = n\}$ d := ? $\{(distinct \wedge d = 1) \vee (\neg distinct \wedge d = 0)\}$ □

(b) The inputs are two sorted arrays A and B of sizes m and n respectively. Assuming that A and B have common elements, the output x is the smallest common element of A and B.

Solution. (Ming-Hsien Tsai)

Assume *size* is a function which counts the size of a given array. We can define two predicates as follows.

$$\begin{aligned} \mathit{in}(x, A) &\triangleq \exists i(1 \leq i \leq \mathit{size}(A) \wedge A[i] = x) \\ \mathit{common}(A, B) &\triangleq \exists x(\mathit{in}(x, A) \wedge \mathit{in}(x, B)) \end{aligned}$$

The specification would be:

$\{\mathit{size}(A) = m \wedge \mathit{size}(B) = n \wedge \mathit{common}(A, B)\}$ x := ? $\{\mathit{in}(x, A) \wedge \mathit{in}(x, B) \wedge \forall y(\mathit{in}(y, A) \wedge \mathit{in}(y, B) \rightarrow x <= y)\}$ □

4. Prove the (partial) correctness of the following program. (20 %)

```
{(x = n) ∧ (n ≥ 0)}
S1: y := 0;
S2: while x > 0 do
   S3: y := y + (2x − 1);
   S4: x := x − 1
od
{y = n²}
```

Solution. (Ming-Hsien Tsai)

$$\dfrac{A \qquad \dfrac{\text{pred. calculus + algebra}}{x \geq 0 \wedge y = n^2 - x^2 \wedge \neg(x > 0) \rightarrow y = n^2}}{\{x = n \wedge n \geq 0\}\ \mathrm{S1};\ \mathrm{S2}\ \{y = n^2\}}\ \text{WP}$$

A:

$$\dfrac{\dfrac{C \qquad \dfrac{}{\{x \geq 0 \wedge 0 = n^2 - x^2\}\ \mathrm{S1}\ \{x \geq 0 \wedge y = n^2 - x^2\}}\ \text{Assign}}{\{x = n \wedge n \geq 0\}\ \mathrm{S1}\ \{x \geq 0 \wedge y = n^2 - x^2\}}\ \text{SP} \qquad B}{\{x = n \wedge n \geq 0\}\ \mathrm{S1};\ \mathrm{S2}\ \{x \geq 0 \wedge y = n^2 - x^2 \wedge \neg(x > 0)\}}\ \text{Sequence}$$

B:

$$\dfrac{D \qquad \dfrac{}{\{x-1 \geq 0 \wedge y = n^2 - (x-1)^2\}\ \mathrm{S4}\ \{x \geq 0 \wedge y = n^2 - x^2\}}\ \text{Assign}}{\dfrac{\{x \geq 0 \wedge y = n^2 - x^2 \wedge x > 0\}\ \mathrm{S3};\ \mathrm{S4}\ \{x \geq 0 \wedge y = n^2 - x^2\}}{\{x \geq 0 \wedge y = n^2 - x^2\}\ \mathrm{S2}\ \{x \geq 0 \wedge y = n^2 - x^2 \wedge \neg(x > 0)\}}\ \text{While}}\ \text{Sequence}$$

C:

$$\dfrac{\text{pred. calculus + algebra}}{x = n \wedge n \geq 0 \rightarrow x \geq 0 \wedge 0 = n^2 - x^2}$$

D:

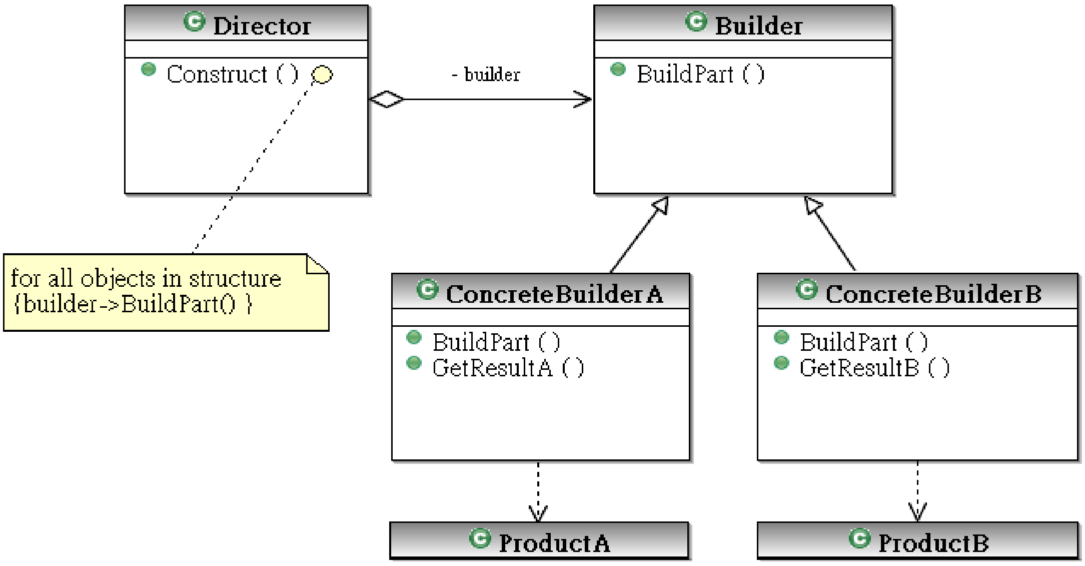
$$\dfrac{E \qquad \dfrac{}{\{x-1 \geq 0 \wedge y + (2x-1) = n^2 - (x-1)^2\}\ \mathrm{S3}\ \{x-1 \geq 0 \wedge y = n^2 - (x-1)^2\}}\ \text{Assign}}{\{x \geq 0 \wedge y = n^2 - x^2 \wedge x > 0\}\ \mathrm{S3}\ \{x-1 \geq 0 \wedge y = n^2 - (x-1)^2\}}$$

E:

$$\dfrac{\text{pred. calculus + algebra}}{x \geq 0 \wedge y = n^2 - x^2 \wedge x > 0 \rightarrow x - 1 \geq 0 \wedge y + (2x-1) = n^2 - (x-1)^2}$$

□

5. Describe as complete as possible in words what the following UML diagram is specifying: (10 %)



Solution. (Chi-Jian Luo)

- *Builder*:
  - specifies an abstract interface for creating parts of a Product object.
  - includes an abstract method BuildPart().
- *ConcreteBuilderA* and *ConcreteBuilderB*:
  - constructs and assembles parts of the product by implementing the Builder interface.
  - defines and keeps track of the representation it creates.
  - provides an interface for retrieving the product.
  - includes two methods BuildPart() and GetResult().
- *Director*:
  - constructs an object using the *Builder* interface.
  - includes a method Construct().
- *ProductA* and *ProductB*:
  - represents the complex object under construction. *ConcreteBuilder* builds the product's internal representation and defines the process by which it's assembled.
  - includes classes that define the constituent parts, including interfaces for assembling the parts into the final result.
- *Aggregation builder*:
  - each *Director* has a *Builder.*
  - when a *Director* is destroyed, the *Builder* may continue to exist.
- *Generalization* between *Builder* and *ConcreteBuilder*:
  - each *ConcreteBuilder* has all methods of the *Builder.*
  - the methods must be implemented.
- *Denendency* between *ConcreteBuilder* and *Product*:
  - a change to the definition of *Product* will result in a change to *ConcreteBuilder.*

□

6. Suppose you are designing a programmable roving robot. The robot has the following classes for its sensor and controller:

- `GPS`: a Global Positioning System class which supports the method `locate (double & longitude, double & latitude)`.
- `Stepper`: a robot mobility unit which supports the method `move (double & dx, double & dy)`.
- `Arm`: a robotic arm which supports the methods `retrieve (void)` and `deliver (void)`.

Suppose the class `Robot` has the following definition:

```
class Robot {
  public:
    GPS gps;
    Stepper stepper;
    Arm arm;
    ...
};
```

Your goal is to define its programmable interface so that the robot can follow instructions programmed by users. Answer the following questions by writing pseudo C++ or Java code.

(a) Please use the Command pattern to define the class hierarchy and all classes that support the following instructions: (5 %)

- `GOTO X, Y`: goes to the absolute location (`X`, `Y`).
- `GET`: extends its arm to retrieve item(s).
- `PUT`: extends its arm to deliver item(s).

Solution.

```
class Command {
  public:
    virtual void Execute () = 0;
  protected:
    virtual Command ();
};

class GotoCommand : public Command {
  public:
    GotoCommand (GPS *gps, Stepper *stepper, int x, int y);
    virtual void Execute ();
  private:
    GPS *_gps;
    Stepper *_stepper;
    int _x, _y;
};

class GetCommand : public Command {
  public:
    Get (Arm *arm);
    virtual void Execute ();
  private:
    Arm *_arm;
};

class PutCommand : public Command {
  public:
    Put (Arm arm);
    virtual void Execute ();
  private:
    Arm *_arm;
};
```

□

(b) Suppose we would like the robot to accept a sequence of instructions. How would you modify your design? (5 %)

Solution. Use macro commands.

```
class MacroCommand : public Command {
  public:
    MacroCommand ();
    virtual ~MacroCommand ();

    virtual void Add (Command *);
    virtual void Remove (Command *);

    virtual void Execute ();
  private:
    List<Command *> *_cmds;
};
```

□

(c) After programming the robot a while, we realize macro instructions such as `GET X, Y` and `PUT X, Y` are very useful. How would you add them in your design? (5 %)

Solution.

```
class PosGetCommand : public MacroCommand {
  public:
    PosGetCommand (GPS *gps, Stepper *stepper, Arm *arm,
                   int x, int y) {
      GotoCommand goto (gps, stepper, x, y);
      GetCommand get (arm);

      Add (&goto);
      Add (&get);
    }
};

class PosPutCommand : public MacroCommand {
  public:
    PosPutCommand (GPS *gps, Stepper *stepper, Arm *arm,
                   int x, int y) {
      GotoCommand goto (gps, stepper, x, y);
      PutCommand put (arm);

      Add (&goto);
      Add (&put);
    }
};
```

□

(d) Consider the following sequence of instructions: `GET 0.0, 0.0`, `PUT 1.0, 1.0`, `GET 0.0, 0.0`, `PUT 2.0, 2.0`. Clearly, it is more efficient if the robot retrieves all items at once. That is, it executes the following instructions instead: `GET 0.0, 0.0`, `GET 0.0, 0.0`, `PUT 1.0, 1.0`, `PUT 2.0, 2.0`. Which design pattern would you use to make it smarter? Please modify your design properly. (5 %)

Solution. Use Chain of Responsibility.

```
class TaskHandler {
  public:
    TaskHandler (TaskHandler *s) : _successor (s) { }
    virtual void HandleTask () {
      if (_successor) { _successor->HandleTask (); }
    }
  private:
    TaskHandler *_successor;
};

class GotoCommand : public Command, public TaskHandler {
  public:
    GotoCommand (GPS *gps, Stepper *stepper, int x, int y,
                 TaskHandler *s);
    virtual void Execute ();
    ...
};
```

□

7. A simplistic car consists of an engine, a steering wheel, a brake, and wheels. Suppose you are designing a car controller which can be used in Ford Anglia of years 1962 and 1963.

   (a) Among Abstract Factory, Builder, Factory Method, and Prototype patterns, which one would you choose in your design? Please explain your design decision in English or Chinese. (5 %)

   Solution. Pick any one of Abstract Factory, Builder, Factory Method, and Prototype patterns with any explanation. □

   (b) Please realize your design choice in pseudo C++ or Java code. (5 %)

   Solution. Implement your choice properly. □

   (c) Since there can be at most four wheels in a car, please modify your code to ensure others would not create more wheels unwillingly. (5 %)

   Solution. Use Singleton on wheels.

```
class Wheel {
  public:
    static Wheel *Instance () {
      if (counter > 0) {
        --counter;
        return new Wheel ();
      }
    }
  protected:
    Wheel ();
  private:
    static int counter;
};
int Wheel::counter = 4;
```

□

Appendix

- System LK: Axioms for Equality

  Let $t, s_1, \cdots, s_n, t_1, \cdots, t_n$ be arbitrary terms.

  $$\frac{}{\vdash t = t}$$

  For every n-ary function f,

  $$\frac{}{s_1 = t_1, \cdots, s_n = t_n \vdash f(s_1, \cdots, s_n) = f(t_1, \cdots, t_n)}$$

  For every n-ary predicate P (including =),

  $$\frac{}{s_1 = t_1, \cdots, s_n = t_n, P(s_1, \cdots, s_n) \vdash P(t_1, \cdots, t_n)}$$

  Note: The = sign is part of the object language, not a meta symbol.