SE 2AA4 Winter 2007

Final Examination Answer Key

Instructor: William M. Farmer

Revised: 21 April 2007

(1) [2 pts.] A typical programming language has a mathematically precise semantics. Is this statement true or false?

A.) True.

B.) [✓ False.]

(2) [2 pts.] It is a sign of good software design if each module is used by no more than one other module. Is this statement true or false?

A.) True.

B.) [✓ False.]

(3) [2 pts.] Unlike Oberon, modules cannot be directly implemented in C and Java. Is this statement true or false?

A.) [✓ True.]

B.) False.

(4) [2 pts.] Software development usually follows the waterfall model closely. Is this statement true or false?

A.) True.

B.) [✓ False.]

(5) [2 pts.] Refinement is another name for rapid prototyping. Is this statement true or false?

A.) True.

B.) [✓ False.]

(6) [2 pts.] A procedure that is totally correct which respect to a pre- and post condition specification S is always partially correct with respect to S as well. Is this statement true or false?

A.) [✓ True.]

B.) False.

(7) [2 pts.] A before/after MIS specifies the interface of a module as a state machine. Is this statement true or false?

A.) [✓ True.]

B.) False.

(8) [2 pts.] In Java, every throwable object must satisfy the Catch or Specify Requirement for exceptions. Is this statement true or false?

A.) True.

B.) [✓ False.]

(9) [2 pts.] A specification is usually written before a product is implemented, while a description is usually written after a product is implemented. Is this statement true or false?

A.) [✓ True.]

B.) False.

(10) [2 pts.] A software product without a requirements specification cannot be considered correct nor can it be considered incorrect. Is this statement true or false?

A.) [✓ True.]

B.) False.

(11) [2 pts.] Which kind of software verification is usually the most expensive?

A.) Blackbox testing.

B.) Clearbox testing.

C.) Product inspection.

D.) [✓ Mathematical verification.]

(12) [2 pts.] The development of a software product's architecture is done as part of the

A.) Requirements phase.

B.) [✓ Design phase.]

C.) Implementation phase.

D.) Verification phase.

(13) [2 pts.] Testing is most effective for showing

A.) Correctness of an implementation.

B.) [✓ Incorrectness of an implementation.]

C.) Trustworthiness of an implementation.

D.) That an implementation satisfies a specification.

(14) [2 pts.] As a rule, procedures in a functional programming language do not

A.) Use recursion.

B.) [✓ Have side-effects.]

C.) Have return values.

D.) All of the above.

(15) [2 pts.] Which kind of module is an exception to the rule that the interface of a module should be small and orthogonal?

A.) Object.

B.) Abstract data structure.

C.) [✓ Definitional extension.]

D.) All of the above.

(16) [2 pts.] Which software engineering principle is used in modular design?

A.) Separation of concerns.

B.) Abstraction.

C.) Anticipation of change.

D.) [✓ All of the above.]

(17) [2 pts.] The state of a data structure can be changed using a

A.) Constructor.

B.) Selector.

C.) [✓ Mutator.]

D.) Field.

(18) [2 pts.] The access privileges assigned to a Unix file are grouped into three read-write-execute lists. The first list gives the access privileges granted to

A.) The root account.

B.) The accounts in the group assigned to the file.

C.) [✓ The account that owns the file.]

D.) The account that executes the file.

(19) [2 pts.] Which software design strategy can significantly raise the maintainability of a software component?

A.) Design for change.

B.) Product families.

C.) Little languages.

D.) [✓ All of the above.]

(20) [2 pts.] Which software design strategy can significantly raise the reusability of a software component?

A.) Refinement.

B.) Transformation.

C.) [✓ Little theories.]

D.) All of the above.

(21) [2 pts.] A macro uses ____________ variable binding.

A.) [✓ Call-by-name.]

B.) Call-by-value.

C.) Call-by-reference.

D.) All of the above.

(22) [2 pts.] The interface of a module is

A.) A language of types, constants, procedures, exceptions, etc.

B.) A set of services.

C.) A contract between the module and the other modules that use it.

D.) [✓ All of the above.]

(23) [2 pts.] The Java programming language is portable because

- A.) It is object oriented.
- B.) It employs a C-like syntax.
- C.) It has garbage collection.
- D.) [✓ It can be compiled to easily interpreted byte code.]

(24) [2 pts.] Which of the following giants of computing did not win a Turing Award?

- A.) Donald Knuth.
- B.) [✓ Gottfried Leibnitz.]
- C.) Tony Hoare.
- D.) John Backus.

(25) [2 pts.] Let $\mathbf{N}$ denote the set of natural numbers. What is the value of the expression

$$(\lambda\, x : \mathbf{N} \,.\, (\lambda\, y : \mathbf{N} \,.\, x^y))(2)(3)$$

after applying beta-reduction as many times as possible?

- A.) $(\lambda\, y : \mathbf{N} \,.\, 2^3)$.
- B.) $(\lambda\, y : \mathbf{N} \,.\, 3^2)$.
- C.) $\boxed{2^3.}$
- D.) 3^2.

(26) [10 pts.] The *magnitude* of a vector v, written $|v|$, is the length of v. Two vectors u and v are *parallel*, written $u \mid\mid v$, if u and v have the same direction or have exactly opposite directions. Let `Vector` be a type of vectors. The formula

$$\begin{array}{l} \forall\, v : \texttt{Vector} \,. \\ \quad \mathsf{if}(|v| = 0, \\ \quad\; \texttt{mouse}(v)\uparrow \quad [\rightsquigarrow \texttt{ZeroVectorException}], \\ \quad\; |\texttt{mouse}(v)| = 1 \wedge \texttt{mouse}(v) \mid\mid v) \end{array}$$

is an axiomatic input/output specification for the Java method

```
public static Vector mouse(Vector v) throws ZeroVectorException;
```

Recall that that the interface of the VectorPlus class of Exercise 4 contains the following Java methods:

```
public float getX();
public float getY();
public static Vector iVector();
public static Vector jVector();
public static Vector mul(float r, Vector v);
public static Vector add(Vector u, Vector v);
public static float getMagnitude(Vector v);
public static float getAngle(Vector v) throws ZeroVectorException;
public static Vector zeroVector();
public static float dot(Vector u, Vector v);
```

Using the services provided by the VectorPlus class, write an implementation of mouse that satisfies the axiomatic input/output specification given above. Comments are not necessary. Points will be taken off for any irrelevant code.

Answer:

```
public static Vector mouse(Vector v) throws ZeroVectorException {
    float m = VectorPlus.getMagnitude(v);
    if (m == 0)
        throw new ZeroVectorException(v);
    else
        return VectorPlus.mul(1/m,v);
}
```

(27) [20 pts.] Below is a before/after MIS for a module that stores a circle. Write a complete module in C, consisting of a header file `circle.h` and a code file `circle.c`, that implements the MIS. Comments are not necessary. Points will be taken off for any irrelevant code.

Before/after MIS:

- Module name: `circle`.
- Imported modules: None required.
- Interface:

```
procedure center_x(): float;
procedure center_y(): float;
procedure radius(): float;
procedure resize(m: float);
procedure reposition(a,b: float);
exception NegRadius;
```

- State constants: None required.
- State variables:

  x : `float` [initially $x = 0$].
  y : `float` [initially $y = 0$].
  r : `float` [initially $r = 0$].

- Behavior rules:

Name	**Input**	**Output**	**Transition**	**Exception**
`center_x`		x		
`center_y`		y		
`radius`		r		
`resize`	m : `float`		$r' = m * r$	$r' < 0 \rightsquigarrow$ `NegRadius`
`reposition`	a, b : `float`		$x' = x + a$ $y' = y + b$	

Answer:

```
/* Start of circle.h */
float center_x();
float center_y();
float radius();
void resize(float m);
void reposition(float a, float b);
/* End of circle.h */

/* Start of circle.c */
#include <stdio.h>

static float x = 0;
static float y = 0;
static float r = 0;

float center_x() {
  return x;
}

float center_y() {
  return y;
}

float radius() {
  return r;
}

void resize(float m) {
  r = m * r;
  if (r < -1)
    printf("Error in resize(%f): New radius is negative.\n", m);
}

void reposition(float a, float b) {
  x = x + a;
  y = y + a;
}
/* End of circle.c */
```

(28) [20 pts.] Recall that the interface of the `List` class of Exercise 5 contains the following Java methods:

```
public Element getMember(int i)
  throws BadIndexException;

public static List nil();

public static List cons(Element e, List k);

public static List take(int i, List k)
  throws BadIndexException;

public static List drop(int i, List k)
  throws BadIndexException;

public boolean same(Element e);

public String toString();
```

Write a complete class in Java named `Set` that implements an abstract data type of *sets* represented as objects of type `List`. The class `Set` should implement the interface `Element` so that sets of sets can be constructed and should contain no public fields and only the following public methods:

- A constructor

  ```
  public static Set empty();
  ```

  that creates the object of type `Set` that represents the empty set.

- A constructor

  ```
  public static Set adjoin(Element e, Set s);
  ```

  that creates the object of type `Set` obtained by adding `e` to `s`. (This object is `s` itself if `e` is already a member of `s`.)

- A constructor

  ```
  public static Set union(Set s, Set t);
  ```

  that creates the object of type `Set` that is the union of `s` and `t`.

- A predicate

  ```
  public boolean isMember(Element e);
  ```

  that is true iff `e` is a member of this object.

- The methods

```
public boolean same(Element e);
```

and

```
public String toString();
```

from the `Element` interface.

Comments are not necessary. Points will be taken off for any irrelevant code.

Answer:

```
package exercise5;

public class Set implements Element {

    private List list_;

    private Set(List k) {
        list_ = k;
    }

    public static Set empty() {
        return new Set(List.nil());
    }

    public static Set adjoin(Element e, Set s) {
        if (s.isMember(e))
            return s;
        else
            return new Set(List.cons(e,s.list_));
    }

    public static Set union(Set s, Set t) {
        try {
            if (s.same(empty()))
                return t;
            else
                return union(new Set(List.drop(1,s.list_)),
                             adjoin(s.list_.getMember(0),t));
        }
        catch (BadIndexException x) {
            // BadIndexException should not occur.
            System.out.println(x.toString());
            return empty();
        }
    }

    public boolean isMember(Element e) {
        try {
            return !same(empty())
                && (list_.getMember(0).same(e)
                    || (new Set(List.drop(1,list_))).isMember(e));
        }
        catch (BadIndexException x) {
            // BadIndexException should not occur.
            System.out.println(x.toString());
            return false;
        }
    }

    public boolean same(Element e) {
        return e != null
            && e instanceof Set
            && list_.same(((Set)e).list_);
    }

    public String toString() {
        return list_.toString();
    }

}
```