Name ______________________________

Student number ______________________________

SE 2A04 Fall 2002

Midterm Test 2 Answer Key

Instructor: William M. Farmer

You have 50 minutes to complete this test consisting of 8 pages and 11 questions. You may use your notes and textbooks. Circle the *best* answer for the multiple choice questions, and write the answer to the other questions in the space provided. You are welcome to write your answers in pencil. However, the instructor will not remark answers written in pencil if he thinks it is *possible* that the answer was erased and rewritten. Good luck!

(1) [5 pts.] The Oberon-2 programming language allows local modules to be declared within a module. Is this statement true or false?

(a) True.

(b) $\boxed{\text{False.}}$

(2) [5 pts.] A specification is usually considered to be less abstract than a description. Is this statement true or false?

(a) True.

(b) $\boxed{\text{False.}}$

(3) [5 pts.] Let S_1, S_2, and S_3 be specifications such that S_2 is a refinement of S_1 and S_3 is a refinement of S_2. If a product P satisfies S_3, it necessarily satisfies S_1. Is this statement true or false?

(a) $\boxed{\text{True.}}$

(b) False.

(4) [5 pts.] The following is a relational specification of a function.

$$R = \{(x, y) \in \mathbf{Z} \times \mathbf{Z} \mid x = y \vee x = -y\}.$$
$$D = \mathbf{Z}.$$

How many functions satisfy this specification?

(a) 0.
(b) 1.
(c) 2.
(d) [✓ Infinitely many.]

(5) [5 pts.] Which of the following should usually not be a component of a module interface?

(a) A type.
(b) [✓ A variable.]
(c) A constant.
(d) A procedure.

(6) [5 pts.] What is the "secret" of the `Vectors` module of Lab Exercise 3?

(a) How vectors are represented.
(b) How many vectors are stored.
(c) When vectors are created.
(d) [✓ All of the above.]

(7) [5 pts.] Suppose a software system consists of the modules

$$M_0, M_1, \ldots, M_k.$$

Which change to M_0 would *not* require any changes to the other modules $M_1, \ldots, M_k$?

(a) Removing a component from the interface of M_0.
(b) Modifying the specification of the interface of M_0.
(c) [✓ Reimplementing a procedure in M_0.]
(d) All of the above.

(8) [5 pts.] The `Vector` module of Lab Exercise 2 that is implemented using cartesian coordinates is a finite state machine. If n is the size of the type `REAL`, how many possible states can the module have?

(a) n.

(b) $2 * n$.

(c) $\boxed{n^2.}$

(d) n^3

(9) [20 pts.] Let `VecPlay` be an Oberon-2 module that imports the `Math` module and that contains the following declarations:

```
TYPE

  Vector = POINTER TO VectorRec;

  VectorRec =
    RECORD
      x: REAL;
      y: REAL;
    END;

CONST max = 1000;

VAR VecArray: ARRAY max OF Vector;
```

Each member of the type `Vector` represents a two-dimensional vector; `NIL` represents the zero vector. `VecArray` stores 1000 vector representations. For each v : `Vector`, let $|v|$ denote the magnitude of the vector represented by v. In this module `VecPlay`, write a single procedure named `MaxMagInVecArray` that takes no input and returns the maximum of the set

$$\{|v| \mid v \text{ is stored in } \texttt{VecArray}\}.$$

Answer:

```
PROCEDURE MaxMagInVecArray(): REAL;
  VAR i: INTEGER;
      m,n: REAL;
      v: Vector;
BEGIN
  FOR i := 0 TO max - 1 DO
    v := VecArray[i];
    IF v # NIL THEN
      n := Math.Sqrt((v^.x * v^.x) + (v^.y * v^.y));
      IF m < n THEN m := n END;
    END;
  END;
  RETURN m
END MaxMagInVecArray;
```

(10) [20 pts.] Below is an axiomatic input/output MIS for a definitional extension of the `Vectors` module of Lab Exercise 3. Write a complete Oberon-2 module named `VectorsExt` that implements the MIS. Points will be taken off for any irrelevant extra code.

Axiomatic input/output MIS

(a) Imported modules: `Vectors`.

(b) Interface:

```
INTERFACE VectorsExt;
  PROCEDURE Dot(u,v: Vectors.Vector): REAL;
  PROCEDURE LexOrd(u,v: Vectors.Vector): BOOLEAN;
END VectorsExt.
```

(c) Exceptions: none required.

(d) Axioms:

- $\forall\, u, v : \texttt{Vectors.Vector}\ .\ \texttt{Dot}(u, v)) =$
  $(\texttt{Vectors.Xval}(u) * \texttt{Vectors.Xval}(v)) +$
  $(\texttt{Vectors.Yval}(u) * \texttt{Vectors.Yval}(v))$.
- $\forall\, u, v : \texttt{Vectors.Vector}\ .\ \texttt{LexOrd}(u, v)) \Leftrightarrow$
  $[\texttt{Vectors.Xval}(u) < \texttt{Vectors.Xval}(v) \vee$
  $[\texttt{Vectors.Xval}(u) = \texttt{Vectors.Xval}(v) \wedge$
  $\texttt{Vectors.Yval}(u) < \texttt{Vectors.Yval}(v)]]$.

Answer:

```
MODULE VectorsExt;

IMPORT Vectors;

PROCEDURE Dot*(u,v: Vectors.Vector): REAL;
BEGIN
  RETURN (Vectors.Xval(u) * Vectors.Xval(v)) +
         (Vectors.Yval(u) * Vectors.Yval(v))
END Dot;

PROCEDURE LexOrd*(u,v: Vectors.Vector): BOOLEAN;
BEGIN
  RETURN (Vectors.Xval(u) < Vectors.Xval(v)) OR
         ((Vectors.Xval(u) = Vectors.Xval(v)) &
          (Vectors.Yval(u) < Vectors.Yval(v)))
END LexOrd;

END VectorsExt.
```

(11) [20 pts.] Below is a before/after MIS for a module that simulates a clothes dryer. Write a complete Oberon-2 module named `Dryer` that implements the MIS. Implement the `DoorClosed` exception as a warning. Points will be taken off for any irrelevant extra code.

Before/after MIS:

(a) Imported modules: none required.

(b) Interface:

```
INTERFACE Dryer;
  PROCEDURE Start();
  PROCEDURE CloseDoor();
  PROCEDURE OpenDoor();
  PROCEDURE InsertFilter();
  PROCEDURE RemoveFilter();
END Dryer.
```

(c) Exceptions: `DoorClosed: BOOLEAN`.

(d) State constants: none.

(e) State variables:
c : `BOOLEAN` [initially $c =$ `TRUE`] (c means door is closed).
i : `BOOLEAN` [initially $i =$ `TRUE`] (i means filter is in).
r : `BOOLEAN` [initially $r =$ `FALSE`] (r means dryer is running).

(f) Behavior rules:

Name	**Transition**	**Exception**
`Start`	$c' = c$ $i' = i$ $r' = \mathsf{if}(c \wedge i, \mathtt{TRUE}, \mathtt{FALSE})$	
`CloseDoor`	$c' = \mathtt{TRUE}$ $i' = i$ $r' = r$	
`OpenDoor`	$c' = \mathtt{FALSE}$ $i' = i$ $r' = r$	
`InsertFilter`	$c' = c$ $i' = \mathsf{if}(\neg c, \mathtt{TRUE}, i)$ $r' = r$	$c \Rightarrow$ `DoorClosed`
`RemoveFilter`	$c' = c$ $i' = \mathsf{if}(\neg c, \mathtt{FALSE}, i)$ $r' = r$	$c \Rightarrow$ `DoorClosed`

Answer:

```
MODULE Dryer;

IMPORT Out;

VAR c, i, r: BOOLEAN;

PROCEDURE Reset();
BEGIN
  c := TRUE;
  i := TRUE;
  r := TRUE
END Reset;

PROCEDURE DoorClosedException();
BEGIN
  Out.String("Warning: Door is closed.")
END DoorClosedException;

PROCEDURE Start*();
BEGIN
  r := c & i
END Start;
```

```
PROCEDURE CloseDoor*();
BEGIN
  c := TRUE
END CloseDoor;

PROCEDURE OpenDoor*();
BEGIN
  c := FALSE
END OpenDoor;

PROCEDURE InsertFilter*();
BEGIN
  IF c THEN
    DoorClosedException()
  ELSE
    i := TRUE
  END
END InsertFilter;

PROCEDURE RemoveFilter*();
BEGIN
  IF c THEN
    DoorClosedException()
  ELSE
    i := FALSE
  END
END RemoveFilter;

BEGIN
  Reset
END Dryer.
```