Instructions

1. Read the question paper carefully and answer all the questions below. Write your answers in the provided answer booklet. Before you begin, make sure that you fill in all your personal details correctly and clearly on the front cover of the answer book.

2. This evaluation opportunity comprises of four (4) questions on three (3) pages. Please check that you have received the entire paper before you begin writing.

3. This is a closed book paper. You are not allowed to have any literature with you when writing this test.

4. Switch off your cell phone, and keep it off for the duration of the test. No electronic devices of any form are allowed to be used for the duration of the test.

5. Plagiarism or cheating of any form will not be tolerated. If you are found guilty of any such transgressions, disciplinary action will be taken. This may include suspension of your studies at the University of Pretoria.

6. Number your questions clearly and according to the numbering scheme provided in this question paper (unless otherwise instructed). Provide a clear separation between each of the 5 main questions, preferably with a heading at the start of each. Clearly number each of the sub-questions under each main question.

7. Write clearly and concisely. Use bulleted lists to clearly organize an answer, if you are asked to discuss multiple points. If you need to continue an answer on another page, please indicate this clearly.

8. For all of the test’s questions, you may provide program code to illustrate your point. It is, however, not required to provide program code, as long as your answer is clear enough.
Question 1: Expressions and Assignment Statements (12 marks)

1. **Differentiate** between operator precedence rules, and operator associativity rules. [2]
2. **Describe** how Ruby implements operators. [1]
3. **Describe** two (2) different ways in which a functional side effect can occur. [2]
4. Consider the concept of operator overloading, and answer the following questions:
   (a) **Describe** one (1) problem that can occur when the programming language itself (i.e. not the user of the programming language) overloads operators. [1]
   (b) **Describe** one (1) problem that can occur when the user of a programming language (i.e. not the programming language itself) overloads operators. [1]
5. **Explain** Ada’s approach to type conversions in expressions and assignment statements. [2]
6. **Contrast** the representation of Boolean values in early versions of C (i.e. C89) and Java. [2]
7. **Give an example** of a situation in which the lack of short-circuit operators can cause a problem in a user’s program code. You may simply describe the situation, but it will probably be easier to provide program code or pseudocode. If you provide code, your exact syntax will not be marked, as only the basic structure of your example is important. Make sure that your answer is clear enough to understand, and add notes for clarity, if you need to. [1]

Question 2: Statement-Level Control Structures (13 marks)

1. **Explain** what two (2) language features must be present in a language if a control statement has multiple entry points. [2]
2. Consider the following pseudocode program example, and answer the following questions:
   
   ```
   if (condition1)
     if (condition2)
       statement1
     else
       statement2
   ```

   (a) **Describe** a problem that arises when evaluating these statements. [1]
   (b) **Explain** how Java addresses the above-mentioned problem. [1]
   (c) **Explain** how Python addresses the above-mentioned problem. [1]
3. Consider the concept of multiple-way selection statements. **Explain** how C# ensures more reliable multiple-way selection statements than C++ does. [3]
4. **Contrast** how C++ and Ada determine the number of iterations that a counter-controlled loop will execute. [2]
5. **Contrast** the support for unconditional branching in C and Java. [2]
6. **Explain** what happens when more than one expression evaluates to true in the following example of Dijkstra’s guarded selection statement:
   ```
   if <expression1> -> <statement1>
   [ ] <expression2> -> <statement2>
   fi
   ```

Question 3: Subprograms (13 marks)

1. **Differentiate** between formal parameters and actual parameters. [2]
2. Consider the concept of keyword parameters, and answer the following questions:
   (a) **Describe** one advantage of keyword parameters. [1]
   (b) **Describe** one drawback to keyword parameters. [1]
3. **Differentiate** between how functions and procedures deal with their results. [3]
4. Describe one (1) disadvantage associated with pass-by-value parameters.

5. Consider the following pseudocode example of a subprogram definition:

```plaintext
sub(integer reference a, integer reference b) { ... }
```

Also consider the following call to this subprogram:

```plaintext
call sub(c, c)
```

Answer the following questions:

(a) Describe the main problem that is associated with this specific example, and under what circumstances this problem occurs.

(b) Suggest a possible solution to this problem.

6. Contrast how C++ and C# handle array dimensions when arrays are passed to subprograms.

**Question 4: ADTs and Encapsulation (12 marks)**

1. List the two (2) fundamental requirements that any abstract data type (ADT) must satisfy.

2. Explain what a property is in C#.

3. Classes in Ruby are dynamic. Explain what this concept entails.

4. Consider the concept of parameterized classes. Contrast how C++ and Java handle parameterized classes.

5. General encapsulation constructs (note: this question does not refer to naming encapsulations) are intended to support separate compilation. Explain how C supports separate compilation via encapsulation constructs.

6. Explain what the purpose of naming encapsulations is (note: this question does not refer to general encapsulations for separate compilation).