First name(s): ____________________________________________

Surname: ________________________________________________

Student number: _________________________________________

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 5 questions on 4 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 question, preferably with a heading at the start of each new question.

7. Write clearly and concisely. Use bulleted lists to clearly organise an answer, if you are asked to discuss multiple points. If you need to continue an answer on another page, please indicate this clearly.
Question 1: Preliminaries (5 marks)

1. One of the reasons for studying programming language concepts is to gain a better understanding of the significance of implementation. **Describe** one (1) reason that such an understanding could prove to be practically useful.

2. Consider two hypothetical programming languages, A and B. Both programming languages support several primitive data types and a record data type. In both languages, record fields may be of any type (including both primitive and record types). Programming language A allows a record R to contain a field whose type is itself R. Programming language B considers such nested composition to be an error.
   (a) **Define** the concept of orthogonality in a programming language.
   (b) **Compare** the orthogonality of these two languages, and **explain** your answer.

3. **Describe** one (1) reason that results in pure interpretation being much slower than standard compilation.

Question 2: Programming Language Evolution (10 marks)

1. **Name** the type of application area that COBOL was designed for.

2. Pseudocodes (e.g. Short Code and Speedcoding) were purely interpreted. Only later did high-level languages begin using the standard compilation process that is commonly used today. **Explain** why this was the case.

3. Consider each of the following languages. For each, **briefly explain** the single most important programming language concept introduced by the language:
   (a) SIMULA 67
   (b) LISP
   (c) Smalltalk

4. **Contrast** the approaches that were used by PL/I and ALGOL 68 when providing support for a wide variety of end-user data structure needs.

5. **Name** the programming language paradigm (category of programming language) that Prolog falls within.

Question 3: Names, Bindings and Scopes (13 marks)

1. In a programming language, special words may be either keywords or reserved words. **Differentiate** between keywords and reserved words in a programming language context.

2. Consider the following program code snippet from a hypothetical programming language:
   ```
   myVariable = 12;
   myVariable = "Test";
   ```
   Answer the questions that follow, in context of this programming language:
   (a) **Name** the category of type binding that this language uses for its variables.
   (b) **Name** the category of storage binding that this language uses for its variables

3. **Give an example** of a situation in which the scope and lifetime of a variable are not related to one another. **Make sure to explain** what the scope and lifetime are for the type of variable that you describe.

Page 2 of 4
4. Consider the following code snippet from a hypothetical programming language (note that this language allows nested subprograms):

```plaintext
main() {
    var X;
    func1() {
        var X;  \textit{(A)}
    }
    func2() {
        var Y;
        call func1();
    }
    call func2();
}
```

For each of the following types of scope, write down the referencing environment (including the subprogram scope of all variables in the referencing environment) at the point marked \textit{(A)} in the code:

(a) Static scoping.  \[3\]
(b) Dynamic scoping. \[3\]

**Question 4: Data Types (16 marks)**

1. Consider floating point and decimal data type representations of real numbers. Answer the following questions:
   (a) \textbf{Explain} the difference between the low-level representations of floating point and decimal data types. \[2\]
   (b) \textbf{Name} one (1) advantage that floating point data types have over decimal data types. \[1\]

2. Consider the concept of an array data type, and answer the following questions:
   (a) \textbf{Explain} the underlying representation that a programming language must use for multidimensional arrays, in order to make jagged arrays possible. \[1\]
   (b) \textbf{Explain} how associative arrays differ from normal arrays. \[2\]

3. \textbf{Explain} the difference between elliptical references and fully qualified references in terms of the following hypothetical example of a record:

```plaintext
record BankAccount {
    record AccountHolder {
        string firstName;
        string lastName;
    }
    real balance;
}
```

4. Consider the union data type, and answer the following questions:
   (a) \textbf{Explain} what a union data type is. \[1\]
   (b) \textbf{Name} the type of union that is generally considered to be unsafe. \[1\]

5. \textbf{Explain by means of an example} how a dangling pointer can occur in a language that supports pointers. \[4\]

6. \textbf{Explain} why references are generally considered a safer means of indirect addressing than pointers. \[2\]
Question 5: Expressions and Assignment Statements (6 marks)

1. Programming languages that allow many coercions tend to reduce their ability to type check. Explain how Ada addresses this problem. [1]

2. Consider the following program code snippet from an implementation in a hypothetical programming language (where the <> operator denotes the “not equal to” equality test):

   ```plaintext
   if ((a <> 0) AND ((b / a) == 1))
       print("Test successful");
   ```

   The code snippet contains a potentially serious problem. Name a language feature that can be used to avoid this problem, and explain how it avoids the problem. [2]

3. Consider the following expression in a hypothetical programming language:

   ```plaintext
   a + fun(a);
   ```

   Answer the questions that follow:
   (a) Explain how a function side effect and the operand evaluation order can interact to make the final value of the expression ambiguous. [2]
   (b) Suggest one (1) solution to the above-mentioned problem of function side effects. [1]