Engineering, Built Environment and IT
Department of Computer Science
Programming Languages
COS 333

Examination
15 November 2022

Examiner: Mr W. S. van Heerden
External Examiner: Prof MC du Plessis (NMU)
Instructions:
Read the questions carefully and answer all questions.
This section comprises of **29** questions, and a total of **40** marks.
You have **3** hours to complete this test.
This test is an *online* assessment:
The test will automatically submit when the time (2.5 hours) expires. You can also submit the test yourself if you are done ahead of time.
This test is **closed book** and is subject to the University of Pretoria integrity statement provided below.
You are not allowed to consult any sources other than the MIT/GNU Scheme Reference Manual (provided as documentation for Practical 2) and the SWI-Prolog Reference Manual (provided as documentation for Practical 3).
For practical implementation questions, your Scheme submission must be working Scheme code that will successfully interpret using the online GNU Guile interpreter located at https://rextester.com/l/scheme_online_compiler, while your Prolog submission must be working Prolog code that will be successfully interpreted by the online SWI-Prolog interpreter located at https://swish.swi-prolog.org/. You may (and should) use both these interpreters to test your submissions. Once you have finished writing each program, make sure each is saved in a text file with the appropriate name (described in the question), and submit it to the correct upload slot.
You may use a non-programmable calculator
You may not use any other programmable devices.
Integrity Statement:
The University of Pretoria commits itself to produce academic work of integrity. I affirm that I am aware of and have read the Rules and Policies of the University, more specifically the Disciplinary Procedure and the Tests and Examinations Rules, which prohibit any unethical, dishonest or improper conduct during tests, assignments, examinations and/or any other forms of assessment. I am aware that no student or any other person may assist or attempt to assist another student, or obtain help, or attempt to obtain help from another student or any other person during tests, assessments, assignments, examinations and/or any other forms of assessment.

Timed Test

This test has a time limit of 5 hours.This test will save and submit automatically when the time expires.
Warnings appear when **half the time**, **5 minutes**, **1 minute**, and **30 seconds** remain.*[The timer does not appear when previewing this test]*

QUESTION 1

1. For programming languages that are designed for systems programming applications, the primary concern is efficiency. Consider the following statements, and **select** the one that most correctly identifies the second most important concern in programming languages designed for systems programming.

- Portability.
- Generality.
- Orthogonality.
- Reliability.

1 points

QUESTION 2

1. Pseudocode languages were interpreted rather than compiled. Consider the following statements, and **select** the option that most correctly identifies why these languages were interpreted.

- Interpretation offers the advantage of cross-platform compatibility.
- Interpretation is more reliable in terms of error detection than compilation.
- Certain operations were not supported in hardware at the time, meaning execution was slow regardless of whether interpretation was used.
- Interpretation requires less computing resources than compilation does, meaning it worked better on the limited hardware of the time.

1 points

QUESTION 3

1. Consider the following statements, and **select** the option that most correctly describes Objective-C.

- Object-oriented programming in Objective-C is based on C.
- Object-oriented programming in Objective-C is based on C++.
- Object-oriented programming in Objective-C is based on Smalltalk.
- Objective-C does not support object-oriented programming.

1 points

QUESTION 4

1. Write a Scheme function named `getPositiveOddValues`, which receives one parameter. The parameter is a simple list (i.e. a list containing only atoms). You may assume that the list contains only numeric atoms. The function should yield (not print out) a list containing only the positive (non-zero) odd values contained in the parameter list.

   For example, the function application

   ```
   (getPositiveOddValues '())
   ```

   should yield an empty list because there are no elements in the parameter list. As another example, the function application

   ```
   (getPositiveOddValues '(1 2 7 -3 4))
   ```

   Should yield the list `(1 7)` because 1 and 7 are the only positive odd values in the list (2 and 4 are positive even values, and -3 is a negative value).
   Submit your program code in a file named `s99999999.scm` (where `99999999` is your student number) to the assignment submission slot labelled "Exam: Scheme Function", and select the "True" answer below once you have done so. If you do not make a submission, select the "False" answer below.

   Take careful note of the following requirements:

- Your submission must be working Scheme code that will be successfully interpreted by version 2.2.3 of the GNU Guile interpreter (this means, amongst other things, that lowercase letters should be used for all built-in function names). You may (and should) use the GNU Guile interpreter to test your submission.
- You may define additional helper functions.
- Your Scheme function may only use the following built-in functions in the way they are used in the slides and textbook (failure to do so will result in a zero mark for this question):

  - Function construction: `lambda`, `define`
  - Binding: `let`
  - Arithmetic: `+`, `-`, `*`, `/`, `abs`, `sqrt`, `remainder`, `modulo`, `min`, `max`
  - Boolean values: `#t`, `#f`
  - Equality predicates: `=`, `>`, `<`, `>=`, `<=`, `even?`, `odd?`, `zero?`, `negative?`, `eqv?`, `eq?`
  - Logical predicates: `and`, `or`, `not`
  - List predicates: `list?`, `null?`
  - Conditionals: `if`, `cond`, `else`
  - Quoting: `quote`, `'`
  - List manipulation: `list`, `car`, `cdr`, `cons`
  - Input and output: `display`, `printf`, `newline`, `read`

   ○ Yes

   ○ No

5 points

QUESTION 5

1. Write the Prolog proposition called `stripOccurrences(X, L1, L2)`, which succeeds if the list `L2` contains all the elements in the list `L1`, in order, with all occurrences of `X` removed. For example, the query:

```
?- stripOccurrences(a, [b, c, d], X).
```

should be true with the instantiation `X = [b, c, d]`. Similarly, the query

```
?- stripOccurrences(a, [a, b, a, c, a], X).
```

should be true with the instantiation `X = [b, c]`.

Hint: When testing your proposition, use variables in your queries, in a similar fashion to the example queries provided above.
Submit your program code in a file named `s99999999.pl` (where `99999999` is your student number) to the assignment submission slot labelled "Exam: Prolog Proposition", and select the "Yes" answer below once you have done so. If you do not make a submission, select the "No" answer below.

Take careful note of the following requirements:

- Your submission must be working Prolog code that will be successfully interpreted by the online SWI-Prolog interpreter located at https://swish.swi-prolog.org/. You may (and should) use this SWI-Prolog interpreter to test your submission. Once you have finished writing your program, save it in a text file with the appropriate name, and submit it to the correct upload slot.
- You may define additional helper propositions.
- Note that you may only use the simple constants, variables, list manipulation methods, arithmetic and relational expressions, and built-in predicates discussed in the textbook and slides. You may NOT use any more complex predicates provided by the Prolog system itself. In other words, you must write all your own propositions. Failure to observe this rule will result in all marks for a task being forfeited.

- Yes
- No

5 points

QUESTION 6

1. Consider the following statements about the scope and lifetime of static variables, and **select** the one that is most correct.
   - The scope of the variable is from the point of the variable declaration until the end of its scope, while the lifetime is from the start of program execution until the end of program execution.
   - The scope of the variable is from the start of program execution until the end of program execution, while the lifetime is from the point of the variable declaration until the end of its scope.
   - The scope and lifetime are both from the point of the variable declaration until the end of its scope.
   - The scope and lifetime are both from the start of program execution until the end of program execution.

1 points

QUESTION 7

1. It is possible for pointers to be implicitly dereferenced by a programming language. Consider the following possible disadvantages associated with implicitly dereferenced pointers, and **select** the one that is most correct.

- Void pointers cannot be used.
- Pointer arithmetic is disallowed.
- Pointers cannot be used for indirect addressing.
- Pointers cannot be used for dynamic memory management.

1 points

QUESTION 8

1. Consider the following statements about operator precedence and associativity in APL, and **select** the one that is most correct.

- Precedence rules in APL are more orthogonal than precedence rules in other programming languages, but associativity rules in APL are less orthogonal than associativity rules in other programming languages.
- Associativity rules in APL are more orthogonal than associativity rules in other programming languages, but precedence rules in APL are less orthogonal than precedence rules in other programming languages.
- Precedence and associativity rules in APL are both more orthogonal than precedence and associativity rules in other programming languages.
- Precedence and associativity rules in APL are both less orthogonal than precedence and associativity rules in other programming languages.

1 points

QUESTION 9

1. In Scheme, operator overloading isn't supported due to Scheme's reliance on dynamic typing. **Select** one of the following reasons that most correctly explains why dynamic typing results in a lack of operator overloading in Scheme.

- Dynamic typing means Scheme operators can receive operands of any type.
- Dynamic typing means overloaded operators would be very unreliable.
- Dynamic typing means overloaded versions of operators are automatically generated.
- Dynamic typing means operator support doesn't make sense in Scheme.

1 points

QUESTION 10

1. Ada supports no assignment coercions. Consider the following statements, and **select** one that most correctly describes a result of this lack of assignment coercions.

- Ada is more writable.
- Ada is more orthogonal.
- Ada is more reliable.
- Ada has lower cost.

1 points

QUESTION 11

1. Perl supports multiple assignments. Consider the following statements, and **select** one that most correctly describes an advantage of this support.

- Perl is more writable.
- Perl is more orthogonal.
- Perl has reduced cost.
- Perl is more reliable.

1 points

QUESTION 12

1. Consider the following program code fragment in a hypothetical programming language:

```
int [] myArr = {3, 4, 1, 6, 2}
int index = 0
int length = 5
int threshold = 10
while ((myArr[index] == threshold) or (++index < length))
{ ... }
```

Consider the following statements, and **select** the option that most correctly describes a problem with the above program code.

- If the `or` operator is short circuit evaluated, a side effect will not occur, which causes a problem.
- If the `or` operator is short circuit evaluated, a side effect occurs, which causes a problem.
- If the `or` operator is not short circuit evaluated, a side effect will not occur, which causes a problem.
- If the `or` operator is not short circuit evaluated, a side effect occurs, which causes a problem.

1 points

QUESTION 13

1. Fortran I had a three-way selection statement. In this statement, one branch would be executed if the control variable value was above zero, a second branch would be executed if the control variable value was equal to zero, and a third branch would be executed if the control variable value was below zero. Consider the following statements, and select all the ones that are properties of the three-way selection statement. Note that incorrectly selected options will be penalised.

- [ ] The three-way selection statement allows flow control to be written, which cannot be written using two-way selection and pretest logical loops.
- [ ] The three-way selection statement is very efficient.
- [ ] The three-way selection statement is an example of programmer time being considered more important than execution time.
- [ ] The three-way selection statement is too closely based on the IBM 704 hardware.

2 points

QUESTION 14

1. Counter-controlled loops are supported in C and Ada. Consider the following statements, and **select** the one that is most correct.

- Counter-controlled loops in C are less writable than counter-controlled loops in Ada.
- Counter-controlled loops in C are more costly than counter-controlled loops in Ada.
- Counter-controlled loops in C are less reliable than counter-controlled loops in Ada.
- Counter-controlled loops in C are more orthogonal than counter-controlled loops in Ada.

1 points

QUESTION 15

1. It is possible for a conditional to be part of a `break` used in a loop. Consider the following drawbacks, and **select** the one that applies to situations in which a conditional is part of a `break`.

- Writability is reduced.
- Readability is reduced.
- Orthogonality is reduced.
- Cost is reduced.

1 points

QUESTION 16

1. Consider the following program code in a hypothetical programming language that supports guarded commands:

```
var a = 11

if (a < 0) -> print("Negative")
[] (a mod 2 == 0) -> print("Even")
fi
```

Type the output of the program in the space provided. If the program produces no output, type "Nothing". If the program produces a compile-time error, type "Compilation error". If the program produces a runtime error, type "Runtime error".

1 points

QUESTION 17

1. In Python, subprogram definitions are executable. Consider the following statements about the implications of executable subprogram definitions, and **select** the one that is most correct.

- Writability is reduced.
- Readability is reduced.
- Cost is reduced.
- There is no effect on the programming language, because all languages that support subprograms have executable subprogram definitions.

1 points

QUESTION 18

1. In programming languages like C++, default parameters must appear last in the list of parameters. Consider the following solutions for allowing default parameters to appear anywhere in the list of parameters, and **select** the one that is most correct.

- Use a variable number of parameters, if the programming language supports them.
- Use positional parameters, if the programming language supports them.
- Use keyword parameters, if the programming language supports them.
- It is not possible to allow default parameters to appear anywhere in the list of parameters.

1 points

QUESTION 19

1. Procedures are a specific kind of subprogram. Consider the following statements, and **select** the one that is most correct.

- Procedures are guaranteed to never have side effects.
- Procedures are guaranteed to always have side effects.
- Procedures have a high probability of having side effects, unless they are written carefully.
- Procedures generate side effects unpredictably.

1 points

QUESTION 20

1. Consider the following program code in a hypothetical programming language that allows subprogram names to be passed as parameters:

```
A = 4
subprogram f(sub) {
    A = 9
    call sub()
}
subprogram g() {
    print A
}
subprogram main() {
    A = 2
    call f(g)
}
```

Assume that execution starts with the `main` subprogram, and that the programming language uses static scoping. **Type** the output generated by this program if ad-hoc binding is assumed.

1 points

QUESTION 21

1. Consider the following program code in a hypothetical programming language that supports coroutines:

```
subprogram main() {
    resume g()
}
coroutine f() {
    print "A "
    resume g()
    print "B "
}
coroutine g() {
    print "C "
    resume f()
    print "D "
}
```

Assume that execution starts with the `main` subprogram. **Type** the output generated by this program (note that there is a space at the end of each output).

1 points

QUESTION 22

1. Java supports a mechanism for ensuring that clients can't depend on the implementation of ADTs. Consider the following language features, and **select** the one that ensures a lack of client dependence on implementation.
   - Header files and implementation files.
   - Base classes and derived classes.
   - Classes implementing interfaces.
   - Javadoc and bytecode.

1 points

QUESTION 23

1. Classes are dynamic in Ruby. Consider the following statements about the implications of dynamic classes in Ruby, and **select** the one that is most correct.
   - Dynamic classes increase writability.
   - Dynamic classes increase readability.
   - Dynamic classes decrease cost.
   - Dynamic classes increase reliability.

1 points

QUESTION 24

1. **Briefly outline** how the compiler handles parameterised classes in Java 5.0.

3 points

QUESTION 25

1. Smalltalk supports object-oriented programming. Consider the following statements, and **select** the one that is most correct.
   - Smalltalk's support for object-oriented programming is very writable.
   - Smalltalk's support for object-oriented programming is quite efficient.
   - Smalltalk's support for object-oriented programming is highly orthogonal.
   - Smalltalk is essentially equivalent in terms of programming language evaluation criteria to other object-oriented programming languages such as C++ and Java.

1 points

QUESTION 26

1. Assume a programming language that supports multiple inheritance. There is a parent class called A. Two classes, called B and C, both inherit directly from A. Finally, a class called D inherits from both B and C. Consider the following statements, and **select** the one that causes an **implicit** problem with the described inheritance hierarchy.
   - If any class member is defined in class A.
   - If any class member is defined in class B.
   - If any class member is defined in class C.
   - If any class member with the same name is defined in classes B and C.

1 points

QUESTION 27

1. C++ provides support for private derivation. Consider the following statements, and **select** the one that describes a valid use for private derivation.

- When a derived class wishes to only add to the public interface of its base class.
- When a derived class wishes to only modify how one part of the public interface provided by its base class works.
- When a derived class wishes to only make part of its base class interface publicly visible.
- When a derived class wishes to ensure that it is a subtype of its base class.

1 points

QUESTION 28

1. C++ and Java both support object-oriented message binding. Consider the following statements, and **select** the one that is most correct.

- C++ support for message binding is more writable than Java's support.
- C++ support for message binding is more readable than Java's support.
- C++ support for message binding is less efficient than Java's support.
- C++ support for message binding is more reliable than Java's support.

1 points

QUESTION 29

1. Consider the following statements, and **select** the one that is most correct.

- C# only supports classes, and not structs.
- Classes are more writable than structs in C#.
- Structs are more writable than classes in C#.
- Classes and structs are equally writable in C#.