Comp Sci 3MI3 Midterm Test
McMaster University

Day Class 01, Dr. J. Carette

DURATION: 50 Minutes October 24, 2023

Please CLEARLY print:

NAME:

Student Number:

This examination paper includes 5 pages. It has 9 questions. You are responsible for ensuring that your copy of the examination paper is complete. Bring any discrepancy to the attention of your invigilator. All questions must be answered on the midterm itself.

Special Instructions:

1. The use of notes and textbooks is **not** permitted.
2. Calculators, computers, cell phones, and all other electronic devices are **not** to be used.
3. Read each question carefully.
4. For all code on this midterm, you may use any function from the Haskell Prelude, and any Haskell feature. You may similarly use any feature of SWI-Prolog that has been used up until now.
5. If the space provided is not sufficient, use the back page.

continued on page 2

Question 1 [2 marks]
What learning objective was served by demonstrating the esoteric languages *Piet* (i.e. the one named after the famous modern painter Piet Mondrian).

Question 2 [3 marks]
What is BNF (Backus-Naur Form) and what was it invented for? A complete answer will contrast the situation pre-BNF as well.

Question 3 [2 marks]
Declarative is a programming paradigm where what is to be computed is specified, but how to achieve that aim is not, with Prolog being a leading example of such a language. What is the 'other' paradigm, and what is a good example of such a language?

continued on page 3

Question 4 [6 marks]
Consider the following Prolog code:

```
toto(R, [], R).
toto([X|Xs], [X|Ys], Zs) :- toto(Xs, Ys, Zs).
```

Think of it as a relation where the “output” is the first argument, and the other two are inputs.
Implement it in Haskell. To help you, here is a type signature:

```
toto :: [a] -> [a] -> [a]
```

Question 5 [5 marks]
Consider the following `thing` Prolog relation, seeing the first and third arguments of the relation to be inputs, and the second argument to be the output. `append` is list append, as done in class, and `length` is the Prolog built-in relation between a list (first argument) and its length (second).

```
prefix(L, P) :- append(P , _ , L).
thing(L, N, P) :- prefix(L, P), length(P, N).
```

Given the following Haskell type signature

```
thing :: Eq a => [a] -> [a] -> Integer
```

either translate `thing` to Haskell with that signature, or find a Prolog query that shows that this is not possible (and explain your reasoning). Note that `thing` must have this exact signature and must be a *total, terminating* function.

continued on page 4

Question 6 [3 marks] Consider the following BNF:

```
E ::= N  |  E + E  |  E * E
N ::= 1 | 2| 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Is the expression `1 + 2` a member of E? What about `(1+2)*6` ? Explain your reasoning.

Question 7 [3 marks] Explain why the term $(\lambda x.xx)(\lambda x.xx)$ using either call-by-name or call-by-value reduction strategy does not lead to a terminating evaluation.

continued on page 5

Question 8 [1 marks] Consider the following Haskell declaration:

```
data Expr = Int Integer | Add Expr Expr | Mul Expr Expr
```

where `Add` is to be interpreted as addition and `Mul` as multiplication.
Give the Haskell term that would correspond to $(5 + 17) * 6$.

Question 9 [3 marks] Rewrite the data declaration from the previous question as a class in *finally tagless* style.

THE END.