Swansea University
Prifysgol Abertawe

Faculty of Science and Engineering

Computer Science

CS-205 Declarative Programming

January 2025

Duration: 2 Hours

Do not turn over your question paper until instructed to do so.

The content of this exam paper has been thoroughly checked, however if you think you have spotted an error, please raise your hand and report it to an invigilator, to log the query. The question will not be corrected in the exam venue. You must also note the query in the answer booklet and continue to answer the paper to the best of your ability as it is written. Queries will be noted by the module coordinator and the Board of Examiners when it considers results.

Exam Paper Information & Instructions

Answer all questions.

Please start a new page for each subquestion 1(a), 1(b), and so on. Do not change the order, and clearly indicate when a question is continued elsewhere.

Permitted Materials

Dictionaries – Candidates may only refer to the English and Welsh language dictionaries available

Calculators – Candidates may NOT use a calculator

Open Book – This is NOT an Open Book Exam

Other – Scrap paper

Additional Materials

None

1. Functional programming (25 marks total)

   (a) Alice and Eve have just shared a meal at their favorite place and are thinking of splitting the bill. They have just enough money in their bank account to cover the meal, and Alice has more money than Eve, so they decide to split the bill as follows: either Alice covers it all if it does not make her poorer than Eve, or they pay so that they both end up with the same balance on their respective bank accounts.

   Write a Haskell function, together with a type signature, that takes as inputs Alice's and Eve's account balances together with the price of the bill and returns what Alice should be paying. You should provide a correct type signature and briefly explain which inputs correspond to each protagonist's account balance and the bill in your solution.

   [5 marks]

   (b) A javascript expert writes the following definition in a file, which is intended to add a single element at the end of a list.

```
appendAtTheEnd :: Int -> [Int] -> [Int]
appendAtTheEnd x xs = xs ++ x
```

   When trying to load the file, `ghci` outputs an error.

   For answering the question, recall that `(++)` is list concatenation, which has the following type signature:

```
(++) :: [a] -> [a] -> [a]
```

   i. Is the triggered error a compile-time error or a runtime error?

   [1 mark]

   ii. Explain in at most three sentences why the compiler rejected this definition.

   [2 marks]

   iii. Propose a fix: give a modification of the function that the compiler accepts and has the behaviour desired by the javascript expert.

   [2 marks]

   (c) Write a program with a `main` function that asks the user for input, outputs `"Hi"`, and repeat this behaviour until the input from the user is `"Bye"`. You may find the following functions useful:

```
getLine  :: IO String
putStrLn :: String -> IO ()
return   :: a -> IO a
show     :: Show a => a -> String
```

   [4 marks]

Paper continues on the next page

(d) Say that `ys` is a *sublist* of `xs` if it is obtained by removing elements of `xs`. For instance, `[4,7,2,4]` is a sublist of

```
[1,4,5,4,7,2,5,2,6,4]
```

but `[9]`, `[4,1]` and `[2,2,2]` aren't.

Write a function `maximalOrderedSublist` which outputs a *maximal ordered sublist* of its input, i.e. an ordered list such that, if we insert a new element anywhere in it, it is either no longer ordered or no longer a sublist of the input. For instance, we could have

```
maximalOrderedSublist [1,4,4,5,4,7,2,5,2,6,4] == [1,4,4,5,7]
```

You should make your function polymorphic and use the `Ord` typeclass.

[6 marks]

(e) Consider the following type for trees (with possibly many children at each level).

```
data Tree a = Node a [Tree a]
```

Call a node a *leaf* if it is equal to `Node x []` for some `x`. Write a function `trimLeaves` that removes a single layer of leaves from its input, assuming the input tree is not just given as a single leaf. For instance, assuming that we have the definition

```
leaf :: a -> Tree a
leaf x = Node x []
```

we should have that the expression

```
trimLeaves (Node 0 [Node 1 [leaf 2,
                            Node 3 [leaf 4, leaf 5]
                           ],
                    Node 6 [leaf 7]])
```

should evaluate to `Node 0 [Node 1 [Node 3 []], Node 6 []]`. You may want to define an auxiliary function.

[5 marks]

Paper continues on the next page

2. Logic programming (25 marks total)

   (a) Indicate for the following Prolog queries whether they will fail or not. If the query fails, explain why this is the case. If the query is successful, then give the variable instantiations that Prolog will return.

      i. `?- staff(mia,20) = staff(Name,Age).`
      ii. `?- connection([Z,2],[1,Y]) = connection(X,X).`
      iii. `?- [11,12|L] = [11,12,13|K].`
      iv. `?- X+7 = 7+6.`
      v. `?- h(X,Y,g(X)) = h(a,h(Z),Z).`

[5 marks]

   (b) Consider the following Prolog code:

```
creative(tim).
happy(anna).
hasFriend(anna,paul).
hasFriend(paul,tim).
hasFriend(X,Y) :- hasFriend(X,Z),hasFriend(Z,Y).
hasWonderfulLife(X) :- happy(X).
hasWonderfulLife(X) :- hasFriend(X,Y), creative(Y).
```

      i. Draw a proof tree for `hasWonderfulLife(X)` and indicate which paths yield solutions (4 marks).
      ii. Does this proof tree have an infinite path? If no, justify your answer, if yes, mark it in the proof tree (2 marks).

[6 marks]

   (c) An arithmetical puzzle is given below:

3	x	A1	-	A2	=	7
x	■	x	■	+	■	
A3	+	A4	x	A5	=	13
x	■	-	■	+	■	
A6	-	8	+	9	=	8
=	■	=	■	=	■	
21		16		16		

Paper continues on the next page

The nine grey cells are to contain all the distinct digits from 1 to 9. Digits are given for three of the cells (3, 8 and 9) and variables A1 to A6 represent the six unknown distinct digits (from 1,2,4,5,6 and 7). The six arithmetical equations given by the three rows and three columns determine the six variables where normal operator precedence is used. So the equation in the second row is A3 +(A4*A5) =13 (and not (A3+A4)*A5=13).

Construct a simple Prolog program to solve this puzzle as follows (2 marks for each question):

i. First, assume that you are given predicate `genList(N,S,L)` which will generate a list `L` of `N` distinct elements from the list `S` (where `S` is of length greater than or equal to `N`). For instance, `genList(2,[1,2,3,4,5],[3,5])` holds.

   Using the `genList` predicate, define a generator

   ```
   generate([A1,A2,A3,A4,A5,A6])
   ```

   which will produce a possible assignment of the variables for the puzzle. (Note this is not necessarily a solution.)

ii. Write `solvePuzzle([A1,A2,A3,A4,A5,A6])` to solve the puzzle by generating a possible solution and then testing to see if all the constraints are satisfied (use =:= to test for equality of two arithmetic expressions).

iii. Give an implementation of the `genList` predicate (used in i). You may use

   ```
   member_rem(X,[X|Xs],Xs).
   member_rem(X,[Y|Ys],[Y|R]) :-
           member_rem(X,Ys,R).
   ```

[6 marks]

(d) Explain in at most 1-2 sentences the following concepts in Declarative Programming (To avoid confusion we specify in which area the concept is used, 2 marks each).

i. Recursion and Tail recursion
ii. Polymorphic Function [Functional Programming]
iii. Lazy evaluation [Haskell]
iv. Generate and Test Paradigm [Logic Programming]

[8 marks]

End of Paper