Prifysgol Abertawe
Swansea University

Faculty of Science and Engineering

Computer Science

CS-205 Declarative Programming

January 2023

Time Allowed: 2 hours

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

Exam Paper Information

None

Special Instruction(s)

None

Specific Items

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

Question 1: Functional Programming

(a) A painter asks for 1.20£ per squaremeter of wallspace painted. If, however, this yields a price below 25£ per room, he sets the price to 25£. Write a Haskell function that takes the length, width, and height of a room as inputs and computes the price to get a room painted.

[4 marks]

(b) For all of the following Haskell definitions, indicate if they typecheck, and if they do, write down their (most general) types.

(i)
```
myTriple = ('a',"abc",True)
```

(ii)
```
add11 = 1 + True
```

(iii)
```
churchThree f x = f (f (f x))
```

(iv)
```
mapApp fs x = map (\ f -> f x) fs
```

(v)
```
atLeast2 x y z = x /= y || y /= z
```

[5 marks]

(c) Let the following Haskell program be given

```
main :: IO ()
main = getLine >>= mainLoop
  where mainLoop previousInput = do
                                 {
                                   putStrLn (previousInput ++ "!");
                                   i <- getLine;
                                   if i /= previousInput then
                                     mainLoop i
                                   else
                                     return ()
                                 }
```

(i) Explain informally how this code works. Under which condition will this program terminate?

(ii) Give the output assuming that the user enters successively `Hi <newline> Program <newline> ? <newline> ? <newline>` (where each `<newline>` corresponds to hitting the return/enter key).

(iii) What is the type of the subexpression `mainLoop`?

[5 marks]

(d) Write a Haskell function

```
sumsOf2 :: Int -> [(Int, Int, Int)]
```

that takes as input a number n and outputs all triples (x, y, z) such that $1 \leq x \leq y \leq z \leq n$ and $x + y = z$. For instance, we could have `sumsOf2 4` equal to `[(1,1,2),(1,2,3),(1,3,4),(2,2,4)]` (but you are free to enumerate the triples in any order you want).

[3 marks]

(e) Recall that Haskell ships with the following algebraic datatype which is convenient to do error-handling

```
data Maybe a = Just a | Nothing
```

Write a function

```
groupByPairsIfEven :: [a] -> Maybe [(a,a)]
```

which takes as input a list of elements $[x_0, x_1, x_2, x_3, \ldots, x_{2n}, x_{2n+1}]$ and outputs the list of pairs $[(x_0, x_1), (x_2, x_3), \ldots, (x_{2n}, x_{2n+1})]$ wrapped by `Just` if the input has an even length and `Nothing` otherwise.

[4 marks]

(f) Consider the following data type for arithmetic expressions containing the constant 1, the variable x, $+$ and $\times$.

```
data Expr = One | X | Plus Expr Expr | Times Expr Expr
```

For instance, the syntactic expression $1 + (x(1 + (1 + x)))$ would be represented by the constant `Plus One (Times X (Plus One (Plus One X)))`. Note that when fully unfolded, these correspond to polynomials with positive integer coefficients. Write a haskell function

```
degree :: Expr -> Int
```

which takes as input such a representation of type `Expr` of such an expression and outputs the degree of the underlying polynomial (recall that the degree of a polynomial $p_d x^d + \ldots + p_1 x + p_0$ is d if $d \neq 0$).

[4 marks]

Question 2: Programming in Prolog and Verification in Haskell

(a) Briefly explain what we understand by *declarative programming*, and give two reasons why every computer scientist should learn about it.

[3 marks]

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

(i) `?- TOM = mia.`

(ii) `?- f(2,a) = g(X,B).`

(iii) `?- f(a,g(X,b),h(Z)) = f(X,g(Y,Z),X) .`

(iv) `?- [a,b|A] = [a,b,c|B].`

(v) `?- h(a,Y,g(X)) = h(X,Z,Z).`

[5 marks]

(c) Proof search: Explain the advantages of Prolog's proof search strategy, and describe how it works by using the following example. Draw a proof tree for `beach(X)` and indicate which paths yield solutions.

```
weekend(sat).
weekend(sun).
bankholiday(mon).
warm(sun).
sunny(sun).
warm(mon).
beach(X) :- (weekend(X), sunny(X)); bankholiday(X).
beach(X) :- warm(X), sunny(X).
```

[6 marks]

(d) Given the following program:

```
miracle([],[]).
miracle([H,_|T],L):-miracle(T,NewL),append(NewL,[H],L).
```

(i) What does the query `?- miracle([1,2,3,4,5,6],Y).` yield?

(ii) Rewrite the program `miracle` in such a way that it uses an *accumulator*.

[5 marks]

(e) Program Verification in Haskell: consider the following base functions

```
(.) :: (b -> c) -> (a -> b) -> a -> c
(.) f g x = f (g x)

filter :: (a -> Bool) -> [a] -> [a]
filter p []                   = []
filter p (x : xs) | p x       = x : filter p xs
                  | otherwise = filter p xs

map :: (a -> b) -> [a] -> [b]
map f []       = []
map f (x : xs) = f x : map f xs
```

Show by induction on lists that for finite lists `xs` and arbitrary `p` and `f` (which we assume to terminate without errors on all inputs), we always have

```
filter p (map f xs) = map f (filter (p . f) xs)
```

Hint: Please show the base case in detail. For the step case it is sufficient to give the main idea of the proof.

[6 marks]

End of Paper