ECM2418 Computer Languages and Representations Continuous Assessment Functional and Logic Programming

Dr David Wakeling

Handed out	Handed in
Thursday 27th October 2022 (T1:05)	Thursday 15th December 2022 (T1:12)

This Continuous Assessment is worth 40% of the module mark.

Functional programming solutions must run on the implementation at

```
https://replit.com/languages/haskell
```

Logic programming solutions must run on the implementation at

```
https://swish.swi-prolog
```

All students are reminded of the University regulations on academic honesty and plagiarism.

Question 1: Party Time

Every week, *The Sunday Times* newspaper publishes a Teaser. Teaser 3026, of Sunday 20th September 2020, was as follows.

> A four-digit number with different positive digits and with the numbers represented by its last two digits a multiple of the number represented by its first two digits, is called a PAR.
>
> A pair of PARs is a PARTY if no digit is repeated and each PAR is a multiple of the missing positive digit.

I wrote down a PAR and challenged Sam to use it to make a PARTY. He was successful.

I then challenged Beth to use my PAR and the digits in Sam's PAR to make a different PARTY. She too was successful.

What was my PAR?

Note that in this Teaser, zero is not considered to be a positive digit.

Question 1.1

Show a Haskell function "`digits`" that takes a positive integer and returns a list of its digits, so that

```
digits 9124
   ====> [9,1,2,4]
```

```
digits :: Int -> [Int]
digits n
  |  n < 10      =  [n]
  |  otherwise   =  digits d ++ [m]
     where
     (d, m) = n `divMod` 10
```

(5 marks)

Question 1.2

Show a Haskell function "`isPar`" that returns true if an integer (which can be assumed to have four digits) is a PAR, and false otherwise, so that

```
isPar 2678
   ====> True
```

```
isPar :: Int -> Bool
isPar n
   =  length us == 4 && notElem 0 us && n1 `mod` n2 == 0
      where
      ds                =  digits n
      us                =  nub ds
      [d3,d2,d1,d0]     =  ds
      n1                =  d1 * 10 + d0
      n2                =  d3 * 10 + d2
```

Show also a Haskell definition “pars” that describes a list of all 44 PARs.

```
pars :: [Int]
pars
  =  [ p  | p <- [1234 .. 9876], isPar p ]
```

(10 marks)

Question 1.3

Show a Haskell function “isParty” that returns true if a pair of integers (which can be assumed to be PARs) is a PARTY, and false otherwise, so that

```
    isParty (2754, 1836)
        ====> True
```

```
isParty :: (Int, Int) -> Bool
isParty (p, q)
  =  is == [] && p `mod` w == 0 && q `mod` w == 0
     where
     ds1   =  digits p
     ds2   =  digits q
     is    =  intersect ds1 ds2
     us    =  union ds1 ds2
     [w]   =  [1..9] \\ us
```

Show also a Haskell definition “partys” that describes a list of all 14 PARTYs.

```
partys :: [(Int, Int)]
partys
  =  [ (p,q) | p <- pars, q <- pars, isParty (p,q) ]
```

(10 marks)

By inspecting these PARTYs you should be able to obtain the solution to Teaser 3026. However, this is *not* part of the assessment.

The answer to Teaser 3026 is 1785.

Question 2: Bilateral Symmetry

Teaser 3132, of Sunday 1st October 2022, was as follows.

My son, at a loose end after A-levels, asked me for a mental challenge, As we'd been discussing palindromes, I suggested he try to find an arrangement of the digits 1 to 9 with the multiplication symbol "x" to give a palindrome as the answer, and he came up with 29678x1453 = 43122134. I said "Now try to find the smallest such palindromic product starting in 4, where the last digit of the smallest number is still 3. He found that smallest product, and, interestingly, if he added the two smaller numbers instead of multiplying them, then added 100, he also go a palindrome.

What was the smallest product?

Question 2.1

Show a Haskell function "`number`" that takes a list of digits and returns the positive integer formed from those digits, so that

```
number [9,1,2,4]
   ====> 9124
```

```
number :: [Int] -> Int
number xs
  =  foldr f 0 (reverse xs)
     where
     f x y = x + 10 * y
```

(5 marks)

Question 2.2

Show a Haskell function "`splits`" that returns all of the non-trivial splits of a list as a list of pairs, so that

```
splits [1,2,3,4]
   ====> [([1],[2,3,4]),([1,2],[3,4]),([1,2,3],[4])]
```

```
splits :: [a] -> [([a],[a])]
splits s
  =  [ splitAt n s | n <- [1..length s-1] ]
```

Show also a Haskell "`possibles`" definition that uses the "`permutations`" function from the "`Data.List`" library to list all 2,903,040 pairs that make up the Teaser solution space.

```
possibles :: [([Int],[Int])]
possibles
   =  concatMap splits (permutations [1..9])
```

(10 marks)

Question 2.3

Show a Haskell function "`isAcceptable`" that takes a possible Teaser solution, and returns true if it is an actual solution to the Teaser; that is, one where the palindromic product starts with a '4', and the last digit of the smallest number is '3', so that

```
   isAcceptable ([7,1,6,3], [5,9,2,4,8])
      ====> True
```

```
isAcceptable :: ([Int], [Int]) -> Bool
isAcceptable (as, bs)
  =   ends3 ms && starts4 ds && palindrome ds
      where
      a  = number as
      b  = number bs
      ms = digits (min a b)
      ds = digits (a * b)

ends3 :: [Int] -> Bool
ends3 xs
   =  last xs == 3

starts4 :: [Int] -> Bool
starts4 xs
  =  head xs == 4

palindrome :: [Int] -> Bool
palindrome xs
   =  xs == reverse xs
```

Finally, show a Haskell "`acceptables`" definition that uses the "`isAcceptable`" function to produce a list of all six acceptable solutions to the Teaser.

```
acceptables :: [([Int],[Int])]
acceptables
   =  [ p | p <- possibles, isAcceptable p ]
```

(10 marks)

By inspecting these acceptable solutions, you should be able to obtain the answer to Teaser 3132. However, this is *not* part of the assessment.

The answer to Teaser 3132 is $23 \times 1769548 = 40699604$.

Question 3: Suko

Every day, *The Times* newspaper publishes a *Suko puzzle*. Readers are challenged to place the digits 1 to 9 in the cells of a grid so that numbers in four white circles are the sums of the digits in the cells surrounding them, and the numbers in the three coloured circles are the sums of the digits in the cells of the same colour. Suko 3566 of Thursday 12th August 2022, and its solution are shown in Figure 1.

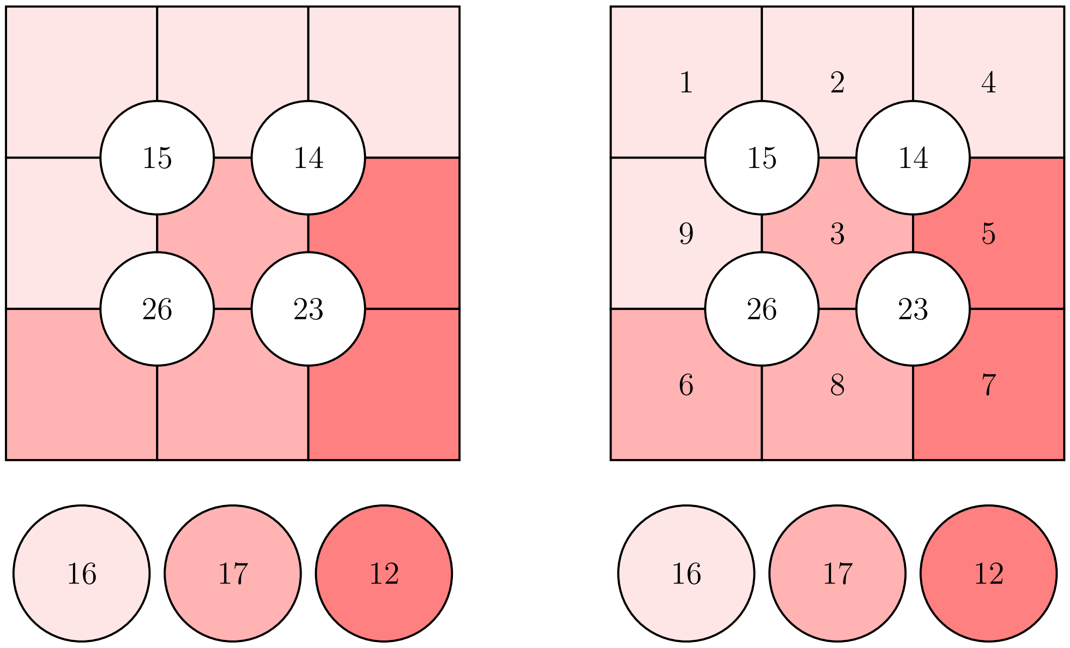


Figure 1: Suko 3566.

Question 3.1

Show a Prolog predicate "`indices`" that takes a list of indices (numbering from zero) and a list of elements, and returns a list of the elements with the indices, so that

```
indices( [1,3,4], [a,b,c,d,e,f], ES )
ES = [b,d,e]
```

```
indices( [], _, [] ).
indices( [I|IS], XS, [E|ES] )
  :- index( I, XS, E ), indices( IS, XS, ES ).

index( 0, [H|_], H ).
index( I, [_|T], E )
  :- I1 is I-1, index( I1, T, E ).
```

(5 marks)

Question 3.2

Show a Prolog predicate "`possible`" that may be stepped with $\langle$ *NEXT* $\rangle$ to return successive arrangements of the digits '1' . . . '9' that might be possible solutions of a Suko puzzle, so that

```
possible( GRID ).
GRID = [1, 2, 3, 4, 5, 6, 7, 8, 9]  ⟨ NEXT ⟩
GRID = [2, 1, 3, 4, 5, 6, 7, 8, 9]  ⟨ NEXT ⟩
GRID = [2, 3, 1, 4, 5, 6, 7, 8, 9]  ⟨ NEXT ⟩
GRID = [2, 3, 4, 1, 5, 6, 7, 8, 9]  ⟨ NEXT ⟩
GRID = [2, 3, 4, 5, 1, 6, 7, 8, 9]  ⟨ NEXT ⟩
...
```

```
possible( GRID )
  :- permutation( [1,2,3,4,5,6,7,8,9], GRID ).

permutation( [], []).
permutation( [H|T], S)
  :- permutation( T, P ),
     append( X, Y, P ),
     append( X, [H|Y], S ).
```

(10 marks)

Question 3.3

Show a Prolog predicate "`acceptable`" that checks if a solution to a Suko puzzle is acceptable, so that for the puzzle of Figure 1

```
acceptable( 15, 14, 26, 23
          , [0,1,2,3], 16
          , [4,6,7], 17
          , [5,8], 12
          , [1, 2, 4, 9, 3, 5, 6, 8, 7] ).
true
```

Here, the first four arguments are the numbers from the four white circles of the puzzle, the next six arguments are a list of cells of each color (numbering sequentially from top-left to bottom-right, from zero) followed by their sums from the three coloured circles of the puzzle, and the final argument is the solution grid itself.

```
acceptable( T0, T1, T2, T3, US, U, VS, V, WS, W, GRID)
  :- total( [0,1,3,4], GRID, T0 ),
     total( [1,2,4,5], GRID, T1 ),
     total( [3,4,6,7], GRID, T2 ),
     total( [4,5,7,8], GRID, T3 ),
     total( US, GRID, U ),
     total( VS, GRID, V ),
     total( WS, GRID, W ).

total( IS, GRID, V )
  :- indices( IS, GRID, VS ), sum( VS, V ).

sum( [], 0 ).
sum( [H|T], V )
  :- sum( T, W ), V is H + W.
```

Finally, show a Prolog predicate "`suko`" that uses the "`possible`" and "`acceptable`" predicates to solve Suko problems, so that

```
suko( 15, 14, 26, 23
    , [0,1,2,3], 16
    , [4,6,7], 17
    , [5,8], 12
    , GRID ).
GRID = [1, 2, 4, 9, 3, 5, 6, 8, 7]
```

```
suko( T0, T1, T2, T3, US, U, VS, V, WS, W, GRID )
  :- possible( GRID ),
     acceptable( T0, T1, T2, T3, US, U, VS, V, WS, W, GRID ).
```

(10 marks)

Question 4: The Bearings' Trait

Teaser 3125, of Sunday 14th September 2022, was as follows.

> At Teaser Tor trig. point I found a geocaching box. The three-figure compass bearings (bearing 000 = north, 090 = east, etc.) from there to the church spires at Ayton, Beeton and Seaton were needed to decode the clue to the next location.
>
> Each spire lay in a different compass quadrant (eg. 000 to 090 [sic] is the North-East quadrant). Curiously, each of the numerals 1 to 9 occurred in these bearings and none of the bearings were prime values.
>
> Given the above, if you chose one village at random to be told only its church spire's bearing, it might be that you could not calculate the other two bearings with certainty, but it would be more likely that you could.
>
> Give the three bearings in ascending order.

Question 4.1

Show a Prolog predicate "`prime`" that takes and integer and returns true if it is a prime number, so that

```
prime( 17 ).
true
```

Note that a simple algorithm based on successive division is all that is required here.

```
prime( N )
  :- \+ factorisable( 2, N ).

factorisable( F, N )
  :- F < N,
     0 is N mod F.
factorisable( F, N )
  :- F < N,
     F1 is F + 1,
     factorisable( F1, N ).
```

(5 marks)

Question 4.2

Show a Prolog predicate "`possible`" that may be stepped with ⟨ *NEXT* ⟩ to return successive bearings that might be possibe solutions of the Bearings' Trait, so that

```
possible( X, Y, Z ).
X = Θ, Y = Ω, Z = Δ   ⟨ NEXT ⟩
X = Θ, Y = Ω, Z = Δ+1  ⟨ NEXT ⟩
X = Θ, Y = Ω, Z = Δ+2  ⟨ NEXT ⟩
X = Θ, Y = Ω, Z = Δ+3  ⟨ NEXT ⟩
...
```

Note that it is for you to choose good initial bearings Θ, Ω and Δ from which to commence the search.

```
possible( X, Y, Z )
  :- between( 145, 180, X ),
     between( 245, 270, Y ),
     between( 345, 360, Z ).
```

(10 marks)

Question 4.3

Show a Prolog predicate "`acceptable`" that checks if a solution to the Bearings' Trait is acceptable, so that

```
acceptable( 169, 247, 358 ).
true
```

```
acceptable( X, Y, Z )
  :-  \+ prime( X ),
      \+ prime( Y ),
      \+ prime( Z ),
      digits( X, [A,B,C] ),
      digits( Y, [D,E,F] ),
      digits( Z, [G,H,I] ),
      difference( [1,2,3,4,5,6,7,8,9], [A,B,C,D,E,F,G,H,I], []).

digits( N, [N] )
  :- N < 10.
digits( N, W )
  :- N >= 10,
     divMod( N, 10, D, M ),
     digits( D, R ),
     append( R, [M], W ).

divMod( A, B, D, M )
  :- D is A div B,
     M is A mod B.

difference( [], _, [] ).
difference( [X|XS], YS, [X|RS] )
   :- \+ member( X, YS ),
      difference( XS, YS, RS ).
difference( [X|XS], YS, RS )
   :- member( X, YS ),
      difference( XS, YS, RS ).
```

Finally, show a Prolog predicate “trait” that uses the “`possible`” and “`acceptable`” predicates to solve the Bearings’ Trait.

```
trait( X, Y, Z )
  :-  possible( X, Y, Z ),
      acceptable( X, Y, Z ).
```

(10 marks)

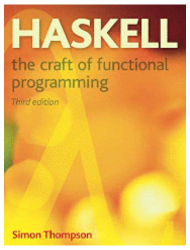
By inspecting the successive triples, you should be able to obtain the solution to Teaser 3125. However, this is not part of the assessment.

The answer to Teaser 3125 is 159, 267, 348.

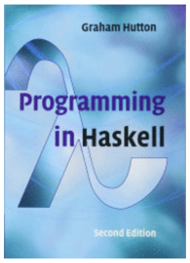
Assessment Materials and Criteria

You should submit a single “.zip” file to the EBART system. Other compression formats, such as “.rar”, “.7z”, “.gz” and “.bz2” are unacceptable, and will receive a mark of zero. The “.zip” file should contain four completed template files “`Party.hs`” (containing the answer to Question 1), “`Bilateral.hs`” (containing the answer to Question 2), “`Suko.pl`” (containing the answer to Question 3) and “`Trait.pl`” (containimng the answer to Question 4).

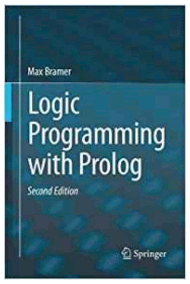
Readings



S. Thompson, *The Craft of Functional Programming (Third Edition)*, Addison Wesley, 2011, ISBN 978-0201882957.



R. Bird, *Programming in Haskell (Second Edition)*, Cambridge University Press, 2016, ISBN 978-1316626229.



M. Bramer, *Logic Programming with Prolog (Second Edition)*, Springer, 2013, ISBN 978-1447154860.