COURSEWORK SPECIFICATION
ECM2418 — Computer Languages and Representations
Module Leader: David Wakeling
Academic Year: 2024/25

Title: **Functional and Logic Programming Assignment**

Suubmission deadline: **Tuesday 19th November, 2024, 12:00**

This assessment contributes **40%** of the total module mark and assesses the following **Intended Learning Outcomes**:

- Demonstrate an understanding of the principles of functional programming and read and write simple programs in a functional programming language;
- Demonstrate an understanding of the principles of logic programming and read and write simple programs in a logic programming language;
- critically analyse a problem;
- systematically break down a problem into its components.

This is *an individual* assessment and you are reminded of the University's Regulations on Collaboration and Plagiarism. You must avoid plagiarism, collusion or any other academic misconduct. Further details about Academic Honesty and Plagiarism can be found at `https://ele.exeter.ac.uk/course/view.php?id=1957`.

AI-supported

This module is *AI-supported.* That is, it is one

> *where ethical and responsible use of GenAI tools in the development of an assessment is supported. This may include using GenAI tools to summarise literature, improve the structure of your work or quality of English language. All use of GenAI tools should be acknowledged in a statement submitted with their assessment and referenced appropriately. Students are asked to keep a record of the tools, prompts and outputs used so they are able to produce these if necessary at a viva and demonstrate how they have built on this content to ensure the work is original.*

Question 1: Gill's primes

Every week, *The Sunday Times* newspaper publishes a Teaser. Teaser 3154, *Gill's Primes*, by Bill Kinally, of Sunday 5th March 2023, was as follows.

Jack told Gill "I have found three equally-spaced prime numbers, 29, 41, and 53. The difference between the first and second is the same as the difference between the second and third, and there are no repeated digits in the six digits of my primes." Gill told Jack she had also found three equally-spaced primes, each having three digits with no repeated digits in the nine digits of her primes. She said, "If I told you that the three-digit sum of each of my primes is an odd number then you should be able to find them."

In ascending order, what are Gill's three primes?

The answer given by the newspaper was **157**, **283**, **409**.

The generate-and-select design pattern and some library functions can be used to describe a good solution to this problem, obtained by filtering values from a generator using a selector.

```
import Data.List

main :: IO ()
main
  = print (filter selector1 generator1)
```

This program should print `[([4,0,9],[2,8,3],[1,5,7]),([1,5,7],[2,8,3],[4,0,9])]`. The correct answer may then be read off directly.

Question 1.1

As a warm-up, show a Haskell function `number` that given a list of integers, seen as a list of digits, returns the corresponding single integer.

This function will be assessed by the number of tests that it passes, as counted by the function `x_warmup1` below. The expected answer is 5.

```
x_warmup1 :: Int
x_warmup1
 = n5
   where
   n1 = x_unit (number        [] ==    0)  0
   n2 = x_unit (number       [5] ==    5) n1
   n3 = x_unit (number      [4,7] ==  47) n2
   n4 = x_unit (number    [6,2,0] == 620) n3
   n5 = x_unit (number [0,2,7,4] == 274) n4

x_unit x n
 = if x then n+1 else n
```

(5 marks)

Question 1.2

Show a Haskell function `generator1` that returns a list of tuples of lists of integers (*S1*, *S2*, *S3*) that may provide solutions to the Teaser. That is, for which *S1*, *S2* and *S3* are lists of three integers, seen as lists of three digits, with no repeats in the nine digits taken together.

This function will be assessed by the number of tests that it passes, as counted by the function `x_generator1` below. The expected answer is 10.

```
x_generator1 :: Int
x_generator1
 = n10
   where
   n1  = x_unit (elem    ([2,4,1],[3,6,5],[0,7,8]) generator1)  0
   n2  = x_unit (elem    ([4,1,6],[3,0,2],[5,7,8]) generator1) n1
   n3  = x_unit (elem    ([1,0,5],[3,6,4],[2,7,8]) generator1) n2
   n4  = x_unit (elem    ([1,6,2],[7,3,5],[0,4,8]) generator1) n3
   n5  = x_unit (elem    ([8,6,4],[3,5,2],[0,7,1]) generator1) n4
   n6  = x_unit (notElem ([0,2,4],[3,5,2],[9,7,1]) generator1) n5
   n7  = x_unit (notElem ([9,8,7],[0,1,2],[3,4,8]) generator1) n6
   n8  = x_unit (notElem ([0,1,2],[0,3,4],[4,7,6]) generator1) n7
   n9  = x_unit (notElem ([9,8,7],[6,5,4],[9,2,1]) generator1) n8
   n10 = x_unit (notElem ([0,8,7],[0,1,2],[0,4,6]) generator1) n9
```

(10 marks)

Question 1.3

Show a Haskell function `selector1` that returns true for tuples of lists of digits (*S1*, *S2*, *S3*) that provide solutions to the Teaser. That is, for which *S1*, *S2* and *S3* are lists of three integers, seen as lists of three digits, where the sum of the three digits is an odd number, the three digits form prime numbers, and the three prime numbers are equally-spaced.

This function will be assessed by the number of tests that it passes, as counted by the function `x_selector1` below. The expected answer is 10.

```
x_selector1 :: Int
x_selector1
 = n10
   where
   n1  = x_unit (      selector1 ([1,5,7],[2,8,3],[4,0,9]))   0
   n2  = x_unit (      selector1 ([4,0,9],[2,8,3],[1,5,7]))  n1
   n3  = x_unit (not (selector1 ([1,0,9],[2,8,3],[4,5,7]))) n2
   n4  = x_unit (not (selector1 ([5,4,7],[2,8,3],[0,1,9]))) n3
   n5  = x_unit (not (selector1 ([8,6,4],[3,5,2],[0,7,1]))) n4
   n6  = x_unit (not (selector1 ([1,5,7],[0,2,9],[8,6,3]))) n5
   n7  = x_unit (not (selector1 ([0,8,9],[2,6,3],[5,7,1]))) n6
   n8  = x_unit (not (selector1 ([7,5,1],[6,8,3],[0,2,9]))) n7
   n9  = x_unit (not (selector1 ([1,9,7],[5,2,0],[8,4,3]))) n8
   n10 = x_unit (not (selector1 ([0,8,9],[4,1,2],[7,3,5]))) n9
```

(10 marks)

Question 2: Common names

Teaser 3137, *Common Names*, by Victor Bryant, of Sunday 6th November 2022, was as follows.

Eight friends met at a party; their ages in whole numbers of years were all different. They were Alan, Cary, James, Lucy, Nick, Ricky, Steve and Victor, with Lucy being the youngest. For each of them, the square of their age was a three-figure number consisting of three different digits. Furthermore, for any two of them, the squares of their ages had at least one digit in common precisely when their names had a least one letter in common.
In alphabetical order of their names, what are the eight ages?

The answer given by the newspaper was **19**, **31**, **29**, **16**, **25**, **23**, **28** and **27**. Note, however, that to obtain this answer, one has to assume that names are not case-sensitive: "alan" not "Alan", "cary", not "Cary", "james" not "James" and so on.

The generate-and-select design pattern and some library functions can be used to describe a good solution to this problem, obtained by filtering values from a generator using a selector.

```
import Data.List

main :: IO ()
main
  = print (filter selector2 generator2)
```

This program should print `[(19,31,29,16,25,23,28,27)]`, the one correct answer.

Question 2.1

As a warm-up, show a Haskell function `digits` that given a single integer, returns the corresponding list of integers, seen as a list of its digits,

This function will be assessed by the number of tests that it passes, as counted by the function `x_warmup2` below. The expected answer is 5.

```
x_warmup2 :: Int
x_warmup2
 = n5
   where
   n1 = x_unit (digits    0 ==       [0])  0
   n2 = x_unit (digits    5 ==       [5]) n1
   n3 = x_unit (digits   47 ==     [4,7]) n2
   n4 = x_unit (digits  620 ==   [6,2,0]) n3
   n5 = x_unit (digits 2745 == [2,7,4,5]) n4

x_unit x n
 = if x then n+1 else n
```

(5 marks)

Question 2.2

Show a Haskell function `generator2` that returns a list of tuples (*A1*, *A2*, *A3*, *A4*, *A5*, *A6*, *A7*, *A8*) that may be solutions to the Teaser. That is, for which *A1*, . . . *A8* are all different integers, the squares of *A1*, . . . *A8* are three-digit integers consisting of three different digits, and *A4* is the smallest of the integers *A1*, . . . *A8*.

This function will be assessed by the number of tests that it passes, as counted by the function `x_generator2` below. The expected answer is 10.

```
x_generator2 :: Int
x_generator2
 = n10
   where
   n1  = x_unit (elem    (14,17,27,13,24,29,23,19) generator2)  0
   n2  = x_unit (elem    (16,24,25,13,19,14,29,23) generator2) n1
   n3  = x_unit (elem    (17,27,28,13,24,23,25,14) generator2) n2
   n4  = x_unit (elem    (24,14,17,13,23,25,16,27) generator2) n3
   n5  = x_unit (elem    (29,28,14,13,23,18,19,16) generator2) n4
   n6  = x_unit (notElem (14,24,13,13,27,25,17,29) generator2) n5
   n7  = x_unit (notElem (13,31,25,13,17,14,23,27) generator2) n6
   n8  = x_unit (notElem (25,17,19,13,25,24,28,29) generator2) n7
   n9  = x_unit (notElem (16,19,14,13,18,18,19,28) generator2) n8
   n10 = x_unit (notElem (13,14,18,13,25,29,16,25) generator2) n9
```

(10 marks)

Question 2.3

Show a Haskell function `selector2` that returns true for tuples (*A1*, *A2*, *A3*, *A4*, *A5*, *A6*, *A7*, *A8*) that provide solutions to the Teaser. That is, when Alan has age *A1*, Cary has age *A2*, James has age *A3*, Lucy has age *A4*, Nick has age *A5*, Ricky has age *A6*, Steve has age *A7* and Victor has age *A8*, and the squares of their ages have at least one digit in common precisely when their names have a least one letter in common.

This function will be assessed by the number of tests that it passes, as counted by the function `x_selector2` below. The expected answer is 10.

```
x_selector2 :: Int
x_selector2
 = n10
   where
   n1  = x_unit (     selector2 (19,31,29,16,25,23,28,27))   0
   n2  = x_unit (not (selector2 (14,16,25,13,31,23,24,27))) n1
   n3  = x_unit (not (selector2 (14,24,27,13,23,16,17,31))) n2
   n4  = x_unit (not (selector2 (16,17,23,13,25,14,24,27))) n3
   n5  = x_unit (not (selector2 (16,27,18,13,25,14,19,31))) n4
   n6  = x_unit (not (selector2 (17,14,23,13,16,24,25,19))) n5
   n7  = x_unit (not (selector2 (17,16,14,13,25,19,24,31))) n6
   n8  = x_unit (not (selector2 (19,14,25,13,16,27,31,24))) n7
   n9  = x_unit (not (selector2 (19,25,14,13,16,31,24,23))) n8
   n10 = x_unit (not (selector2 (23,31,14,13,24,27,17,25))) n9
```

(10 marks)

Question 3: Bilateral symmetry

Teaser 3132, of Sunday 1st October 2022, *Bilateral Symmetry*, by Susan Bricket, 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?

The answer given by the newspaper was **40699604**.

The generate-and-select design pattern can be used to describe a good solution to this problem, obtained by selecting values produced by a generator.

```
main
  :- generator3(X), selector3(X), write(X).
```

Initially, this program should print `[[1,7,6,9,5,4,8],[2,3]]`. After pressing *NEXT*, it should print `[[2,3],[1,7,6,9,5,4,8]]`. (Or perhaps these are printed the other way around.) Either way, after pressing *NEXT* again, it should print nothing more. The correct answer may then be calculated as $23 \times 1769548 = 40699604$.

Question 3.1

As a warm-up, show a Prolog predicate `number` that given a list of integers, seen as digits, yields the corresponding single integer.

This predicate will be assessed by the number of tests that it passes, as counted by the predicate `x_warmup3` below. The expected answer is 5.

```
x_warmup3(N5) :-
  x_unit(number(       [],     0),  0, N1),
  x_unit(number(      [9],     9), N1, N2),
  x_unit(number(    [4,7],    47), N2, N3),
  x_unit(number(  [2,5,4],   254), N3, N4),
  x_unit(number([9,8,7,6], 9876), N4, N5).

x_unit(F, M, N)
  :- call(F) -> N is M + 1; N is M.
```

(5 marks)

Question 3.2

Show a Prolog predicate **generator3** that yields in succession (with *NEXT*) all pairs of lists (*AS*,*BS*) that may be solutions to the Teaser. That is, for which *AS* and *BS* are lists of integers, seen as digits between 1 to 9, where all of the digits appear in one list or the other.

This predicate will be assessed by the number of tests that it passes, as counted by the predicate **x_generator3** below. The expected answer is 10.

```
x_generator3(N10) :-
  x_unit(   generator3([[1],[2,3,4,5,6,7,8,9]] ),  0,  N1),
  x_unit(   generator3([[1,2,3,4,5,7],[9,8,6]] ), N1,  N2),
  x_unit(   generator3([[1,3,2,5],[4,8,9,7,6]] ), N2,  N3),
  x_unit(   generator3([[1,2,3],[6,4,9,8,5,7]] ), N3,  N4),
  x_unit(   generator3([[4,8,3,9,7,6],[1,2,5]] ), N4,  N5),
  x_unit(\+ generator3([[9,2,5,4,8,3,1,7,6],[]]), N5,  N6),
  x_unit(\+ generator3([[8,4,9,3,7],[1,2,5,8]] ), N6,  N7),
  x_unit(\+ generator3([[7,8,6,4,9,3],[2,5]]   ), N7,  N8),
  x_unit(\+ generator3([[2],[2,5,9,8,3,4,7]]   ), N8,  N9),
  x_unit(\+ generator3([[],[1,2,6,3,9,5,4,7,8]]), N9, N10).
```

(10 marks)

Question 3.3

Show a Prolog predicate **selector3** that is true for pairs of lists (*AS*,*BS*), that are solutions to the Teaser. That is, for which *AS* and *BS* are lists of integers, seen as lists of digits between 1 to 9, and the product of the corresponding single integers *A* and *B* is a palindrome of digits that starts with 4, the last digit of the smallest of *A* and *B* is 3, and adding *A* and *B* instead of multiplying them, then adding 100 also gives a palindrome of digits.

This predicate will be assessed by the number of tests that it passes, as counted by the predicate **x_selector3** below. The expected answer is 10.

```
x_selector3(N10) :-
  x_unit(   selector3( [[1,7,6,9,5,4,8],[2,3]] ),  0,  N1),
  x_unit(   selector3( [[2,3],[1,7,6,9,5,4,8]] ), N1,  N2),
  x_unit(\+ selector3( [[1,7],[9,6,4,8,3,5,2]] ), N2,  N3),
  x_unit(\+ selector3( [[1,7],[4,2,8,3,5,2,6]] ), N3,  N4),
  x_unit(\+ selector3( [[1,7,6,9,5,2,8],[4,3]] ), N4,  N5),
  x_unit(\+ selector3( [[],[6,9,5,2,8,4,3,1,7]]), N5,  N6),
  x_unit(\+ selector3( [[5,3,1,4,9],[8,2,6,7]] ), N6,  N7),
  x_unit(\+ selector3( [[9,1,5,3,4],[7,6,8,2]] ), N7,  N8),
  x_unit(\+ selector3( [[6,2,8,9],[5,4,3,1,7]] ), N8,  N9),
  x_unit(\+ selector3( [[9,7,3,8,2,1,4,6,5],[]]), N9, N10).
```

(10 marks)

Question 4: In his prime

Teaser 3231, of Sunday 25th August 2024, *In His Prime*, by Victor Bryant, was as follows.

Once, on my grandson's birthday, I asked him the following five questions:

How many days are there in this month?

How many Mondays are there in this month?

How many "prime" days (ie, 2nd, 3rd, 5th, . . .) are there in this month?

How many of those prime days are Saturdays?

How many letters are there when you spell the month?

The total of the five answers was a prime number.

Then I asked him the same questions the next day. No answer was the same as for the day before, but again the total of the five answers was prime.

When I first asked the questions what was the month and day of the week?

The answer given by the newspaper was **February, Monday or Friday (apologies for the multiple answers)**.

The generate-and-select design pattern can be used to describe a good solution to this problem, obtained by selecting values produced by a generator.

```
main
  :- generator4(X), selector4(X), write(X).
```

Initially, this program should print `[[2,29,1],[3,31,2]]`. After pressing *NEXT*, it should print `[[2,29,5],[3,31,6]]`. Or perhaps these are printed the other way around. Either way, after pressing *NEXT* again, it should print nothing more. The two correct answers may then be read off directly — in a 29-day month, the start day and the finish day are the same.

Question 4.1

As a warm-up, show a Prolog predicate `mondays` that given the length of a month D in days (between 28 and 31) and a start day S (between 1 and 7), yields a list of days in the month that are Mondays.

This predicate will be assessed by the number of tests that it passes, as counted by the predicate `x_warmup4` below. The expected answer is 5.

```
x_warmup4(N5)
  :- x_unit(mondays(31,4,[5,12,19,26]),      0, N1),
     x_unit(mondays(30,1,[1,8,15,22,29]),  N1, N2),
     x_unit(mondays(31,6,[3,10,17,24,31]), N2, N3),
     x_unit(mondays(28,7,[2,9,16,23]),     N3, N4),
     x_unit(mondays(30,2,[7,14,21,28]),    N4, N5).

x_unit( F, M, N )
  :- call(F) -> N is M + 1; N is M.
```

(5 marks)

Question 4.2

Show a Prolog predicate `generator4` that yields in succession (with *NEXT*) all pairs of lists ([*M1*,*D1*,*S1*], [*M2*,*D2*,*S2*]) that may be solutions to the Teaser. That is, for which *M1* and *M2* are months (between 1 and 12), *D1* and *D2* are the lengths of those months in days (between 28 and 31) and *S1* and *S2* are the start days of those months (between 1 and 7). The month *M2* is the successor of *M1*.

This predicate will be assessed by the number of tests that it passes, as counted by the predicate `x_generator4` below. The expected answer is 10.

```
x_generator4( N10 )
  :- x_unit(   generator4([[ 1,31,1],[ 2,28,4]]),  0,  N1),
     x_unit(   generator4([[ 1,31,6],[ 2,29,2]]), N1,  N2),
     x_unit(   generator4([[ 2,29,5],[ 3,31,6]]), N2,  N3),
     x_unit(   generator4([[ 8,31,5],[ 9,30,1]]), N3,  N4),
     x_unit(   generator4([[12,31,4],[ 1,31,7]]), N4,  N5),
     x_unit(\+ generator4([[11,30,2],[12,31,3]]), N5,  N6),
     x_unit(\+ generator4([[ 6,20,3],[11,30,3]]), N6,  N7),
     x_unit(\+ generator4([[12,31,4],[11,30,2]]), N7,  N8),
     x_unit(\+ generator4([[ 9,30,1],[ 5,29,2]]), N8,  N9),
     x_unit(\+ generator4([[ 7,31,3],[ 8,31,7]]), N9, N10).
```

(10 marks)

Question 4.3

Show a Prolog predicate `selector4` that is true for pairs of lists ([*M1*,*D1*,*S1*], [*M2*,*D2*,*S2*]) that are solutions to the Teaser. That is, for which *T1* is *D1*, *U1* is *D2*, *T2* is the number of letters in the spelling of *M1*, *U2* is the number of letters in the spelling of *M2*, *T3* is the number of prime days in *M1*, *U3* is the number of prime days in *M2*, *T4* is the number of Mondays in *D1* days with a start day of *S1*, *U4* is the number of Mondays in *D2* days with a start day of *S2*, *T5* is the number of prime Saturdays in *D1* days with a start day of *S1* and *U5* is the number of prime Saturdays in *D2* days with a start day of *S2*. None of (*T1*, *U1*), (*T2*, *U2*), (*T3*, *U3*), (*T4*, *U4*) and (*T5*, *U5*) are the same. The sum of *T1*, *T2*, *T3*, *T4* and *T5* must be a prime number, and the sum of *U1*, *U2*, *U3*, *U4* and *U5* must also be a prime number.

This predicate will be assessed by the number of tests that it passes, as counted by the predicate `x_selector4` below. The expected answer is 10.

```
x_selector4( N10 )
  :- x_unit(   selector4([[ 2,29,1],[ 3,31,2]]),  0,  N1),
     x_unit(   selector4([[ 2,29,5],[ 3,31,6]]), N1,  N2),
     x_unit(\+ selector4([[ 1,31,5],[ 2,28,1]]), N2,  N3),
     x_unit(\+ selector4([[ 1,31,7],[ 2,29,3]]), N3,  N4),
     x_unit(\+ selector4([[ 2,29,6],[ 3,31,7]]), N4,  N5),
     x_unit(\+ selector4([[ 2,29,4],[ 3,31,5]]), N5,  N6),
     x_unit(\+ selector4([[ 6,30,2],[ 7,31,4]]), N6,  N7),
     x_unit(\+ selector4([[ 8,31,3],[ 9,30,6]]), N7,  N8),
     x_unit(\+ selector4([[10,31,3],[11,30,6]]), N8,  N9),
     x_unit(\+ selector4([[10,31,7],[11,30,3]]), N9, N10).
```

(10 marks)

Student declaration

You must prepare a declaration saying which of these applies.

- ☐ I have used GenAI tools for developing ideas.
- ☐ I have used GenAI tools to assist with research or gathering information.
- ☐ I have used GenAI tools to help me understand key theories and concepts.
- ☐ I have used GenAI tools to identiofy trends and themes as part of my data analysis.
- ☐ I have used GenAI tools to suggest a plan or structure for my assessment.
- ☐ I have used GenAI tools to give me feedback on a draft.
- ☐ I have used GenAI tools to generate image[s], figures or diagrams.
- ☐ I have used GenAI tools to proofread and correct grammar or spelling errors.
- ☐ I have used GenAI tools to generate citations or references.
- ☐ Other [please specify].
- ☐ I have not used any GenAI tools in preparing this assessment.

Submission

You should submit a single ".zip" file to the ELE system in the usual way. Other compression formats, such as ".rar", ".7z", ".gz" and ".bz2" are unacceptable, and will receive a mark of zero. The ".zip" file should contain five text files "`Gill.hs`" (containing the answer to Question 1), "`Common.hs`" (containing the answer to Question 2), "`Bilateral.pl`" (containing the answer to Question 3), "`Prime.pl`" (containing the answer to Question 4) and "`Declaration.txt`" (containing a student declaration).

If there is any question as to whether your functional programs compute the correct result, these questions will be answered on the implementation at

`https://www.tutorialspoint.com/compile_haskell_online.php`

If there is any question as to whether your logic programs compute the correct result, these questions will be answered on the implementation at

`https://swish.swi-prolog`