Task one

Introduction

This first task covers the learning that you have undertaken during Semester One. This paper contains a few simple programming tasks. You must solve each of the problems.

Your answers to all of the problems must be written in Clojure in a functional style. You should submit a single Clojure file called `core.clj` that contains your solutions. Each solution must be preceded by a comment that explains the intent behind, and implementation of, the code. You should include tests for all of your solutions.

Tasks

These tasks are each worth 25 marks: 15 for the code, 5 for the explanation and 5 for comprehensive testing. You can find the grading scheme on the module's Blackboard site.

1. Squaring lists

Write a program that is able to take arbitrary length lists of numbers and return a lazy sequence containing the square of each number in the list. Your program should handle all numeric data types that are available in Clojure. Make your code robust so that it does not fail when presented with non-numeric data but, rather, behaves sensibly.

2. Counting coins

There are four types of common coins in US currency: quarters (25 cents), dimes (10), nickels (5) and pennies (1). There are 6 ways to make change for 15 cents:

- A dime and a nickel
- A dime and 5 pennies
- 3 nickels
- 2 nickels and 5 pennies
- A nickel and 10 pennies
- 15 pennies

How many ways are there to make change for a dollar using these common coins where 1 dollar = 100 cents?

Write a recursive change counter.

Less common are dollar coins (100 cents); very rare are half dollars (50 cents). With the addition of these two coins, how many ways are there to make change for $1000? (note: the answer is larger than 2^{32}).

3. Kindergardeners

The kindergarten class is learning about growing plants. The teachers thought it would be a good idea to give them actual seeds, plant them in actual dirt, and grow actual plants. They've chosen to grow grass, clover, radishes, and violets. To this end, they've put little styrofoam cups along the window sills, and planted one type of plant in each cup, choosing randomly from the available types of seeds.

```
[window][window][window]
..........................  # each dot represents a styrofoam cup
..........................
```

There are 12 children in the class: Alice, Bob, Charlie, David, Eve, Fred, Ginny, Harriet, Ileana, Joseph, Kincaid, and Larry. Each child gets 4 cups, two on each row. The children are assigned to cups in alphabetical order.

The following diagram represents Alice's plants:

```
[window][window][window]
VR........................
RG........................
```

So in the row nearest the window, she has a violet and a radish; in the row behind that, she has a radish and some grass.

Your program will be given the plants from left-to-right starting with the row nearest the windows. From this, it should be able to determine which plants belong to which students.

For example, if the garden looks thus:

```
[window][window][window]
VRCGVVRVCGGCCGVRGCVCGCGV
VRCCCGCRRGVCGCRVVCVGCGCV
```

Then if asked for Alice's plants, your program should return `[violets, radishes, violets, radishes]`

Asking for Bob's plants would give `[clover, grass, clover, clover]`

4. Meteor falls

Using data from this source https://data.nasa.gov/resource/y77d-th95.json write a solution that can answer questions such as:

1. Which year saw the most individual meteor falls?
2. Which year saw the heaviest collective meteor fall?

Your program must download the data itself from the URL given in the question not load it from a local file. You should add up to three further questions of your own that you can use to demonstrate different approaches to manipulating the data.

Learning outcomes

This task assesses the following module learning outcomes:

- Apply the functional style of programming to a range of simple problems
- Implement a complex application in a functional language

Submission

Upload your file to the assignment handler on Blackboard by **3pm on 13th January, 2020**.

Marking Grid

Each of your four solutions will be marked using the matrix shown below and the grading scheme that is outlined at https://tinyurl.com/y3smjwk9. This is aligned with the SHU category grading model.

	Marks	We are looking for solutions that include
Quality of your code • Does it produce a correct answer • Use of functions • Use of data structures • Use of list comprehensions • Appropriate recursion • Pattern matching • Use of Clojure specs	15	• The code produces answers but these may be error-prone • Code structures such as lists and functions are generally used in appropriate ways • There may be places in which other structures would be better • Functional techniques including recursion and pattern matching are used sparingly • There are few, if any, places in which different code structures would be a better fit • The code is recognisably “functional” throughout

• Immutable data throughout • Naming of functions and vars		• Clojure language features such as spec are used
Your explanation • Accuracy • Relevance • Completeness	5	• Your work includes some explanations. They are generally accurate but may include occasional errors. • There are some errors in the work. • Important aspects of the explanation are detailed.
Your use of testing • Coverage • Suitability of the tests • Use of relevant frameworks • Quality of reporting	5	• Thorough testing across the functions • The tests are clear and useful • Tests results are presented in a neatly formatted way