1 Language Features

1. Typing
   - Categories:
     - Static vs. Dynamic
     - Strong vs. Weak
   - Are Clojure and/or Python strongly typed?

2. Language Philosophy
   - Is there any program that can be written in Python that can’t be written in Clojure (and vice versa)? If not, then why should both languages exist?
   - What does it mean for a function to be “pure”?
   - Is object oriented programming compatible with “pure” functional programming?
     - Pick a position an argue for it. Reasonable arguments could be made for either answer.
   - Describe the similarities and differences between objects and closures.
   - Bonus question: Syntax aside, have we seen all of the general purpose languages that can exist?
     - I think that a reasonable argument can be made for the claim that we have. What do you think?

2 Clojure

1. Name two features of Clojure that distinguish it from other LISP.

2. Functions vs. Macros
   - List two differences between functions and macros.
   - When should you use a function instead of a macro?
   - When should you use a macro instead of a function?

3. What does the following function return (and why)?

```clojure
(defn conjugator []
  (let [v []]
    (for [value (range 10)]
      (conj v value)))))
```

3 Python

1. Name two features of Python that distinguish it from other imperative languages.

2. What does the function `appender` return (and why)?

```python
def append(container, value):
    container.append(value)
    return container

def appender():
    v = []
    return [append(v, value) for value in range(10)]
```
4 Language Translation

1. Translate the following Clojure code into Python

(a) counter

```clojure
(defn frequency [s]
  (let [keys (into #{} s)]
    (into {}
      (for [key keys]
        [key (count (filter (partial = key) s))])))
)
```

(b) compositions

```clojure
(defn compose [& functions]
  (fn [& args]
    (loop [funs (reverse functions) input args]
      (if (empty? funs)
        (first input)
        (recur (rest funs) [(apply first funs) input])))))
)
```

2. Translate the following Python code into Clojure

(a) balanced brackets

```python
def is_balanced(input):
    stack = []
    pairs = {')':'(',
             ']':'[',
             '}':'{"}
    for char in input:
        if char in "([{":
            stack.append(char)
        if char in ")]}"
            if not stack: return False
            if stack[-1] != pairs[char]: return False
            stack = stack[:-1]
    return not stack
```

(b) simple closure

```python
def counter():
    val = 0
    def new_counter():
        nonlocal val
        value = val
        val += 1
        return value
    return new_counter
```