CSCI 3055U, Assignment 2, Programming in Clojure

Suppose that we would like to implement a functionality called render. We would like to support multiple types of data. The function **render : Data -> String**

* Rendering an integer such as 10 returns the string “INTEGER[10]”
* Rendering a float such as 3.1415 returns the string “FLOAT[3.1415]”
* Rendering a string such as “hello world” returns the string “hello world”
* Rendering a vector such as [1,2.1,3] returns the **concatenation of all the elements** in the vector rendered recursively: “INTEGER[1] FLOAT[2.1] INTEGER 3”.
* Anything else will be rendered as the string “BLANK”

(1) Implement *render* function using a simple (defn render [data] …) In the body of the function, use introspection (type data) to decide on what to do.

(2) Implement *render* function using the Clojure multimedthod facility.

(3) Implement *render* using the protocol facility.

For (1-3), make sure you include some testing code to illustrate that the render function is working properly.

(4) Complete the following table:

|  |  |  |
| --- | --- | --- |
|  | Advantage | Disadvantage |
| (defn render [data] …) |  |  |
| multimethod |  |  |
| protocol |  |  |

(5) What are some ways of handling inheritance?

(6) What does the following code do?

(defn g  
 ([f & colls]  
 (apply concat (apply map f colls))))

(7) Implement *quicksort* function in Clojure. Your function must have the following signature

(defn quicksort [comparator & coll] …)

The quicksort should exist in a separate namespace, called csci3055u.a2

(8) Write a separate testing program to test the implementation of csci3055u.a2/quicksort.

Submission:

* render-1.clj
* render-2.clj
* render-3.clj
* report.pdf  
  contains answers to 4-5.
* quicksort.clj
* quicksort-test.clj

Marking scheme

(1-3)

render-1, render-2, render-3 are correctly implemented 30

render-2, render-2, render-3 are correctly tested 15

(4) 10

(5) 10

(6) 10

(7) 15

(8) 10