**CS2030S RECITATION 07**

Q1

* FP emphasis: No side effects, Immutable Data, Declarative, Lazy evaluation
* This code seems to be imperative rather than declarative
* Cannot comment on immutability / side effects because it is not clear what the code does beyond naming conventions
* While-loop ends only with 1. Still\_alive = false or 2. exception gets thrown
* This implies the still\_alive gets edited, which is a side effect. ie >= 1 of the 4 functions will change the value of still\_alive for
* Pure: accepts inputs, returns values, not edit your original value.
* To change to FP, must be return test\_covid(still\_alive), then remove the while loop that edits the original still\_alive. (so if you created an additional probabl\_alive and edited that, then function returns test\_covid(probably\_alive), it adheres to FP)

What about BFS then? How to construct BFS while following FP principles?

Q2

See code

Sandbox.make(q)

.map(QA::displayQuestion)

.map(questionInside -> questionInside.getAnswer())

.consume(results::add); or .consume(x -> x.add());

Q3 Functor: have at least a constructor and map method

1. Ans is too verbose. Use StringBuffer. Google it…

String stringReverse(String sentence) {

if (sentence.isEmpty()) {

return “”;

} else {

char c = sentence.charAt(0)

String newSentence = “”;

for(int i = 1; i<sentence.length(); i++) {

newSentence += sentence.charAt(i);

}

return stringReverse(newSentence) + c

}

String stringReverse(String s) {

return new String(new StringBuffer(s).reverse());

}

1. Use Map and Filter instead of doing loops etc

String s = "The rain in Spain falls mainly in the plain";

listMap(word -> stringReverse(word),

listFilter(word -> word.indexOf(‘i’) != -1,

intoWords(s)));

listMap(word -> stringReverse(word),

listFilter(word -> word.contains(“I=i”),

intoWords(s)));

KIV: To return a variable number of things -> return a List.