**Assignment 5**

**Harinee Anandh, John Stephan Gutam, Neha Reddy Bogireddy**

1. **(4 pts.) Suppose a coffee shop sells Americano coffee, cappuccino, espresso, latte, cold coffee and frappuccino.** 
   1. **If your coverage criterion says to test how each drink tastes, how many test requirements do you have?**

For testing how each drink tastes, the number of test requirements is 6 because there are 6 drinks.

* 1. **Before tasting each drink, you need to ensure your palate is clean, for example by rinsing with water. What kind of testing problem does this solve? (Hint: This is a one word answer.)**

By rinsing the palate with water, we are making sure that each drink’s taste is not confused with the previously tasted drink, thus ensuring proper testing.

* 1. **If the coffee shop is out of espresso, and you just cannot force yourself to drink cold coffee, what coverage level can you achieve?**

Since the coffee shop is out of espresso and cold coffee cannot be force fed, the coverage level now includes only 4 out of the 6 drinks, which is around 66.67%.

* 1. **You decide that you should also try each drink with and without sugar, and with and without cream. How many test requirements do you now have?**

The total number of test requirements for trying out the 6 drinks with 4 combinations i.e. with sugar, without sugar, with cream and without cream is 6\*4 = 24.

1. **(3 pts.) Ammann & Offutt, Exercises chapter 6.1, Number 3, parts (a), (b), and (c). Answer the questions for the method search().**

**Answer the following questions for the method search() below: public static int search(List list, Object element)**

**//Effects: if list or element is null throw NullPointerException**

**//else if element is in the list, return an index**

**//of element in the list:else return -1**

**//for example, search ([3,3,1],3) = either 0 or 1**

**//search ([1,7,5], 2) = -1 Base your answer on the following characteristic partitioning: Characteristic: Location of element in list**

**Block 1: element is first entry in list**

**Block 2: element is last entry in list**

**Block 3: element is in some position other than first or last**

**(a) “Location of element in list” fails the disjointness property. Give**

**an example that illustrates this.**

“Location of element in list” fails the disjointness property. An example of this statement is a list with a single element i.e., [3]. If the search is for element 3, then it satisfies all three blocks of the characteristic: “Location of element in list”, thus failing the disjointness property.

**(b) “Location of element in list” fails the completeness property. Give an example that illustrates this.**

For a list [2,3,5,6,7] and a search for element 1, the statement is satisfied because it does not match with any of the three blocks in the characteristic since element 1 is absent from the list.

**(c) Supply one or more new partitions that capture the intent of “Location of element in list” but do not suffer from completeness or disjointness problems.**

One other partition that does not suffer from completeness or disjointness problems is “Block 4: Element does not exist in the list”. This does not conflict with the other three blocks and satisfies both disjointness and completeness properties.

1. **(6 pts.) Ammann & Offutt, Exercises chapter 6.2, Number 7, parts (a), to (f). Design an input domain model for the logic coverage web application on the book’s website.**

**Design an input domain model for the logic coverage web application on the book’s website. That is, model the logic coverage web application using the input domain modelling technique.**

1. ***List all of the input variables, including the state variables.***
2. Expression (P)
3. Operators (Negation, And, Or, Implication, Exclusive Or, Equivalence)
4. Variables (A, B, C, D, E, ...)
5. Truth Table (GACC, CACC, RACC, RICC)
6. Test Options (Sequential, Random)

***(b) Define characteristics of the input variables. Make sure you cover all input variables.***

1. Expression:
2. Presence of Negation (!)
3. Usage of Parentheses Nesting (nested, non-nested)

Example: P = (A & B) | (C ^ D)

2. Operators:

1. Order of Operator Precedence (random, strict)

Example (Random): P = A & B | C ^ D

Example (Strict): P = (A & B) | (C ^ D)

1. Combination of Operators (homogeneous, heterogeneous)

Example (Homogeneous (same types of operators)): P = A & B & C

Example (Heterogeneous (different types of operators)): P = A | B ^ C

3. Variables:

1. Variable Naming Convention:

Example (Single letter names): P = A & B

Example (Descriptive names): P = EmployeeAge > 21

1. Variable Reuse:

Example (Repeated variables): P = A & A

Example (Unique variables): P = A & B

4. Truth Table:

Size of Truth Table (small, medium, large)

Example (Small): P = A & B

Example (Medium): P = A & B | C

Example (Large): P = A & B | C ^ D | E

5. Test Options:

Ordering of Test Generation (sequential, random)

Example (Sequential): P = A & B | C

Example (Random): P = C ^ D | A & B

***(c) Partition the characteristics into blocks.***

1. Expression Block:

Sub-blocks: Negation, Parentheses Nesting

1. Operators Block:

Sub-blocks: Order of Precedence, Combination

1. Variables Block:

Sub-blocks: Naming Convention, Reuse

1. Truth Table Block:

Sub-block: Truth Table Size

1. Test Options Block:

Sub-block: Test Generation Order

***(d) Designate one block in each partition as the “Base” block.***

Expression Block: Negation

Operators Block: Order of Precedence

Variables Block: Naming Convention

Truth Table Block: Small Size

Test Options Block: Sequential Order

***(e) Define values for each block.***

Negation: P = !A

Order of Precedence: P = A & B | C ^ D

Naming Convention: P = A & B

Small Size: P = A & B

Sequential Order: P = A & B | C

***(f) Define a test set that satisfies Base Choice Coverage (BCC). Write your tests with the values from the previous step. Be sure to include the test oracles.***

Based on our understanding, Base Choice Coverage (BCC) is a testing criterion that aims to ensure that each "base" or representative choice within a partition of the input domain is covered by at least one test case. So, we selected values from each base block.

1. P = !A

Base block: Expression

Oracle: To check the correct handling of negation in the expression.

1. P = A & B | C ^ D

Base block: Operators

Oracle: To confirm the correct application of operators with a defined order.

1. P = A & B

Base block: Variables

Oracle: To make sure the variable naming convention is handled properly.

1. P = A & B

Base block: Truth table

Oracle: To validate the generation of a small-sized truth table.

1. P = A & B | C

Base block: Test options

Oracle: To check for accurate sequential order in test generation.

1. **(7 pts.) Ammann & Offutt, Exercises chapter 7.2.2, Number 5, parts (a) to (g). Answer questions for the graph.**

**N = {1, 2, 3, 4, 5, 6, 7}**

**N0 = {1}**

**Nf = {7}**

**E = {(1,2), (1,7), (2, 3), (2, 4), (3, 2), (4, 5), (4, 6), (5, 6), (6, 1)}**

**(candidate) test paths:**

**p1 =[1,2,4,5,6,1,7]**

**p2 =[1,2,3,2,4,6,1,7]**

**p3 =[1,2,3,2,4,5,6,1,7]**

1. ***Draw the graph:***
   1. 
2. ***List the test requirements for Edge-Pair Coverage. (Hint: You should get 12 requirements of length 2.)***
   1. 1-2-4
   2. 2-4-5
   3. 4-5-6
   4. 5-6-1
   5. 6-1-7
   6. 1-2-3
   7. 2-3-2
   8. 3-2-4
   9. 4-5-6
   10. 2-4-6
   11. 4-6-1
   12. 6-1-2
3. ***Does the given set of test paths satisfy Edge-Pair Coverage? If not, state what is missing.***
   1. p1 covers all the edges and satisfies the edge-pair coverage.
   2. p2 misses (2-3-2) and (3-2-4) and does not satisfy edge-pair coverage.
   3. p3 covers all the edges and satisfies the edge-pair coverage.
4. ***Consider the simple path [3,2,4,5,6] and the test path [1,2,3,2,4,6,1,2,4,5,6,1,7]. Does the test path tour the simple path directly? With a sidetrip? If so, write down the sidetrip.*** 
   1. The test path [1,2,3,2,4,6,1,2,4,5,6,1,7], tours the simple path [3,2,4,5,6] with a sidetrip [1,2,4,6,1].
5. ***List the test requirements for Node coverage, Edge Coverage, and Prime Path Coverage on the graph.***
   1. Test requirements for Node coverage: {1,2,3,4,5,6,7}
   2. Test requirements for Edge coverage: {(1,2), (2,3), (2,4), (3,2), (4,5), (4,6), (5,6), (6,1), (1,7)}
   3. Prime Path Coverage: {p1, p2, p3}
6. ***List test paths from the given set that achieve Node Coverage but not Edge Coverage on the graph.***
   1. p1 = [1,2,4,5,6,1,7]
7. ***List test paths from the given set that achieve Edge Coverage but not Prime Path Coverage on the graph***
   1. p2 = [1,2,3,2,4,6,1,7]