CS 4501/650
ISP Coverage Criteria

Names:

Purpose: Practice input space testing, apply ISP coverage criteria
Instruction: Work with your neighbors in groups. Consider a web app, Logic Coverage. Design IDM for the app and then write tests that satsifty Base Choice Coverage. Note: Try to keep your partitioning simple and choose a small number of partitions and blocks.

1. List all of the input variables, including the state variables

```
* Predicate P
* 10 buttons
```

2. Define characteristics of the input variables. Make sure you cover all input variables

```
Characteristics for input predicate
* characteristic 1: Number of clauses
* characteristic 2: Number of operators
* characteristic 3: Predicate valid
* characteristic 4: Predicate empty
* characteristic 5: Are any clauses repeated
... (there could be many more)
Note: we probably do not want all these characteristics

Characteristics for input buttons
* characteristic 1: Button is pressed
Note: the buttons are all mutally exclusive, and at least one button
   must be pressed to do any testing. Thus, we cannot press two buttons at one,
   and we always must press at least one button.
   Therefore, ten True/False characteristics would be unnecessary.
```

3. Partition the characteristics into blocks

```
Let's rename the characteristics

For input predicate
* characteristic A: Number of clauses: [0, 1, 2, 3-4, > 4]
* characteristic B: Number of operators: [0, 1, >1]
* characteristic C: Predicate valid: [T, F]
* characteristic D: Predicate empty: [T, F]
* characteristic E: Are any clauses repeated: [T, F]

For input button
* characteristic E: Button pressed: [Truth Table, GACC, CACC, RACC, GICC, RICC,
                              NewExpression, Graph, DataFlow, Minimal-MUMCUT]
```

4. Do all the partitions you design satisfy completeness and disjointness properties? If not, revise the partition(s)

```
Yes
```

5. Define values for each block

```
* characteristic A: Number of clauses: [0, 1, 2, 4, 5]
* characteristic B: Number of operators: [0, 1, 2]
* characteristic C: Predicate valid: [T, F]
* characteristic D: Predicate empty: [T, F]
* characteristic E: Are any clauses repeated: [T, F]
* characteristic E: Button pressed: [Truth Table, GACC, CACC, RACC, GICC, RICC,
                              NewExpression, Graph, DataFlow, Minimal-MUMCUT]
```

6. Write a set of test requirements that satisfies Base Choice Coverage Then, identify any infeasible test requirements you might have.

```
* characteristic A: Number of clauses: [0, 1, 2, 4, 5], (base: 2)
* characteristic B: Number of operators: [0, 1, 2], (base: 1)
* characteristic C: Predicate valid: [T, F], (base: T)
* characteristic D: Predicate empty: [T, F], (base: F)
* characteristic E: Are any clauses repeated: [T, F], (base: F)
* characteristic E: Button pressed: [Truth Table, GACC, CACC, RACC, GICC, RICC,
                              NewExpression, Graph, DataFlow, Minimal-MUMCUT], (base: T

tr1 (base test): [2, 1, T, F, F, Truth Table]
tr2  : [2, 1, T, F, F, GACC]
tr3  : [2, 1, T, F, F, CACC]
tr4  : [2, 1, T, F, F, RACC]
tr5  : [2, 1, T, F, F, GICC]
tr6  : [2, 1, T, F, F, RICC]
tr7  : [2, 1, T, F, F, NewExpression]
tr8  : [2, 1, T, F, F, Graph]
tr9  : [2, 1, T, F, F, DataFlow]
tr10 : [2, 1, T, F, F, Minimal-MUMCUT]
tr11 : [2, 1, T, F, T, Truth Table]
tr12 : [2, 1, T, T, F, Truth Table]
tr13 : [2, 1, F, F, F, Truth Table]
tr14 : [2, 0, T, F, F, Truth Table]
tr15 : [2, 2, T, F, F, Truth Table]
tr16 : [0, 1, T, F, F, Truth Table]
tr17 : [1, 1, T, F, F, Truth Table]
tr18 : [4, 1, T, F, F, Truth Table]
tr19 : [5, 1, T, F, F, Truth Table]
```

7. Write a test set (that satisfies Base Choice Coveage). Write your tests with the values from step 5. Reminder: each test case consists of test inputs and expected output.

```
test 1 (base test): a & b, Truth Table
test 2  : a & b, GACC
test 3  : a & b, CACC
test 4  : a & b, RACC
test 5  : a & b, GICC
test 6  : a & b, RICC
test 7  : a & b, NewExpression : expected P = ""
test 8  : a & b, Graph : expected another app with title "Graph Coverage"
test 9  : a & b, DataFlow : expected another app with title "Data Flow Graph Coverage"
test 10 : a & b, Minimal-MUMCUT : expected another app with title "Minimal MUMCUT Cove
test 11 : a & a, Truth Table
test 12 : "", Truth Table, expected error message (the message will reflect the implem
test 13 : a b &, Truth Table, expected error message (the message will reflect the imp
test 14 : a b, Truth Table, expected error message (the message will reflect the imple
test 15 : a & !b, Truth Table
test 16 : |, Truth Table, expected error message (the message will reflect the impleme
test 17 : !a, Truth Table
test 18 : (a & b) & (c | d), Truth Table]      // infeasible, fixed by changing B to 3
test 19 : a & b | c | d & e, Truth Table]      // infeasible, fixed by changing B to 4
```

Note: expected outcomes may be varied, depending on the testers.
Some testers may compare an entire HTML page while
some testers may focus on portions of the page.

Assuming we don't know what GACC, CACC, RACC, GICC, and RICC do,
we may choose to verify if the app produces associated results
by simply verifying if "result for xxxx" text appears as part of the output.
(while this direction can check if the app produces the results,
we assumes the results satify the criterion selection)

Assuming we know what GACC, CACC, RACC, GICC, and RICC do,
we may choose to verify in details if the app produces the results
that satisfy the chosen criterion.