ECE 653 - ASSIGNMENT 1

ARSHAD MOMIN

Student ID- 20986176

Email- a5momin@uwaterloo.ca

Question 1 :

(a) If possible, identify a test case that does not execute the fault.

Answer :

- a = [[5, 7], [8, 21]]
- b= [[8], [4]]
- matmul(a,b)

Expected Output: " ValueError: Incompatible dimensions"

Actual Output: "ValueError: Incompatible dimensions"

The fault has been detected to be the incompatible dimension of the test case and an error has been caused.

(b) If possible, identify a test case that executes the fault, but does not cause an error.

Answer:

values of 'p' and 'p1' are equal. The number of columns of matrix "a" is equal to the number of columns of matrix "b"

- a = [[5, 7], [8, 21]]
- b = [[5, 7], [8, 21]]
- matmul(a,b)

Expected Output: [[81, 182], [208, 497]]

Actual Output: [[81, 182], [208, 497]]

The fault has been executed and no error caused. The expected output is equal to the actual output.

(c) If possible, identify a test case that results in an error, but not in a failure.

Answer:

- a = [[8], [4]]
- b = [[8], [4]]
- matmul(a, b)

matrix 'a' is of size 1x1 and 'b' is of size 1x1. The size of the matrices become out of range and it therefore shows Index error.

Expected Output: [[96], [48]]

Actual Output: "IndexError: list index out of range"

(d) Identify the first error state. Describe the complete state that includes the process counter pc.

Answer:

Test case a = [[5, 7], [8, 21]] & b = [[8], [4]]

Expected Output : [[68], [148]]

Actual Output: "ValueError: Incompatible dimensions"

First Error state:

a = [[5, 7], [8, 21]]

b = [[8], [4]]

i=0

j=1

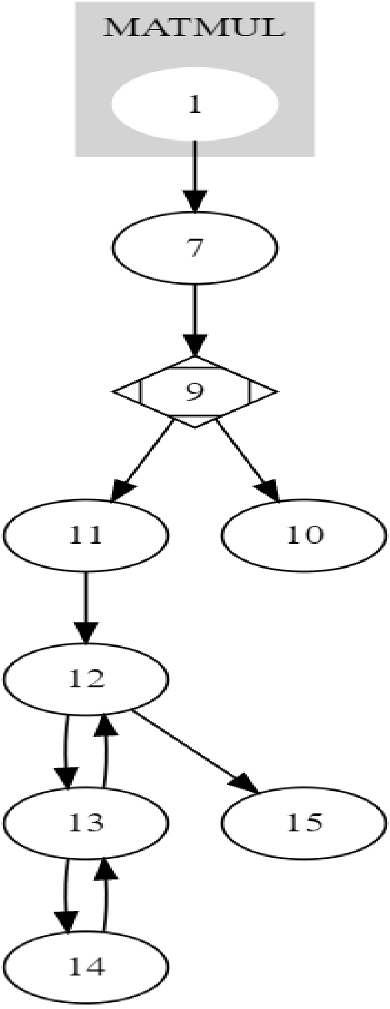
k=1

Process Counter = "c[i][j] = sum(a[i][k] * b[k][j] for k in range(p))",
"ValueError: Incompatible dimensions"

Error = "ValueError: Incompatible dimensions"

(e) CFG- matmul function: (CFG has been generated using Graphviz as recommended in the Assignment 1 pdf)

Considering Node 9 LHS = False and RHS = True



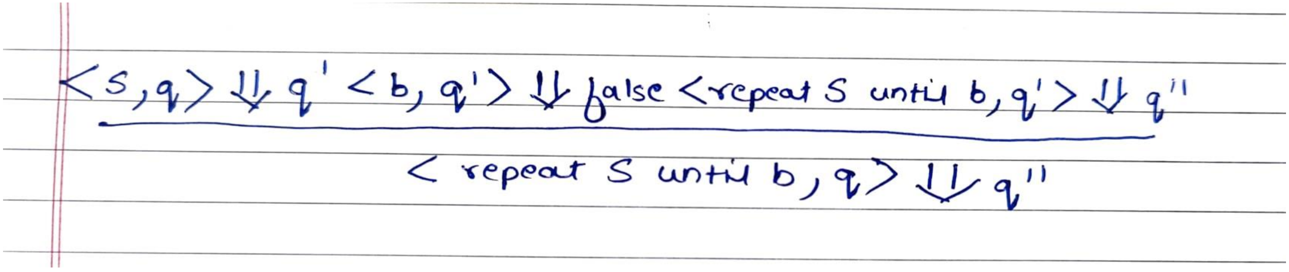
Question 2:

a) **Answer:** The below source code with class RepeatUntilStmt for indicating AST (Abstract syntax tree) node:

```
class RepeatUntilStmt(Stmt):
"""Repeat-until statement"""
def __init__(self):
self.cond = cond
self.stmt = stmt
```

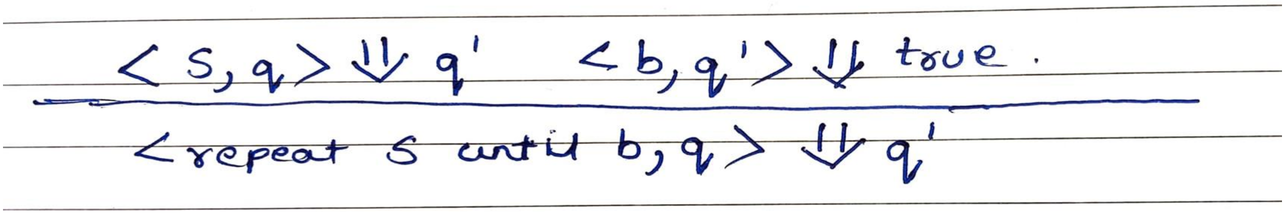
b) **Answer:** Semantics of the repeat-until loop with below iteration:

1. S is executed.
2. b is evaluated.
3. if the current value of b is false, the loop continues to the next iteration;
4. if the current value of b is true, the loop terminates (and statements following the loop are executed).



$$\frac{\langle S, q\rangle \Downarrow q' \quad \langle b, q'\rangle \Downarrow false \quad \langle repeat\ S\ until\ b, q'\rangle \Downarrow q''}{\langle repeat\ S\ until\ b, q\rangle \Downarrow q''}$$

Continues below



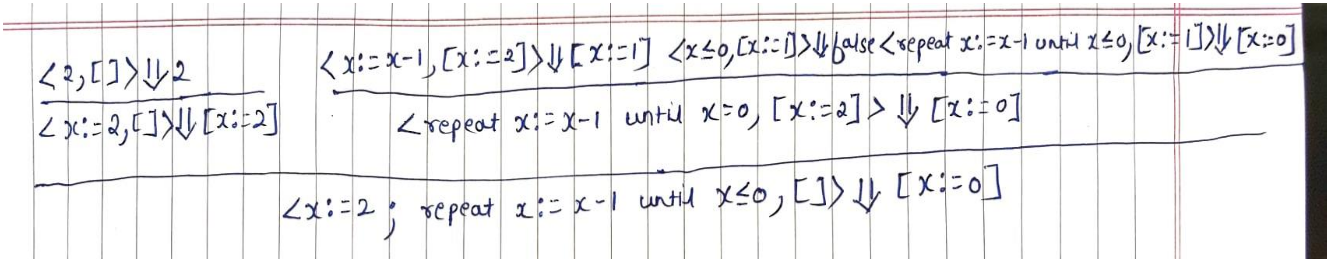
$$\frac{\langle S, q\rangle \Downarrow q' \quad \langle b, q'\rangle \Downarrow true.}{\langle repeat\ S\ until\ b, q\rangle \Downarrow q'}$$

Textual Representation:

$$\frac{< \mathrm{S}, \mathrm{q} >\rightarrow \mathrm{q}' < \mathrm{b}, \mathrm{q}' >\rightarrow \mathrm{false} < \mathrm{repeat\ S\ until\ b}, \mathrm{q}' >\rightarrow \mathrm{q}'' < \mathrm{S}, \mathrm{q} >\rightarrow \mathrm{q}' < \mathrm{b}, \mathrm{q}' >\rightarrow \mathrm{true}}{< \mathrm{repeat\ S\ until\ b}, \mathrm{q} >\rightarrow \mathrm{q}'}$$

c) **Answer:**

To prove the judgement is valid as below:



$$\frac{\dfrac{\langle 2, [\,]\rangle \Downarrow 2}{\langle x:=2, [\,]\rangle \Downarrow [x:=2]} \qquad \dfrac{\langle x:=x-1, [x:=2]\rangle \Downarrow [x:=1] \quad \langle x\leq 0, [x:=1]\rangle \Downarrow false \quad \langle repeat\ x:=x-1\ until\ x\leq 0, [x:=1]\rangle \Downarrow [x:=0]}{\langle repeat\ x:=x-1\ until\ x=0, [x:=2]\rangle \Downarrow [x:=0]}}{\langle x:=2;\ repeat\ x:=x-1\ until\ x\leq 0, [\,]\rangle \Downarrow [x:=0]}$$

Textual Representation:

$$\frac{\dfrac{< 2, [\,] >\rightarrow 2}{< x := 2, [\,] >\rightarrow [x := 2]} \qquad \dfrac{< x := x - 1, [x := 2] >\rightarrow [x := 1] < x \leq 0; [x := 1] \rightarrow \mathrm{false} < \mathrm{repeat}\ x := x - 1\ \mathrm{until}\ x \leq 0, [x := 1] >\rightarrow [x := 0]}{< \mathrm{repeat}\ x := x - 1\ \mathrm{until}\ x = 0, [x := 2] >\rightarrow [x := 0]}}{< x := 2; \mathrm{repeat}\ x := x - 1\ \mathrm{until}\ x \leq 0, [\,] > \rightarrow [x := 0]}$$

d) Answer:

Using semantics from part b) we need to prove **“repeat S until b”** semantically equal to **“S ; if b then skip else (repeat S until b)”**

- **Case 1:**

$$\frac{<S,s>\to s'}{<\text{repeat S until b},\ \ s>\to s'}$$

by axiom $< \text{skip}, s > \to s'$

$$\frac{< \text{skip}, s > \to s'}{< \text{if b then skip else (repeat S until b)},\ s' > \to\ s'}$$

$$\frac{< S, s > \to s', < \text{if b then skip else (repeat S until b)},\ s' > \to s'}{< S;\ \text{if b then skip else (repeat S until b)},\ s' > \to s'}$$

- **Case 2:**

$$\frac{< S, s > \to s', < \text{repeat S until b},\ s' > \to s''}{< \text{repeat S until b}, s > \to s''}$$

$$\frac{< \text{repeat S until b},\ s' > \to s''}{\text{if b then skip else (repeat S until b)},\ s' > \to s''}$$

$$\frac{< S, s > \to s', < \text{if b then skip else (repeat S until b)},\ s' > \to s''}{< S;\ \text{if b then skip else (repeat S until b)},\ s' > \to s''}$$

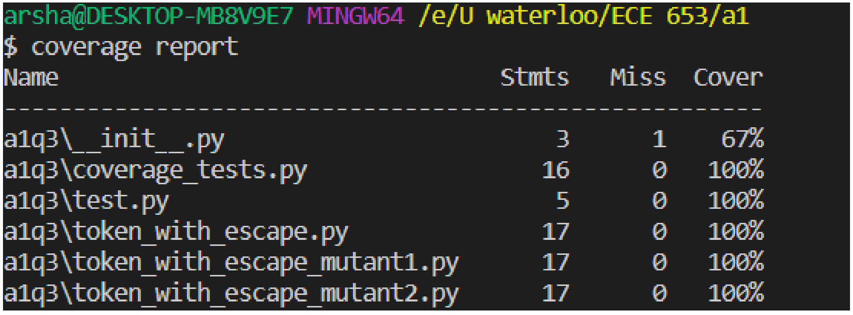
By combination of case 1 and 2 we get the prove of “repeat S until b” semantically equal to “S ; if b then skip else (repeat S until b)”

Question 3:

Discuss whether these are useful test cases—do they say something useful about expected program behaviour beyond what you got from the tests in part (b).

Answer:

1. The test succeeded in the original program whereas there has been failure detected due to mutation created in the “token_with_escape_mutant1” and “token_with_escape_mutant2”:



```
arsha@DESKTOP-MB8V9E7 MINGW64 /e/U waterloo/ECE 653/a1
$ coverage report
Name                                  Stmts   Miss  Cover
---------------------------------------------------------
a1q3\__init__.py                          3      1    67%
a1q3\coverage_tests.py                   16      0   100%
a1q3\test.py                              5      0   100%
a1q3\token_with_escape.py                17      0   100%
a1q3\token_with_escape_mutant1.py        17      0   100%
a1q3\token_with_escape_mutant2.py        17      0   100%
```

2. The tests cases indicate that with creation of mutants the list differ from the expected list from the original program code.
3. The non-stillborn mutant which is syntactically correct and runs the coverage test however the program code provided incorrect results.

Question 4:

Answer:

- int.py:75 -> there is no possible to turn to assert False
- parser.py:172 -> the Boolean exp unable to allocate to the node.
- parser.py:452 -> Unable to create a new line for pattern
- parser.py:453 -> Unable to perform cut function for new line of pattern
- ast.py:158 -> Unable to take [0] value of the list.