ECE653 - Assignment 1

Xu Xiongyi, 20960138
x447xu@uwaterloo.ca

Question 1

a. 13

<a random integer input is 0/<>

b. test case

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

b = [[8, 1], [4, 2]]

c. no possible test case

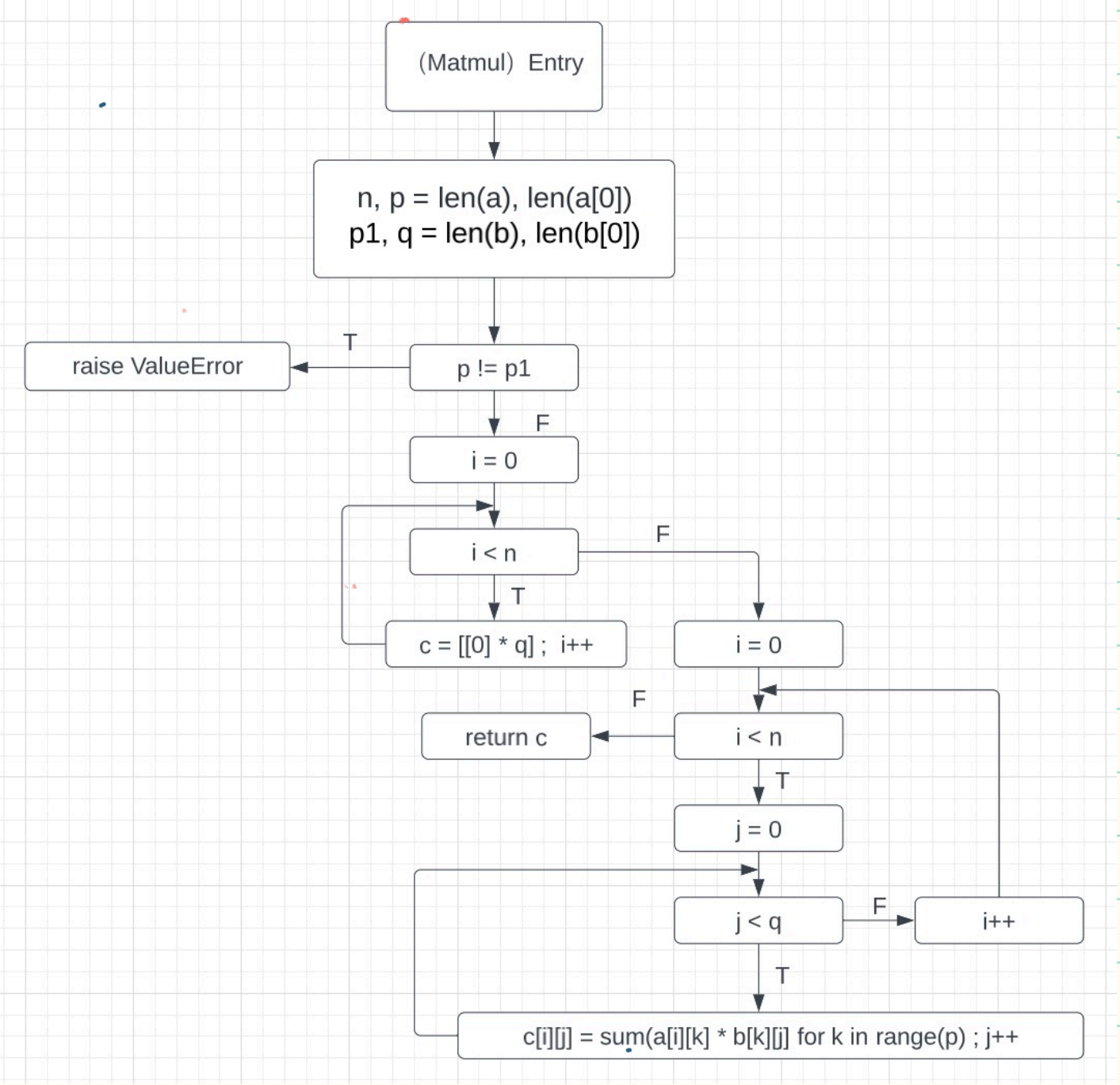
d.

Incorrect Program

State 0	State 1	State 2
a = [[5, 7], [8, 21]]	a = [[1, 7], [8, 21]]	a = [[5, 7], [8, 21]]
b = [[8], [4]]	b = [[8], [4]]	b = [[8], [4]]
n = undefined →	n = 2	⇒ n = 2
p = undefined	p = 2	p = 2
q = undefined	q = 2	q = 2
p_1 = undefined	p_1 = 1	p_1 = 1
pc = matmul(...)	pc = before P != P1	pc = P != P_1
		raise ValueError

n = 2, p = 2, q = 2, p_1 = 1

Qe


(Matmul) Entry
n, p = len(a), len(a[0])
p1, q = len(b), len(b[0])
p != p1
T
raise ValueError
F
i = 0
i < n
F
T
c = [[0] * q] ; i++
i = 0
F
i < n
return c
T
j = 0
j < q
F
i++
T
c[i][j] = sum(a[i][k] * b[k][j] for k in range(p) ; j++

Assignment 1 Question 2

(a)

```
class RepearUntilStmt(Stmt):
    def __init__(self):
        self.cond = cond
        self.body = body
```

(b)

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

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

(c)

Set "repeat $x := x-1$ until $x \le 0$" as "REPEAT"

$$\frac{\langle x, [x:=1]\rangle \Downarrow 1 \qquad \langle 0, [x:=1]\rangle \Downarrow 0}{\dfrac{\langle x, [x:=2]\rangle \Downarrow 2 \qquad \langle 1, [x:=2]\rangle \Downarrow 1 \qquad \langle x-1, [x:=1]\rangle \Downarrow 0}{\dfrac{\langle x-1, [x:=2]\rangle \Downarrow 1 \qquad \langle x:=x-1, [x:=1]\rangle \Downarrow [x:=0] \qquad \langle x \le 0, [x:=0]\rangle \Downarrow \text{true}}{\dfrac{\langle x:=x-1, [x:=2]\rangle \Downarrow [x:=1] \qquad \langle x \le 0, [x:=0]\rangle \Downarrow \text{false} \qquad \langle \text{Repeat}, [x:=1]\rangle \Downarrow [x:=0]}{\dfrac{\langle x:=2, [\,]\rangle \Downarrow \langle x:=2\rangle \qquad \langle \text{REPEAT}, [x:=2]\rangle \Downarrow [x:=0]}{\langle x:=2 \text{ repeat } x:=x-1 \text{ until } x \le 0, [\,]\rangle \Downarrow [x:=0]}}}}$$

(d) if $B(b) = false$, $B(\neg b) = true$

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

By the induction hypothesis, $\langle repeat\ S\ until\ b, q'\rangle \to q''$ is equivalent to $\langle S; while\ \neg b\ do\ S, q'\rangle \to q''$

Therefore, we can deduce like:

$$\frac{\langle S, q\rangle \Downarrow q' \qquad \dfrac{\langle \neg b, q'\rangle \Downarrow true \quad \langle S; while\ \neg b\ do\ S, q'\rangle \Downarrow q''}{\langle while\ \neg b\ do\ S, q'\rangle \Downarrow q''}}{\langle S; while\ \neg b\ do\ S, q\rangle \Downarrow q''}$$

Also, if $B(b) = true$, which means $B(\neg b) = true$

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

By the induction hypothesis, $\langle repeat\ S\ until\ b, q\rangle \to q'$ is equivalent to $\langle S; while\ \neg b\ do\ S, q\rangle \to q'$

$$\frac{\langle S, q\rangle \Downarrow q' \qquad \dfrac{\langle \neg b, q'\rangle \Downarrow false \quad \langle while\ \neg b\ do\ S, q'\rangle \Downarrow q'}{\langle while\ \neg b\ do\ S, q'\rangle \Downarrow q'}}{\langle S; while\ \neg b\ do\ S, q\rangle \Downarrow q'}$$

In Summary, In both case

if $\langle repeat\ S\ until\ b, q\rangle \to q'$ then $\langle S; while\ \neg b\ do\ S, q\rangle \to q'$ (Vice Versa)

Question #3

a. coverage result

```
-------------------------------------------------------
a1q3/__init__.py                      3      1     67%
a1q3/coverage_tests.py               20      0    100%
a1q3/test.py                          5      0    100%
a1q3/token_with_escape.py            17      0    100%
a1q3/token_with_escape_mutant1.py    17      1     94%
a1q3/token_with_escape_mutant2.py    17      8     53%
-------------------------------------------------------
TOTAL                                79     10     87%
(venv)  xuxiongyi@v1040-wn-rt-a-25-122  ~/Downloads/skeleton-main-653/a1
```

b Two mutant Code

```python
def token_with_escape_mutant1(inpt, escape="^", separator="|"):
    """
    Issue  python -m doctest thisfile.py  to run the doctests.

    >>> print(token_with_escape_mutant1('one^|uno||three^^^^|four^^^|^cuatro|'))
    ['one|uno', '', 'three^^', 'four^|cuatro', '']
    """
    result = []
    token = ""
    state = 0
    for c in inpt:
        if state == 0:
            if c != escape: # Logical Connector Replacement
                state = 1
            elif c != separator:   # Logical Connector Replacement
                result.append(token)
                token = ""
            else:
                token += c
        elif state == 1:
            token += c
            state = 0
    result.append(token)
    return result
```

Second One:

```python
def token_with_escape_mutant2(inpt, escape="^", separator="|"):
    """
    Issue  python -m doctest thisfile.py  to run the doctests.

    >>> print(token_with_escape_mutant2('one^|uno||three^^^^|four^^^|^cuatro|'))
    ['one|uno', '', 'three^^', 'four^|cuatro', '']
    """
    result = []
    token = ""
    state = 0
    for c in inpt:
        if state != 0:  # Logical Connector Replacement
            if c == escape:
                state = 1
            elif c == separator:
                result.append(token)
                token = ""
            else:
                token += c
        elif state == 1:
            token += c
            state = 0
    result.append(token)
    return result
```

c. Origin works while two mutants fails

```
(venv)  xuxiongyi@v1040-wn-rt-a-25-122  ~/Downloads/skeleton-main-653/a1  coverage run -m a1q3.test
['n^uo|he^', '', '', 'fu^', '', '^uto']
Print the execution of test_kill_mutant_1(design_input)
FPrint the execution of test_kill_mutant_2(design_input)
F['one|uno', '', 'three^^', 'four^|cuatro', '']
```

Origin works

```
Traceback (most recent call last):
  File "/Users/xuxiongyi/Downloads/skeleton-main-653/a1/a1q3/coverage_tests.py", line 26, in test_kill_mutant_1
    self.assertEqual(text1,text2,message)
AssertionError: Lists differ: ['one|uno', '', 'three^^', 'four^|cuatro', ''] != ['n^uo|he^', '', '', 'fu^', '', '^uto']

First differing element 0:
'one|uno'
'n^uo|he^'
```

```
FAIL: test_kill_mutant_2 (a1q3.coverage_tests.CoverageTests)
Kill mutant 2
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/xuxiongyi/Downloads/skeleton-main-653/a1/a1q3/coverage_tests.py", line 42, in test_kill_mutant_2
    self.assertEqual(text3,text4,message)
AssertionError: Lists differ: ['one|uno', '', 'three^^', 'four^|cuatro', ''] != ['']
```

two mutant fails

Question 4

Coverage report: 96%

coverage.py v6.4, created at 2022-06-04 06:49 +0800

Module	statements	missing	excluded	branches	partial	coverage
wlang/__init__.py	0	0	0	0	0	100%
wlang/ast.py	268	1	0	70	1	99%
wlang/int.py	126	13	0	60	4	91%
wlang/parser.py	358	22	0	14	2	92%
wlang/semantics.py	63	0	0	10	0	100%
wlang/stats_visitor.py	56	7	0	12	1	88%
wlang/test.py	5	0	0	4	1	89%
wlang/test_int.py	98	0	0	10	0	100%
wlang/test_stats_visitor.py	72	0	0	2	0	100%
wlang/test_undef_visitor.py	57	0	0	2	0	100%
wlang/undef_visitor.py	45	0	0	12	0	100%
Total	**1148**	**43**	**0**	**196**	**9**	**96%**

coverage repore

① → Explanation for uncovered line in ast.py

```
157     def __init__(self, op, args):
158         if isinstance(op, list):
159             self.op = op[0]
160         else:
```

And for other uncovered line, I admit that I did not find the right test case to cover them.

② Explanation for uncovered in int.py

```
178 def _parse_args():
179     import argparse
180
181     ap = argparse.ArgumentParser(prog="int", description="WLang Interpreter")
182     ap.add_argument("in_file", metavar="FILE", help="WLang program to run")
183     args = ap.parse_args()
184     return args
185
186
187 def main():
188     args = _parse_args()
189     prg = ast.parse_file(args.in_file)
190     st = State()
191     interp = Interpreter()
192     interp.run(prg, st)
193     return 0
194
195
196 if __name__ == "__main__":
197     sys.exit(main())
```

Explanation:

The majority of the uncovered line is shown at bottom, namely from def main section.

The occurrence of this uncovered line is due to python operation principle. Since the this operation is started from test_in.py , the function entry of parser.py would not be triggered. Therefore, those line won't be reached.

```
72          if node.op == ">":
73              return lhs > rhs
74
75          assert False
76
```

Explanation:

For this uncovered line, we could see from the yellow line and indication shows that line 72 can not jump to 75 because 72 never fail. Therefore assert False will never be true.

③ Explanation for uncovered in parse.py .

```
def main(filename, start=None, **kwargs):
    if start is None:
        start = 'start'
    if not filename or filename == '-':
        text = sys.stdin.read()
    else:
        with open(filename) as f:
            text = f.read()
    parser = WhileLangParser()
    return parser.parse(text, rule_name=start, filename=filename, **kwargs)


if __name__ == '__main__':
    import json
    from tatsu.util import asjson

    ast = generic_main(main, WhileLangParser, name='WhileLang')
    print('AST:')
    print(ast)
    print()
    print('JSON:')
    print(json.dumps(asjson(ast), indent=2))
    print()
```

1. As for the parser.py

The majority of the uncovered line is shown at bottom, namely from def main section.

The occurrence of this uncovered line is due to python operation principle. Since the this operation is started from test_in.py , the function entry of parser.py would not be triggered. Therefore, those line won't be reached.