L170300901-卢兑玧-高级软件测试(AliMili)-Final Exam

1. **Software Specifications, Chapter 4.**
In addition to the logical quantifiers (∀ and ∃) there exists a numeric quantifier (#), which returns (not a logical value TRUE/ FALSE, but) a number: the number of elements for which the quantified predicate is true. For example, in reference to the array below:

1	2	3	4	5	6	7	8	9	10	11
0.9	3.14	2.1	0.9	3.1	5.7	2.89	0.9	2.5	0.9	1.2

the following equations hold:

$$(\#k: a[k] = 0.9) = 4,$$
$$(\#k: a[k] \geq 3) = 3,$$
$$(\#k: a[k] \geq 2) = 6,$$
$$(\#k: a[k] \leq 10) = 11.$$

Use the (#) quantifier to help write the following specification: place in x the smallest value of a and in k a median index where the smallest value appears in a. In the above array, x gets value 0.9 and k gets value 4 or 8.

1. ① page

Serveral clauses

① Place in X a value that exceeds all the cells of a:

$M1 = \{(s,s') \mid \forall h\ 1 \leq h \leq N : x' \geq a[h]\}$

② Place in X a value of the array

$M2 = \{(s,s') \mid \exists h : 1 \leq h \leq N : x' = a[h]\}$

③ Place in k an index of x' in a:

$M3 = \{(s,s') \mid x' = a[k']\}$

④ Ensure that k' is the smallest index:

$M4 = \{(s,s') \mid \forall h\ 1 \leq h < k' : x' \neq a[h]\}$.

Overall specification:

$M = M1 \cap M2 \cap M3 \cap M4$

$M_1 = \{(s,s') \mid (\forall h: 1 \leq h \leq N : x' \leq a[h]) \wedge$
$(\exists l : 1 \leq l \leq N : x' = a[l])\}$

$M_2 = \{(s,s') \mid ((\# j : a[j] = x') \geq 3) \wedge$
$(\exists h : a[h] = x' \wedge h < k') \setminus$
$\wedge (\exists l : a[l] = x' \wedge l > k') \wedge a[k'] = x')$

Overall specification:

$M = M_1 \cap M_2$

2. Correctness Verification, Chapter 5.

Consider the following program on integer variables x, y, z.

```
mult:
{z=0;
 while (y!=0)
      {if (y%2==0) {x=2*x; int1:  y=y/2;}
       else        {z=z+x; int2:  y=y-1;}}}
```

We propose to prove the correctness of this program with respect to:

Precondition: $x = x0 \wedge y = y0$.

Postcondition: $z = x0 \times y0$.

- **a.** Propose a formula for the loop invariant, *inv*.
- **b.** Propose a formula for the intermediate assertion at label *int*1.
- **c.** Propose a formula for the intermediate assertion at label *int*2.
- **d.** Prove the correctness of this program with respect to the proposed specification.

2

V: $\{x=x_0\ y=y_0\}$ $\{z=0$, while $(y!=0)\}$
if $(y\%2==0)$ $\{x=2\times x;$
int 1 $y=y/2;\}$
else $\{z=z+x;$ int 2 $y=y-1;)\}$
$\}$ $\{z=x_0\times y_0\}$

$\text{into} = x=x_0 \wedge y=y_0 \wedge z=0$

We must prove 2 formulas

V_0 $\{x=x_0 \cap y=y_0\}$ $z=0;$ $\{x=x_0 \cap y=y_0 \cap z=0\}$

→ We must prove $\{x=x_0\ \&\ y=y_0\} \rightarrow \{x=x_0\ \&\ y=y_0\ \&\ 0=0\}$ ✓

V_1 $\{x=x_0 \cap y=y_0\ \&\ z=0\}$ while $(y!=0)$ $\{$if$(y\%2==0)$ $\{x=2*x;$
int 1 $y=y/2;\}$
else $\{z=z+x;$ int2: $y=y-1;)\}\}$
$\}$ $\{$
$z=x_0\times y_0$
$\}$

$\text{inv} = \{x\times y+z = x_0\times y_0\}$

To prove V_1, we must prove

V_{10}. $\{x=x_0 \wedge y=y_0 \wedge z=0\} \rightarrow \{x\times y+z = x_0\times y_0\}$ This is a tautology ✓

V_{11}. $\{x\times y+z = x_0\times y_0 \wedge y!=0\}$ if $(y\%2==0)$ $\{x=2*x;$
int 1: $y=y/2;\}$
else $\{z=z+x;$ int2 $y=y-1;)$
$\{x\times y+z = x_0\times y_0\}$

according to the alternation Statement Rule:

V_{110}. $\{x \times y + z = x_0 \times y_0 \wedge y != 0 \wedge y \% 2 == 0\}$
$x = 2 * x$; int 1 : $y = y/2$; $\{x \times y + z = x_0 * y_0\}$

according to the statement Rule:
int 1 $\{x \times y + z = 2 \times x_0 \times y_0 - z\}$
V_{1100} $\{x \times y + z = x_0 \times y_0 \wedge y != 0 \wedge y \% 2 == 0\}$
$x = 2 * x$; $\{x \times y + z = 2 \times x_0 \times y_0 - z\}$

then we must prove
$\{x \times y + z = x_0 \times y_0 \wedge y != 0 \wedge y \% 2 == 0\} \rightarrow$
$\{2 \times x \times y + z = 2 * x_0 \times y_0 - z\}$

V_{1101}. $\{x \times y + z = 2 \times x_0 + y_0 - z\}$ $y = y/2$; $\{x \times y + z = x_0 \times y_0\}$
then we must prove
$\{x \times y + z = 2 \times x_0 \times y_0 - z\} \rightarrow \{x \times y/2 + z = x_0 \times y_0\}$

V_{111} $\{x \times y + z = x_0 \times y_0 \wedge y != 0 \wedge y \% 2 == 1\}$
$z = z + x$; int2 $y = y + 1$; $\{x \times y + z = x_0 \times y_0\}$
according to the sequence Statement Rule:
int2 : $\{x \times y + z = x_0 \times y_0 + x\}$
V_{1110} $\{x \times y + z = x_0 \times y_0 \wedge y != 0 \wedge y \% 2 == 1\}$ $z = z + x$
$\{x \times y + z = x_0 \times y_0 + x\}$
then we must prove
$\{x \times y + z = x_0 \times y_0 \wedge y != 0 \wedge y \% 2 == 1\}$
$\rightarrow \{x + y + z + x = x_0 y_0 + x\}$

V_{1111} $\{x \times y + z = x_0 \times y_0 + x\}$ $y = y + 1$; $\{x \times y + z = x_0 \times y_0\}$
then we must prove $\{x \times y + z = x_0 \times y_0 + x\}$
$\rightarrow \{x \times (y+1) + z = x_0 \times y_0\}$

V_{12} $\{x \times y + z = x_0 \times y_0 \wedge y == 0\} \rightarrow \{z = x_0 \times y_0\}$

Prove up//

3. Functional Criteria of Test Data Generation, Chapters 8, 9.

a. Consider the sorting program given on page 29 of chapter 8. Generate the following mutants of this program:

- m6: obtained by replacing (N-2) by (N-1) in line 4.
- m7: obtained by replacing (N-1) by (N-2) in line 7.
- m8: obtained by replacing {minval=a[j]} by {minval=a[i]} in line 6.

Run program P and each of the mutants (m6, m7, m8) on the test data T given in page 30 and draw a table similar to that of page 35. Did the test data distinguish all the mutants? If it did not, is that because undistinguished mutants are equivalent to P or because the test data is inadequate? If the test data is inadequate, propose other tests.

b. Consider the following symbols, and write code to generate these symbols randomly, according to the proposed probability distribution. Run this code in a loop that iterates 100 000 times and show how many of each symbol it produces.

a	b	c	d	e	f	g	h	i	J
0.12	0.08	0.1	0.09	0.11	0.15	0.06	0.03	0.01	0.25

3.a)

T	m6	m7	m8
t_1	True	True	True
t_2	True	True	True
t_3	True	True	True
t_4	True	False	True
t_5	True	True	True
t_6	True	True	True
t_7	True	False	True
t_8	True	False	True
mutant distinguished?	No	Yes	No

All mutants not distinguished by the test data are equivalent to P.

3- b.

```python
import random

def generate_symbol():
    ra = random.random()
    if ra < 0.12:
        return 'a'
    elif ra < 0.2:
        return 'c'
    elif ra < 0.3:
        return 'c'
    elif ra < 0.39:
        return 'f'
    elif ra < 0.5:
        return 'e'
    elif ra < 0.65:
        return 'f'
    elif ra < 0.71:
        return 'g'
    elif ra < 0.74:
        return 'h'
    elif ra < 0.75:
        return 'i'
    else:
        return 'j'

if __name__=='__main__':
    result = {'a' : 0, 'b' : 0, 'c' : 0, 'd' : 0, 'e' : 0, 'f' : 0, 'g' : 0, 'h' : 0, 'i' : 0, 'j' : 0}
    for _ in range(100_000):
        random_symbol = generate_symbol()
        result[random_symbol] = result[random_symbol] + 1
    print(result)
```

```
PS C:\VSCodeWorkSpace\Python> & C:/Users/dnjsd/AppData/Local/Programs/Python/Python37/python.exe c:/VSCodeWorkSpace/Python/test1.py
{'a': 11994, 'b': 0, 'c': 17987, 'd': 0, 'e': 11032, 'f': 23872, 'g': 6027, 'h': 3043, 'i': 965, 'j': 25080}
PS C:\VSCodeWorkSpace\Python>
```

4. Structural Criteria of Test Data Generation, Chapter 10.

a. Consider the following program on integer variables x, y, z (and implicit variables is and os, for input stream and output stream).

```
mult:
{read(x); read(y); z=0;
 while (y!=0)
     {if (y%2==0) {x=2*x; y=y/2;}
      else        {z=z+x; y=y-1;}}}
```

Compute the path function then the path condition of the following paths:

```
p0: read(x); read(y); z=0; ((y!=0)?false);
p1: read(x); read(y); z=0;
    ((y!=0)?true); ((y%2==0)? true) {x=2*x; y=y/2;}
    ((y!=0)?true); ((y%2==0)? false) {z=z+x; y=y-1;}
    ((y!=0)?false);
p2: read(x); read(y); z=0;
    ((y!=0)?true); ((y%2==0)? false) {z=z+x; y=y-1;}
    ((y!=0)?true); ((y%2==0)? true) {x=2*x; y=y/2;}
    ((y!=0)?true); ((y%2==0)? false) {z=z+x; y=y-1;}
    ((y!=0)?false);
p3: read(x); read(y); z=0;
    ((y!=0)?true); ((y%2==0)? true) {x=2*x; y=y/2;}
    ((y!=0)?true); ((y%2==0)? true) {x=2*x; y=y/2;}
    ((y!=0)?true); ((y%2==0)? false) {z=z+x; y=y-1;}
    ((y!=0)?false);
p4: read(x); read(y); z=0;
    ((y!=0)?true); ((y%2==0)? true) {x=2*x; y=y/2;}
    ((y!=0)?true); ((y%2==0)? false) {z=z+x; y=y-1;}
    ((y!=0)?true); ((y%2==0)? true) {x=2*x; y=y/2;}
    ((y!=0)?false);
```

b. Draw a table of definitions and uses of the program for variables x, y, z. (you may want to write the program one statement per line, and number the lines for ease of reference).

c. Choose a du-path in this program for variable z, then generate test data to exercise the selected path.

④ P0. path function

$$\left\{ (s,s') \;\middle|\; \begin{array}{l} length(is) \geq 2 \wedge h(t(is)) = 0 \wedge \\ x' = h(is) \wedge y' = h(t(is)) \wedge \\ is' = t^2(is) \wedge z' = 0 \wedge os' = os \end{array} \right\}$$

path condition

$length(is) \geq 2 \wedge h(t(is)) =$

P1. path function

$$\left\{ (s,s') \;\middle|\; \begin{array}{l} length(is) \geq 2 \wedge h(t(is)) = 2 \wedge \\ x' = h(is) \times 2 \wedge y' = (h(t(is))/2+1) \wedge \\ is' = t^2(is) \wedge z' = 2 \times h(is) \wedge os' = os \end{array} \right\}$$

path condition

$Length(is) \geq 2 \wedge h(t(is)) = 2$

P2. path function

$$\left\{ (s,s') \;\middle|\; \begin{array}{l} length(is) \geq 2 \wedge h(t(is)) = 3 \wedge \\ x' = h(is) \times 2 \wedge y' = (h(t(is))/2+1) \wedge \\ is' = t^2(is) \wedge z' = 3 \times h(is) \wedge os' = os \end{array} \right\}$$

path condition

$length(is) \geq 2 \wedge h(t(is)) = 3$

P3. path function

$$\left\{ (s,s') \;\middle|\; \begin{array}{l} length(is) \geq 2 \wedge h(t(is)) = 4 \wedge \\ x' = h(is) \times 4 \wedge y' = (h(t(is))/4+1) \wedge \\ is' = t^2(is) \wedge z' = 4 \times h(is) \wedge os' = os \end{array} \right\}$$

path condition

$Length(is) \geq 2 \wedge h(t(is)) = 4$

P4. path function: $\emptyset$

path condition: $\emptyset$

b.

Line		x D	x C	x P	x T	D	C	P	T	z D	z C	z P	z T
1	{read(x);	V											
2	read(y);					V							
3	z=0;									V			
4	while (y!=0)							V					
5	{if(y%2==0)						V	V					
6	{x=2*x;	V	V										
7	y=y/2;}					V	V						
8	else {												
9	z=z+x;		V							V	V		
10	y=y-1; }					V	V						
11	}												
12	}			V					V				V

c. for c:
definitions: lines 3,9
uses line 9
definition use path: [3, 4, 8, 9]
pre-path: empty
post-path: infinity, we choose one at will.
y=y-1; ((y!=0) ? false);
Test Data: is = (3, 1, ···);

5. **Test Oracles and Test Driver Design, Chapters 11, 12.**

Consider the space defined by an integer variable x, an integer array a [1..N], and an index variable k (between 0 and N+1).

a. Write a specification R that provides for placing in k an index where x occurs in a, assuming that x does occur in a.
b. Write an (acceptance testing) oracle that checks for correctness with respect to specification R.
c. Write a program P that searches for x in a starting at the lower end of the array, assuming that x does appear in a.
d. Write a (fault repair) oracle that checks that program P does perform as intended by the programmer (re: question c).

6. Test Outcome Analysis, Chapter 13.

Consider a program whose execution history is recorded in the following table:

Number of faults repaired	Number of executions before the next failure
0	12
1	31
2	30
3	98
4	342
5	875
6	2321

a. Using the reliability model $MTTF_N = MTTF_0 \times R^N$, estimate the MTTF of this product once the 7[th] fault has been repaired.

b. Estimate the probability that this product will run for 2400 times without failure.

Date Page

(6).

a.

N	Inter-failure Run	Log
0	12	1.08
1	31	1.49
2	30	1.48
3	98	1.99
4	342	2.53
5	875	2.94
6	2321	3.37

$\log(MTTF_0) = 0.97 \rightarrow MTTF_0 = 9.33$

$\log R = 0.39 \rightarrow R = 2.45$

$\therefore MTTF_7 = 9.33 \times 2.45^7 = 4944$

b. $MTTF = 2321$

$\therefore MTTF = \int_0^{\infty} t\lambda e^{-\lambda t}\, dt = \frac{1}{\lambda}$

$\therefore \lambda = \frac{1}{MTTF} = 0.0017$

$R = e^{\lambda t} = e^{-\frac{2400}{2321}} = 0.356$

$\therefore$ the probability is 0.356