ECE 322

SOFTWARE TESTING AND MAINTENANCE

Fall 2015

Assignment #4

SOLUTIONS

Due date: Monday, November 2, 2015 by 3:00 PM
(return to the appropriate box- 2nd floor of ECERF building)

Total: 45 points

5 points

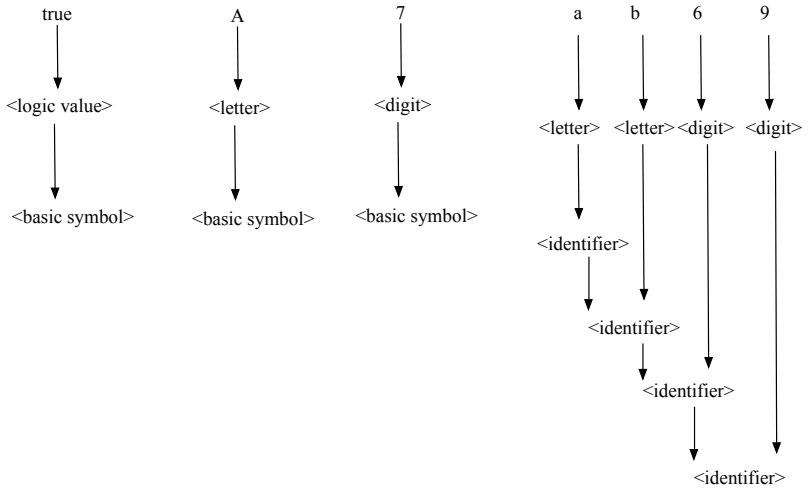
1. Consider the part of ALGOL syntax given in the form of BNF rules

<identifier> ::= letter> | <identifier> <letter> | <identifier> <digit>
<basic symbol> ::= <letter> | <digit> | <logical value>
<letter> ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<logical value> ::= **true** | **false**

Develop a collection of test cases exercising all BNF rules.

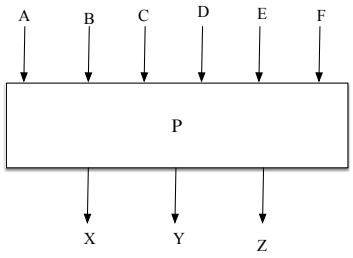
Solution

We develop test cases to exercise all BNF rules here: **true**, A, 7, ab69



20 points

2. Consider the program P with 6 inputs A, B, C, D, E, F and 3 outputs X, Y, and Z. The input variables assume some values shown in the table below.

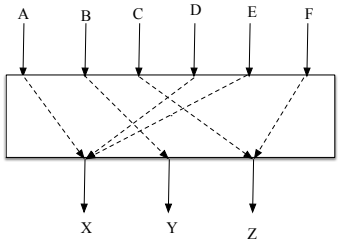


input	values
A	0, 1, 2, 3
B	a, b, c, d, e
C	W, S, E, N
D	2.5, 2.6, 2.7
E	yes, no
F	α, β

Consider the following testing strategies:

(a) *pairwise testing*. Include the details about the sizes of the pertinent orthogonal arrays, say $L_?(?^?)$.

Solution
One has to select an orthogonal table producing the smallest number of test cases $L_{25}(5^6)$.
(b) *input–output analysis*. Consider the situation when (i) you do not know functional dependencies among the input and output variables, and (ii) where these dependencies are known (as shown in the figure below).



What is the number of test cases required when using these two strategies.

Solution
Unknown relationships: 4*5*4*3*2*2 =960 test cases. If the functional relationships are known the upper bound of the number of test cases is 37 (namely, 4*3*2 for x, 5 for y and 4*2 for z).

(c) Assume that all inputs (A, B,…,F) assume one of *n* possible values. Compare the number of test cases required by the pairwise testing and input-output analysis being regarded as a function of *n*. Plot the results (number of test cases) for *n* assuming values 2, 3,…7. Compare the results and draw conclusions.

Solution
We find orthogonal arrays $L_?(n^6)$ for n =2, 3, …7 or the closest to those.

20 points

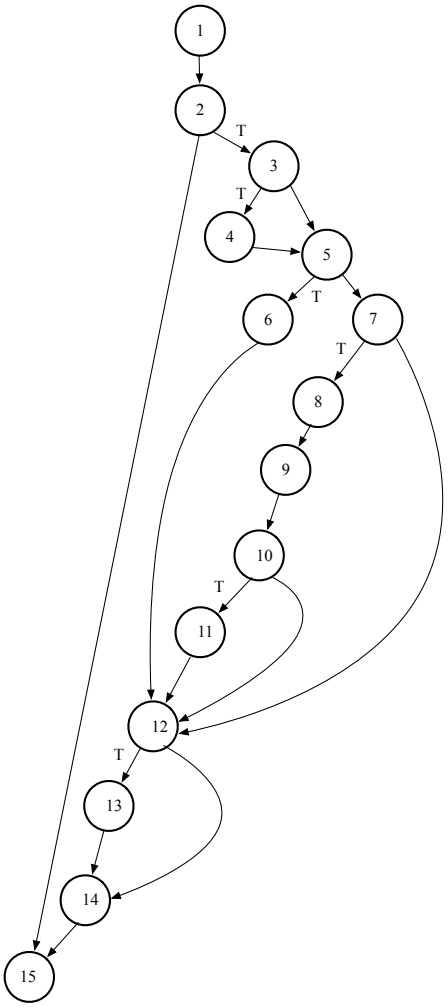
3.For the piece of code shown below do the following:
(a) draw a control flow graph. What is its McCabe complexity measure?

(b) propose a suite of test cases to complete the following coverage criteria: (i) statement coverage, and (ii) branch coverage

Solution

```
#define scan -1
int match(char *a)
  {
    int n1, n2;
1   int j = 0, N = strlen(a), state = next1[0];
    dequeinit(); put(scan);
    while (state)2
      {
        if (state ==3 scan)4 { j++; put(scan); }
        else if (ch[state]5== a[j])
          6put(next1[state]);
        else if (ch[state]7== ' ')
          {
            8n1 = next1[state]; n2 = next2[state];
            9push(n1); if (n1 !10= n2) push(n2)11;
          }
        if (dequeempty() || j==N) return 012;13
        state = pop()14;
      }
    return j;15
  }
```

control flow graph



The cyclomatic complexity is 7:
(i) using the basic formula by counting the number of nodes and edges: 20-15+2 =7, (ii) counting the number of binary decision boxes: 6+1 = 7, (iii) counting the number of planar regions: 7

In choosing a collection of test cases it is to be assumed that in testing there is control over all of the parameters in the scope of the program (int *next1, int*next2, int*ch). Building the test cases involves choosing next1, next2, a, and ch

```
#define scan -1
int match(char *a)
  {
    int n1, n2;
    int j = 0, N = strlen(a), state = next1[0];
    dequeinit(); put(scan);
    while (state)
      {
        if (state == scan) { j++; put(scan); }
        else if (ch[state] == a[j])
            put(next1[state]);
        else if (ch[state] == ' ')
          {
            n1 = next1[state]; n2 = next2[state];
            push(n1); if (n1 != n2) push(n2);
          }
        if (dequeempty() || j==N) return 0;
        state = pop();
      }
    return j;
  }
```