Software Testing and Reliability

Xiaoyuan Xie 谢晓园
xxie@whu.edu.cn
计算机学院E301

Lecture 3

Static Review

Static Analysis

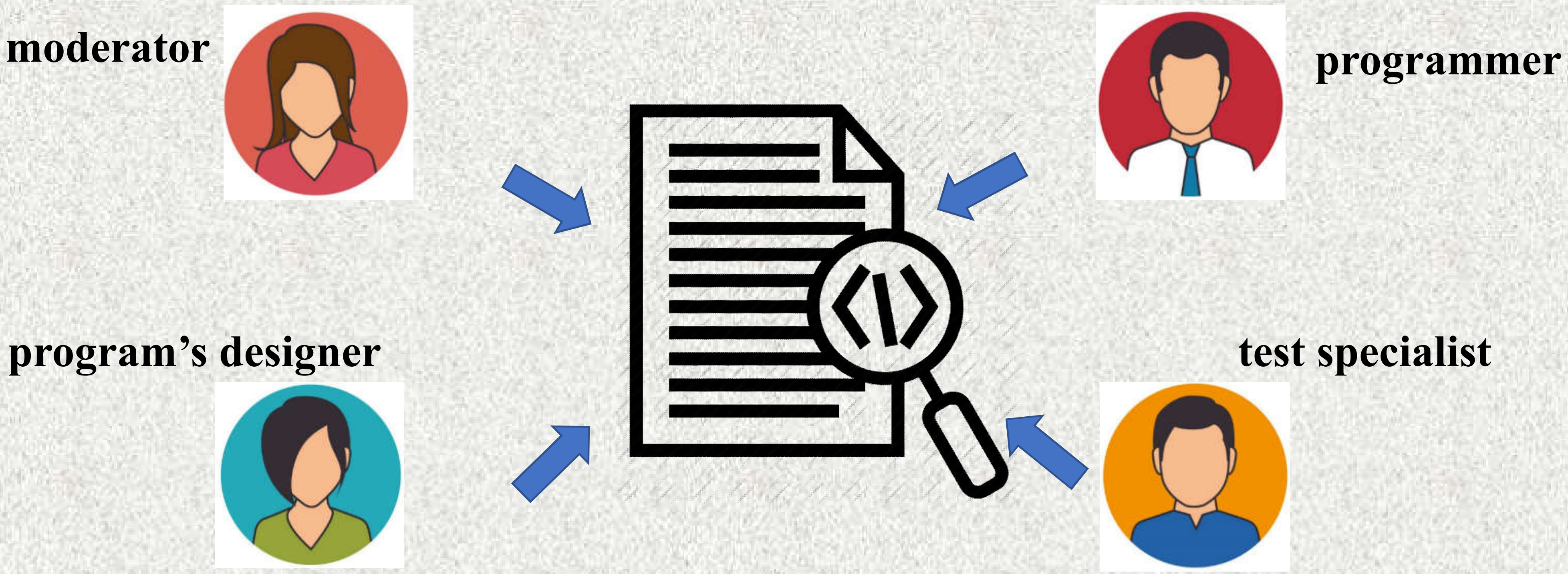
- Analyze the structural properties of programs
- No execution of programs

Characteristics

- Similar to syntactic analysis
- Analysis is input independent
- Difficult to analyze pointers, array element references, etc.
- **Graph theory** is extensively used in static analysis (decidable and undecidable problems in graph theory)

Code Inspection

A set of procedures and error-detection techniques for group code reading.



Code Inspection

Checklist

- Language dependent
- Problem domain dependent
- Programming practice and style

Code Inspection

Examples of checklist items

- Array subscripts within bounds
- Variables initialized before use
- Division by zero
- Integer division
- Mixed mode comparison
- Mixed mode computation
- Operator precedence

Code Inspection

Examples of checklist items (continued)

- Termination of loops
- Termination of programs
- Parameters and arguments match
- Parameter passing mechanisms
- Data types and structures

Code Inspection

Examples of checklist items (continued)

- Programming practice and style
  - Rules to define identifiers
  - Preference of control structures

Inspection Process

- Preparation prior to the inspection meeting
  - Checklist
  - Previous inspection reports
- Inspection meeting
  - to identify errors and anomalies
  - to examine the program with respect to a checklist
- Reports
- Follow-up

Summary

- Different from desk checking
- Programmers identify most of the errors
- Though it is simple in concept, inspection is powerful in revealing errors (with lower debugging costs)
- Preparation prior to meeting is important

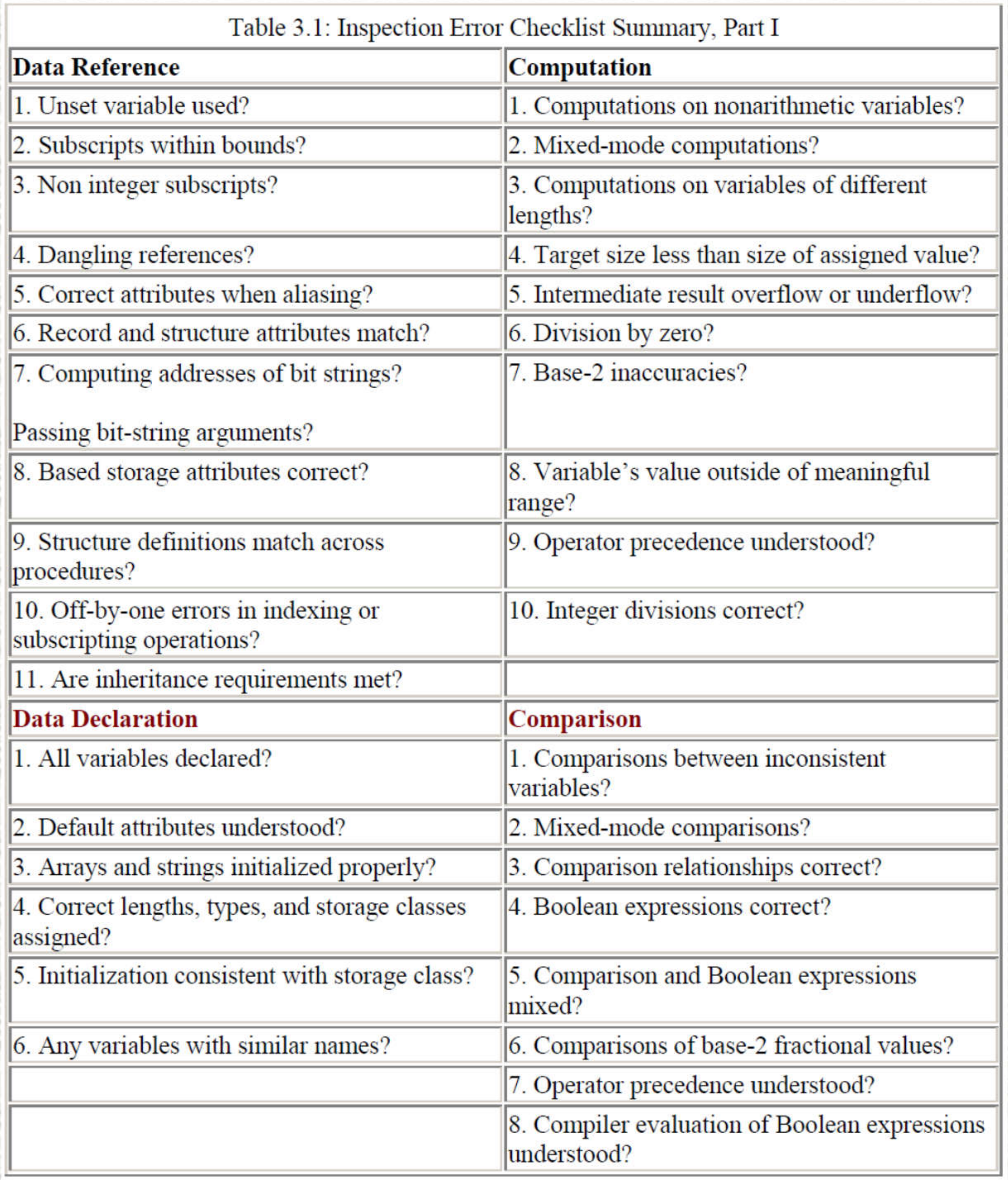


Table 3.1: Inspection Error Checklist Summary, Part I

Data Reference	**Computation**
1. Unset variable used?	1. Computations on nonarithmetic variables?
2. Subscripts within bounds?	2. Mixed-mode computations?
3. Non integer subscripts?	3. Computations on variables of different lengths?
4. Dangling references?	4. Target size less than size of assigned value?
5. Correct attributes when aliasing?	5. Intermediate result overflow or underflow?
6. Record and structure attributes match?	6. Division by zero?
7. Computing addresses of bit strings? Passing bit-string arguments?	7. Base-2 inaccuracies?
8. Based storage attributes correct?	8. Variable's value outside of meaningful range?
9. Structure definitions match across procedures?	9. Operator precedence understood?
10. Off-by-one errors in indexing or subscripting operations?	10. Integer divisions correct?
11. Are inheritance requirements met?	
Data Declaration	**Comparison**
1. All variables declared?	1. Comparisons between inconsistent variables?
2. Default attributes understood?	2. Mixed-mode comparisons?
3. Arrays and strings initialized properly?	3. Comparison relationships correct?
4. Correct lengths, types, and storage classes assigned?	4. Boolean expressions correct?
5. Initialization consistent with storage class?	5. Comparison and Boolean expressions mixed?
6. Any variables with similar names?	6. Comparisons of base-2 fractional values?
	7. Operator precedence understood?
	8. Compiler evaluation of Boolean expressions understood?

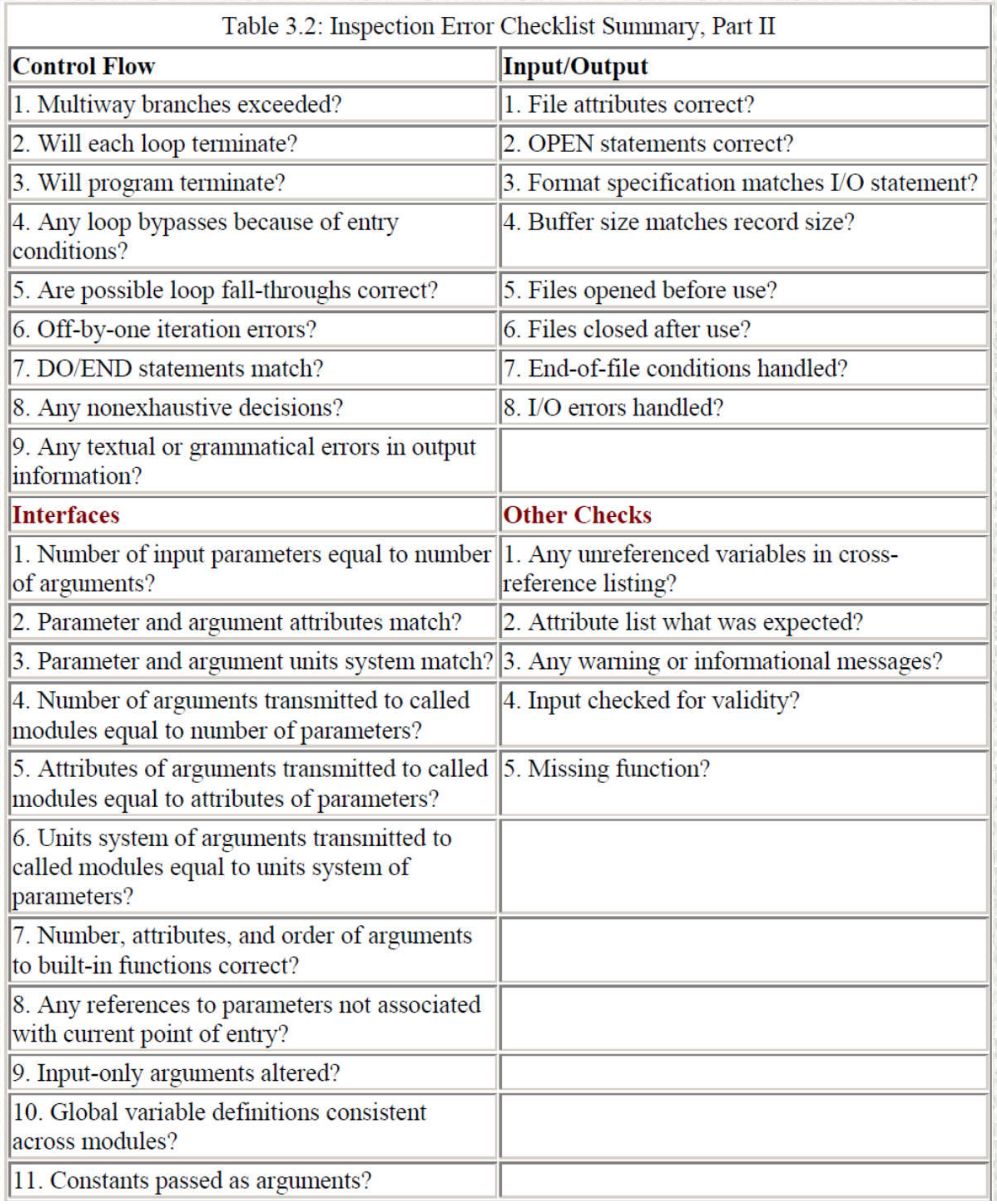
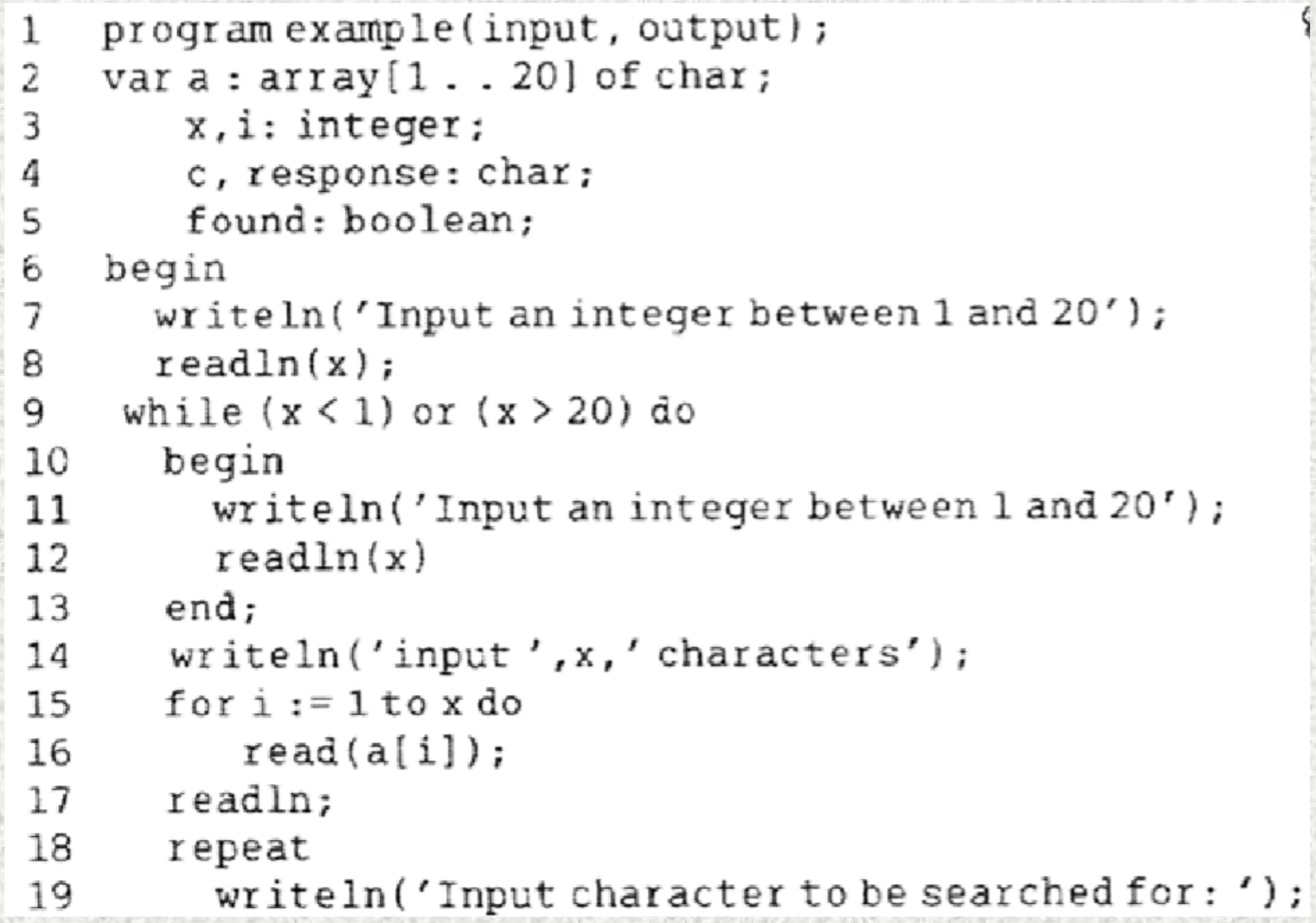


Table 3.2: Inspection Error Checklist Summary, Part II

Control Flow	**Input/Output**
1. Multiway branches exceeded?	1. File attributes correct?
2. Will each loop terminate?	2. OPEN statements correct?
3. Will program terminate?	3. Format specification matches I/O statement?
4. Any loop bypasses because of entry conditions?	4. Buffer size matches record size?
5. Are possible loop fall-throughs correct?	5. Files opened before use?
6. Off-by-one iteration errors?	6. Files closed after use?
7. DO/END statements match?	7. End-of-file conditions handled?
8. Any nonexhaustive decisions?	8. I/O errors handled?
9. Any textual or grammatical errors in output information?	
Interfaces	**Other Checks**
1. Number of input parameters equal to number of arguments?	1. Any unreferenced variables in cross-reference listing?
2. Parameter and argument attributes match?	2. Attribute list what was expected?
3. Parameter and argument units system match?	3. Any warning or informational messages?
4. Number of arguments transmitted to called modules equal to number of parameters?	4. Input checked for validity?
5. Attributes of arguments transmitted to called modules equal to attributes of parameters?	5. Missing function?
6. Units system of arguments transmitted to called modules equal to units system of parameters?	
7. Number, attributes, and order of arguments to built-in functions correct?	
8. Any references to parameters not associated with current point of entry?	
9. Input-only arguments altered?	
10. Global variable definitions consistent across modules?	
11. Constants passed as arguments?	

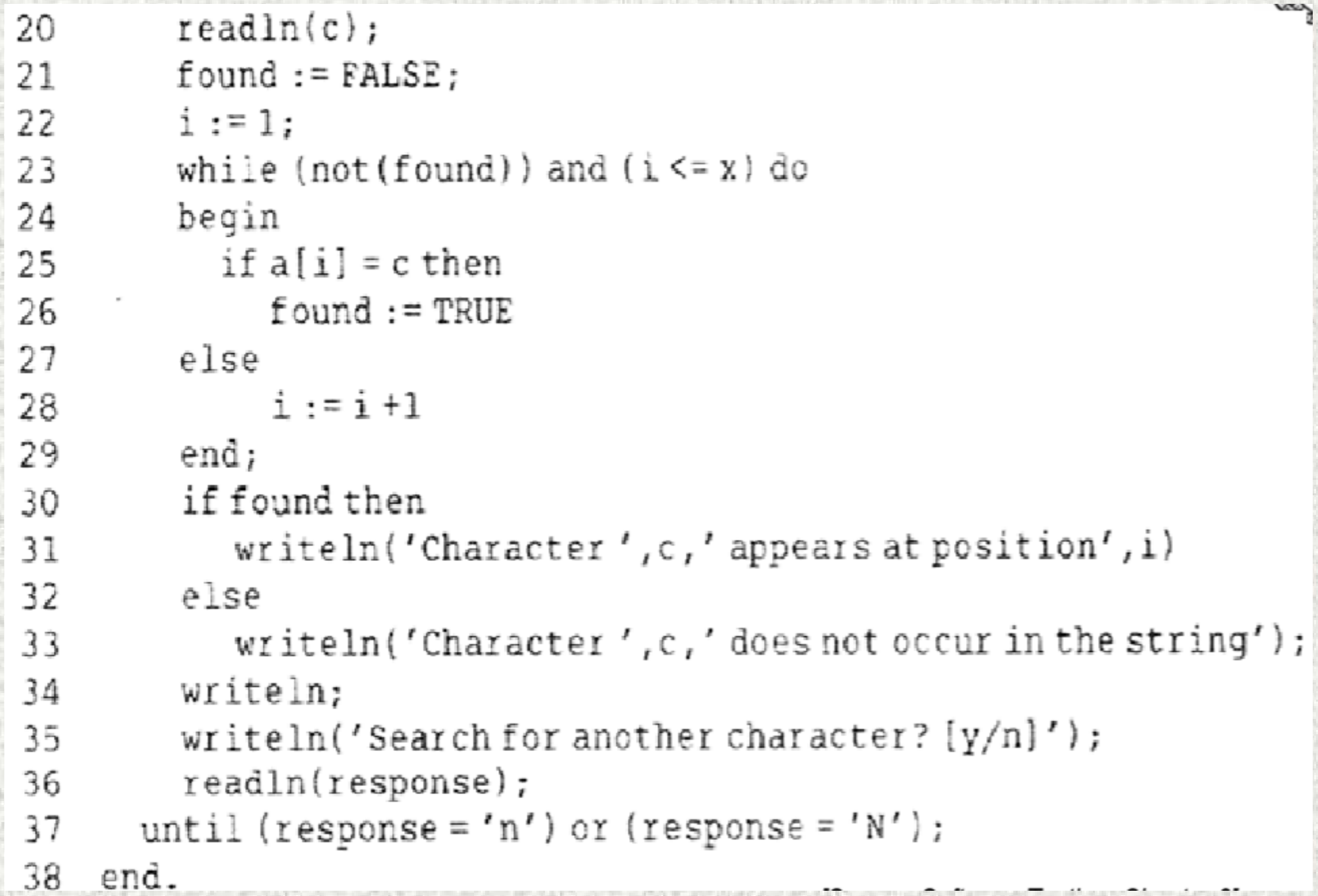
White-box Testing (control-flow coverage)

Program under Study



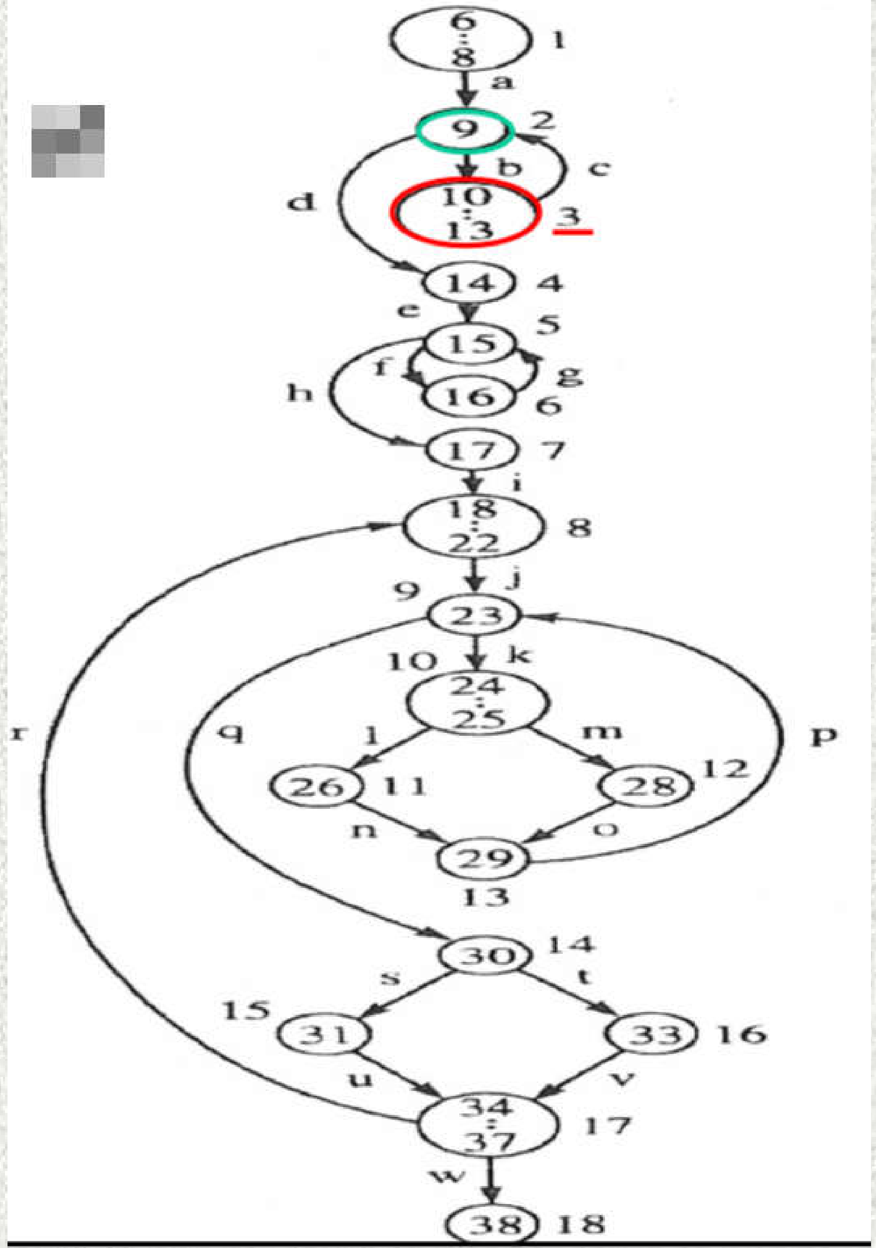
```
program example(input, output);
var a : array[1 . . 20] of char;
    x,i: integer;
    c, response: char;
    found: boolean;
begin
   writeln('Input an integer between 1 and 20');
   readln(x);
 while (x < 1) or (x > 20) do
   begin
      writeln('Input an integer between 1 and 20');
      readln(x)
   end;
   writeln('input ',x,' characters');
   for i := 1 to x do
       read(a[i]);
   readln;
   repeat
      writeln('Input character to be searched for: ');
```

Program under Study (continued)



```
      readln(c);
      found := FALSE;
      i := 1;
      while (not(found)) and (i <= x) do
      begin
        if a[i] = c then
          found := TRUE
      else
          i := i +1
      end;
      if found then
         writeln('Character ',c,' appears at position',i)
      else
         writeln('Character ',c,' does not occur in the string');
      writeln;
      writeln('Search for another character? [y/n]');
      readln(response);
    until (response = 'n') or (response = 'N');
  end.
```

Directed Graph of the Program



Node, Edge and Branch

- Node

  -- A block of statements

  -- For example,

  -- Each node is labelled as a number, that is, 1, 2, 3, …in the previous slide

  -- Node 3 is associated with 10-13th lines of code

- Edge –link between nodes
- Branch

  -- An edge, associated with the true or false branch of a decision node(or called predicate)

  -- For example,

  -- Node 2 (9th line of code) is a decision node, which has two out-going edges (b and d), -- b and d are called the branches of Node 2.

  -- Each edge is labelled as a character, that is, a, b, c, … in the previous slide