1. p1-16 Reliability vs. cost of execution
   - Conflicting criteria
   - Example: Java demands all references to array elements
   - be checked for proper indexing but that increases
   - execution costs

2. p1-12 Imperative languages, most dominant, because of
   - von Neumann computers
   - Data and programs stored in memory
   - Memory is separate from CPU
   - Instructions and data are piped from memory to CPU
   - Basis for imperative languages
   - Variables model memory cells
   - Assignment statements model piping
   - Iteration is efficient (because instructions are stored in adjacent cells of memory)
   - p1-23 Connection speed between a computer’s
   - memory and its processor determines the
   - speed of a computer
   - Program instructions often can be executed
   - a lot faster than the above connection
   - speed; the connection speed thus results in
   - a bottleneck

3. a. False
   b. False
   c. p1-28 (the machine code is kept for subsequent calls)

4.a 畫兩顆不同的 parse tree

4.b
   \[<S> \rightarrow <expr>\]
   \[<expr> \rightarrow <id> + <expr>\]
   \[\mid <id> - <expr>\]
   \[\mid <id>\]
   \[<id> \rightarrow A \mid B \mid C\]

4.c
\(<S> \rightarrow <expr>\)
\(<expr> \rightarrow <id> \{(+|-)<id>\}\)
\(<id> \rightarrow A \mid B \mid C\)

5. a  Phrases: bBab, ab

    Simple phrases: ab
    Handle: ab

5. b  bBA

    Handle: bBa

6. a) pass  b) fail  c) pass

7. the code used to represent character

    ASCII: 1 byte, can represent 128 characters.
    Unicode: 2 bytes, can represent characters from most nature languages

8.  2640, 2820

9. a: (p6-47) the field is named, but any or all the enclosing record names can be

    omitted, as long as the resulting reference is unambiguous.
    b: the lengths of the rows need not be the same
    c: the address of the variable
    d: a language is strongly typed if type errors are always detected.
    e: two variable names can be used to access the same memory location

10. Static scoping: x is 5.

    Dynamic scoping: x is 10

11. p6-62A pointer points to a heap-dynamic variable that has been de-allocated

    ,  False

12. 125

13. a)  a: (1), b: (5), p: (2), variable pointed by p: (7)

    b)  a: (11), b: (10), p: (11), variable pointed by p: (8)

    c) b and the variable pointed by p