

## Solutions for Dragon Book

### **1.1.1 What is the difference between a compiler and an interpreter?**

Whereas compiler translates program with source language to a program with target language, interpreter on the other hand directly executes the operations specified in source program.

### **1.1.2 What are the advantages of: (a) a compiler over an interpreter (b) an interpreter over a compiler?**

- (a) Target program produced by a compiler is usually much faster than an interpreter at mapping inputs to outputs.
- (b) An interpreter can usually give better error diagnostics than a compiler, because it executes source program statement by statement.

### **1.1.3 What advantages are there to a language processing system in compiler produces assembly language rather than machine language?**

Assembly language is easier to produce as output and easier to debug.

### **1.1.4 A compiler that translates a high-level language into another high-level language is called a source-to-source translator. What advantages are there to using C as a target language for a compiler?**

C compilers are available for any platform, which makes your language available on any platform and architecture where C is available. C compilers optimize aggressively as well.

### **1.1.5 Describe some of the tasks that an assembler needs to perform.**

1. Read input line from ASM file.
2. Parse the opcode.
3. Based on the opcode, ASM parser knows the next word. At this point it has 8 bits which needs to be translated into the instruction.
4. 8 bits is written to a binary file as two character hex number.
5. Repeat from step one until all instructions are processed.

**1.3.1 Indicate which of the following terms apply to which of the following languages:**

- |                |                     |                      |         |           |
|----------------|---------------------|----------------------|---------|-----------|
| a) imperative  | d) object-oriented  | g) fourth-generation |         |           |
| b) declarative | e) functional       | h) scripting         |         |           |
| c) von Neumann | f) third-generation |                      |         |           |
| 1) C           | 3) Cobol            | 5) Java              | 7) ML   | 9) Python |
| 2) C++         | 4) Fortran          | 6) Lisp              | 8) Perl | 10) VB    |

Scripting: Python, Perl

Declarative: ML

Functional: ML

Imperative: C, Java, Fortran

Object-oriented: C++, Java, VB

Von-Neumann: C, Fortran

Third-generation: Fortran, Cobol, Lisp, C, C++, Java

**1.6.1 For the block-structured C code of Fig. 1.13(a), indicate the values assigned to w, x, y, and z.**

w = 13; x = 11; y = 13; z = 11;

**1.6.2 Repeat Exercise 1.6.1 for the code of Fig. 1.13(b).**

w = 9; x = 7; y = 13; z = 11;

**1.6.3** For the block-structured code of Fig. 1.14, assuming the usual static scoping of declarations, give the scope for each of the twelve declarations.

$w_1$ : B1 — B3

$x_2$ : B2 — B3

$w_4$ : B4 — B5

$x_1$ : B1 — B2

$z_2$ : B2 — B3

$x_4$ : B4 — B5

$y_1$ : B1 — B5

$w_3$ : B3

$y_5$ : B5

$z_1$ : B1 — B2

$x_3$ : B3

$z_5$ : B5

**1.6.4** What is printed by the following C code?

3 2

### 2.2.1 Consider the context-free grammar:

$$S \rightarrow S S + \mid S S * \mid \mathbf{a}$$

- a) Show how the string **aa+a\*** can be generated by this grammar.
- b) Construct a tree for this string.
- c) What language does this grammar generate? Justify your answer.

(a)

- 1.  $S \rightarrow S S *$
- 2.  $S \rightarrow S S + a *$
- 3.  $S \rightarrow a a + a *$

(b)