F.1 Chapter 1 Solutions

1.1 Every computer can do the same thing as every other computer. A smaller or slower computer will just take longer.

1.2 No.

1.3 It is hard to increase the accuracy of analog machines.

1.4 Ambiguity.

1.5 (a) inputs to first (x) box are a and x
output of first (x) box is ax
inputs to second (+) box are ax and b
output of second (+) box is $ax + b$

(b) inputs to first (+) box are w and x
output of first (+) box is $w + x$
inputs to second (+) box are y and z
output of second (+) box is $y + z$
inputs to third (+) box are $(w + x)$ and $(y + z)$
output of third (+) box is $w + x + y + z$
inputs to fourth (x) box are $(w + x + y + z)$ and .25
output of fourth (x) box is $0.25(w + x + y + z)$, which is the average

(c) The key is to factor $a^2 + 2ab + b^2 = (a + b)^2$
inputs to first (+) box are a and b
output of first (+) box is $a + b$
inputs to second (x) box are $(a + b)$ and $(a + b)$
output of second (x) box is $(a + b)^2 = a^2 + 2ab + b^2$

1.6 Any ambiguous statement is fine. For example: I ate my sandwich on a bed of lettuce. The sandwich might have been sitting on a bed of lettuce on the plate, or I might have been sitting on a bed of lettuce eating a sandwich.

1.7 If the taxi driver is honorable, he/she asks you whether time or money is more important to you, and then gets you to the airport as quickly or as cheaply as possible. You are freed from knowing anything about the various ways one can get to the airport. If the taxi driver is dishonorable, you get to the airport late enough to miss your flight and/or at a taxi fare far in excess of what it should have been, as the taxi driver takes a very circuitous route.

1.8 He could mean a lot of things. This statement is ambiguous as it could mean different things.

Some reasonable interpretations are: a) John saw the man in "the park with a telescope" b) John saw the "man in the park" with a telescope.

As this statement is ambiguous, it is unacceptable as a statement in a program.

1.9 Yes, if phrased in a way that is definite and lacks ambiguity.

1.10 Definiteness: each step is precisely stated.

Effective Computability: each step can be carried out by a computer.

Finiteness: the procedure terminates.

1.11 (a) Lacks definiteness: Go south on Main St. for a mile or so.

(b) Lacks effective computability: Find the integer that is the square root of 14.

(c) Lacks finiteness: Do something. Repeat forever.

1.12 (a) Lacks definiteness, since it does not specify how two rows are to be added. Also, the 3rd or the 4th row could be added to the first row. So there are two posible answers.

(b) This is not effectively computable, because there is no end to the number line. Anything involving infinity must not be effectively computable. This is also not finite, for the same reason.

(c) This is an algorithm.

(d) This is not finite, so it is not an algorithm. If, as Calvin suspects, the coin is weighted, they will be flipping that coin forever.

(e) This is not finite, so it is not an algorithm. Steps 1 to 6 calculate, albeit in a long way, the number - 1. If the given number is negative or zero, then there will never be a time when you get 0 at the end of step 6.

1.13 Both computers, A and B, are capable of solving the same problems. Computer B can perform subtraction by taking the negative of the second number and adding it to the first one. As A and B are otherwise identical, they are capable of solving the same problems.

1.14 (a) 120 transformation processes.

(b) Any 3 of this form are fine: "Sort Algorithm 3, Fortran program, SPARC ISA, SPARC microarchitecture 1".

(c) 120 again.

1.15 Advantages of a higher level language: Fewer instructions are required to do the same amount of work. This usually means it takes less time for a programmer to write a program to solve a problem. High level language programs are generally easier to read and therefore know what is going on. Disadvantages of a higher level language: Each instruction has less control over the underlying hardware that actually performs the computation that the program frequently executes less fficiently.
NOTE: this problem is beyond the scope of Chapter 1 or most students.

1.16 Possible operations, data types, addressing modes.

1.17 An ISA describes the interface to the computer from the perspective of the 0s and 1s of the program. For example, it describes the operations, data types, and addressing modes a programmer can use on that particular computer. It doesn't specify the actual physical implementation. The microarchitecture does that. Using the car analogy, the ISA is what the driver sees, and the microarchitecture is what goes on under the hood.

1.18 A single microarchitecture typically implements only one ISA. However, many microarchitectures usually exist for the same ISA.

1.19 (a) Problem: For example, what is the sum of the ten smallest positive integers.

(b) Algorithm: Any procedure is fine as long as it has definiteness, effective computability, and finiteness.

(c) Language: For example, C, C++, Fortran, IA-32 Assembly Language.

(d) ISA: For example, IA-32, PowerPC, Alpha, SPARC.

(e) Microarchitecture: For example, Pentium III, Compaq 21064.

(f) Circuits: For example, a circuit to add two numbers together.

(g) Devices: For example, CMOS, NMOS, gallium arsenide.

1.20 Refering to the levels of transformation as the levels of abstraction is a reasonable characterization. Each level in Figure 1.6 is essentially a level of abstraction, abstracting the other levels. For example, if the problem statement said "Find the average of two numbers", you have abstracted the rest of the system away. Now, lets take the Language level. If you have a C language program, the lower levels are abstraced away. You dont have to worry about the exact ISA or microarchitecture you will run the programon. Similarily, you should be able to draw examples for all the other levels.

1.21 It is in the ISA of the computer that will run it. We know this because if the word procesing software were in a high- or low-level programming language, then the user would need to compile it or assemble it before using it. This never happens. The user just needs to copy the files to run the program, so it must already be in the correct machine language, or ISA.

1.22 The transformation from Problems to Algorithms is the most difficult step. There is ambiguity in a Problem statement which needs to be resolved in order to generate an algorithm. This requires the intelligence to actually understand the problem and make sense out of it. All the other transformations can be performed by a program written to perform that transformation.

1.23 ISA's don't change much between successive generations, because of the need for backward compatibility. You'd like your new computer to still run all your old software.