Chapter 1: Review Exercise Solutions

R1.1

A well-designed computer program is easy to use without any special knowledge. For example, most people can learn to navigate webpages with only a few minutes of practice. On the other hand, programming a computer requires special knowledge about what the computer can fundamentally do and how you would communicate with it through a programming language.

R1.2

Typically program code is stored on a hard disk, CD/DVD disc, or in some other central location across a network. User data is often more likely to be stored on a local hard disk, although it can also be stored on a network or even a CD/DVD for backup storage.

R1.3

The monitor, speakers, and printer serve as the primary devices to give information to the user. The keyboard and mouse are the primary devices that take user input.

R1.4

It's very likely your cell phone is a programmable computer. If you can take pictures, send email/text messages, and/or surf the web with your phone, it is a programmable computer. If your cell phone can only send and receive phone calls, it is probably a single-function device.

R1.5

One advantage of Java over machine code is that Java statements are independent of the machine (computer) they are being executed on; machine code statements differ from one type of machine to the next. Another advantage of Java is that it is much more readable and understandable (by humans) than machine code.

R1.6

a) Solutions here will vary based on user and IDE preference. On a UNIX-based system using the Eclipse IDE you may see a path like

```
/home/nancy/JFE/src
```

While on a Microsoft Windows machine you might find a directory like:
b) Again, solutions can vary. On Unix using Eclipse you might see:

/home/nancy/JFE/bin

A Microsoft Windows machine might be:

C:\Users\nancy\Documents\JFE\bin

c) The answer to this question is dependent on the type of operating system and version of Java. On a Unix based system using Java 1.6 you might find rt.jar here:

/usr/lib/jvm/java-6-sun-1.6.0.13/jre/lib/rt.jar

While on a Microsoft Windows platform you might find it here:

C:\Program Files\Java\jdk1.6.0_10\jre

R1.7

The program prints the following:

39 + 3
42

Java interprets the statement "39 + 3" as a string and thus prints out the literal characters 39 + 3. Java interprets the second statement 39 + 3 as an operation between two numbers, so it first calculates the value 39 + 3 = 42 then prints out the result 42.

R1.8

HelloWorld

Because there are no spaces after the System.out.print("Hello"); the next line prints World directly after Hello is printed.

R1.9

Java interprets the comma in the println method to mean that two strings are passed to println. It’s likely the programmer meant to do this:

System.out.print("Hello, World!");
R1.10

This version omits a semicolon after the `println` statement and omits a curly brace to close the class:

```java
public class HelloPrinter1
{
    public static void main(String[] args)
    {
        System.out.println("Hello, World!");
    }
}
```

This version omits quotes around the `Hello, World!` statement and adds a semicolon after the `main` declaration:

```java
public class HelloPrinter2
{
    public static void main(String[] args);
    {
        System.out.println(Hello, World!);
    }
}
```

This version omits parenthesis around the `println` argument and omits the `static` reserved word in the `main` declaration:

```java
public class HelloPrinter3
{
    public void main(String[] args)
    {
        System.out.println "Hello, World!";
    }
}
```

This version has a subtle logic error in the sense that it doesn’t print a newline character after `Hello, World!` is printed:

```java
public class HelloPrinter4
{
    public static void main(String[] args)
    {
        System.out.print("Hello, World!");
    }
}
```

R1.11

Syntax errors are discovered by compiling your Java program; the Java compiler will report them directly. Logic errors are discovered by testing your program by running it and verifying the correct output; if the output is incorrect, you have a logic error.
R1.12

balance = $10,000
total months = 0
while balance is greater than $0
    increase balance by 0.5% of its value
    decrease balance by $500
    add 1 to the total number of months
years to deplete = total months / 12

R1.13

Yes, any interest rate that causes more money to be added each month than was taken out would prevent the solution to R1.12 from terminating. We can guarantee that our algorithm terminates by checking for this condition before the calculation starts:

balance = $10,000
monthly interest rate = 0.5%
monthly withdrawal amount = $500
if balance * monthly interest rate > monthly withdrawal amount
    report that the balance will never reach zero
else
    total months = 0
    while balance is greater than $0
        increase balance by 0.5% of its value
        decrease balance by $500
        add 1 to the total number of months
    years to deplete = total months / 12

R1.14

// Inputs are width, length, height, number of windows, number of doors, // window width, window height, door width, door height

surface area of side 1 = width * height
surface area of side 2 = length * height
surface area of doors = door width * door height * number of doors
surface area of windows = window width * window height * number of windows
surface area without doors or windows = (surface area of side 1) * 2 + (surface area of side 2) * 2
// This is what we want to know:
total surface area = surface area without doors or windows -
(surface area of doors + surface area of windows)

R1.15

// Inputs are one way distance, fuel efficiency, train ticket price

cost of gallon of gas = 4 // $4 per gallon
car maintenance = .05 // $0.05 per mile
cost of car = one way distance / fuel efficiency * cost of gallon of gas +
one way distance * car maintenance
if cost of car < train ticket price
  choose to drive car.
else
  choose to take train.

R1.16

// Inputs are one way distance, beginning mileage, ending mileage, number of work days
// Question asks only for fraction "commuting to work" thus we don't need to count the fraction commuting back home

total miles traveled = ending mileage - beginning mileage
miles traveled to work = one way distance * number of work days

// This is what we want to know:
fraction of miles for commute = miles traveled to work / total miles traveled

// Personal use is whatever is left over
fraction of miles for personal use = 1 - fraction of miles for commute

R1.17

By itself, any program you write in the computer has no basis to make any assumptions unless data is provided to base it on. For example, there’s no built in data that can tell a computer what the price of gas is, let alone how much you personally drive. Thus, we supplied these data to the program; without them the program would have no basis to calculate which car was a better deal.
The pseudocode below is based on the following facts.

1) We can calculate \( \pi \) directly from the formula given if we first multiply both sides by 4.
2) We can determine when we have achieved 6 digits of accuracy by comparing the difference between the current and previous calculation of \( \pi \). If that difference is less than 0.00001 we know we have achieved the 6 digits of accuracy (3.14159).

\[
\begin{align*}
\pi \text{ estimate} &= 4 \\
\pi \text{ old estimate} &= 4 \\
current \text{ denominator} &= 3 \\
term \text{ negative} &= \text{true} \\
error &= 1 \quad \text{// Initialize the error to something big}
\end{align*}
\]

// Keep calculating while the difference is greater than the 6 digits of accuracy required
while error > 0.00001

\[
\begin{align*}
\pi \text{ old estimate} &= \pi \text{ estimate} \\
current \text{ term} &= 4 \div current \text{ denominator} \\
\text{if term negative}
\end{align*}
\]

\[
\begin{align*}
\pi \text{ estimate} &= \pi \text{ estimate} - current \text{ term} \\
\text{else}
\end{align*}
\]

\[
\begin{align*}
\pi \text{ estimate} &= \pi \text{ estimate} + current \text{ term} \\
\text{flip the value of term negative} \\
\text{add 2 to current denominator} \\
\text{error} &= \text{absolute value of the difference between } \pi \text{ estimate and } \pi \text{ old estimate}
\end{align*}
\]

R1.19

Every day at 5 P.M. please do the following:

1) Insert the USB memory stick into the computer.
2) Create a new directory on the USB memory stick entitled “BACKUP – DATE” where “DATE” is replaced with the current date.
3) Copy the files from the “My Documents” folder on the computer to the folder you just created on the USB stick.
4) Double check to make sure the new directory contains the files you copied.
   a. If the folder is empty, something is wrong. It’s possible you backed up to the wrong directory. Try it again and be careful which directory you copy into.
5) Once you verified the copy is complete, remove the USB stick and store it someplace safe.
6) Thank you!

R1.20

// Calculate tip and total amount
tip = bill * 0.15
total = bill + tip

// Split the total bill

splitAmount = total / numPeople
splitBill = bill / numPeople
splitTip = tip / numPeople
Print bill, tip, total.
Print "Amount per person =" + splitAmount.
Print "Bill and tip per person =" + splitBill + " " + splitAmount.