**Chapter 3**

**Exercise 1**

3.10 Compare and contrast the if single-selection statement and the while repetition statement. How are these two statements similar? How are they different?

Similarities: Both are control flow statements in Java and use conditions to determine execution.

Differences: The if statement executes a block of code once if the condition is true, while the while loop executes repeatedly as long as the condition remains true.

3.11 Explain what happens when a Java program attempts to divide one integer by another. What happens to the fractional part of the calculation? How can you avoid that outcome?

Java performs integer division if both operands are integers, discarding the fractional part.

Use floating-point division by converting one operand to double.

3.12 Describe the two ways in which control statements can be combined.

Sequential execution – Statements execute in the order they appear.

Nesting – One control statement is placed inside another (e.g., an if inside a while loop).

3.13 What type of repetition would be appropriate for calculating the sum of the first 100 positive integers? What type would be appropriate for calculating the sum of an arbitrary number of positive integers? Briefly describe how each of these tasks could be performed.

Sum of first 100 integers: Use a for loop since the count is known.

Sum of an arbitrary number of integers: Use a while loop with user input to stop when needed.

3.14 What is the difference between preincrementing and postincrementing a variable?

In Preincrement (++x), it increments x first, then uses the new value while Postincrement (x++) uses x first, then increments it.

3.15 Identify and correct the errors in each of the following pieces of code. [Note: There may be more than one error in each piece of code.]

a) if (age >= 65);

System.out.println("Age is greater than or equal to 65");

else System.out.println("Age is less than 65)";

Answer:

if (age >= 65) {

System.out.println("Age is greater than or equal to 65");

} else {

System.out.println("Age is less than 65");

}

b) int x = 1, total;

while (x <= 10) {

total += x; ++x;

}

Answer:

int x = 1, total = 0;

while (x <= 10) {

total += x;

++x;

}

c) while (x <= 100)

total += x; ++x;

Answer:

int x = 1;

while (x <= 100) {

total += x;

++x;

}

d)

while (y > 0) {

System.out.println(y);

++y;

Answer:

while (y > 0) {

System.out.println(y);

    --y;

**Exercise 2**

**3.17 (Gas Mileage**)

Drivers are concerned with the mileage their automobiles get. One driver has kept track of several trips by recording the miles driven and gallons used for each tankful. Develop a Java application that will input the miles driven and gallons used (both as integers) for each trip. The program should calculate and display the miles per gallon obtained for each trip and print the combined miles per gallon obtained for all trips up to this point. All averaging calculations should produce floating-point results. Use class Scanner and sentinel-controlled repetition to obtain the data from the user.

1. Uses Scanner to get miles driven and gallons used.

2. Calculates miles per gallon (mpg = miles / gallons).

3. Uses a sentinel-controlled loop to allow multiple entries.

4. Displays MPG for each trip and combined MPG.

**3.18 (Credit Limit Calculator)**

Develop a Java application that determines whether any of several department-store customers has exceeded the credit limit on a charge account. For each customer, the following facts are available:

a) account number

b) balance at the beginning of the month

c) total of all items charged by the customer this month

d) total of all credits applied to the customer’s account this month

e) allowed credit limit.

The program should input all these facts as integers, calculate the new balance (= beginning balance + charges – credits), display the new balance and determine whether the new balance exceeds the customer’s credit limit. For those customers whose credit limit is exceeded, the program should display the message "Credit limit exceeded".

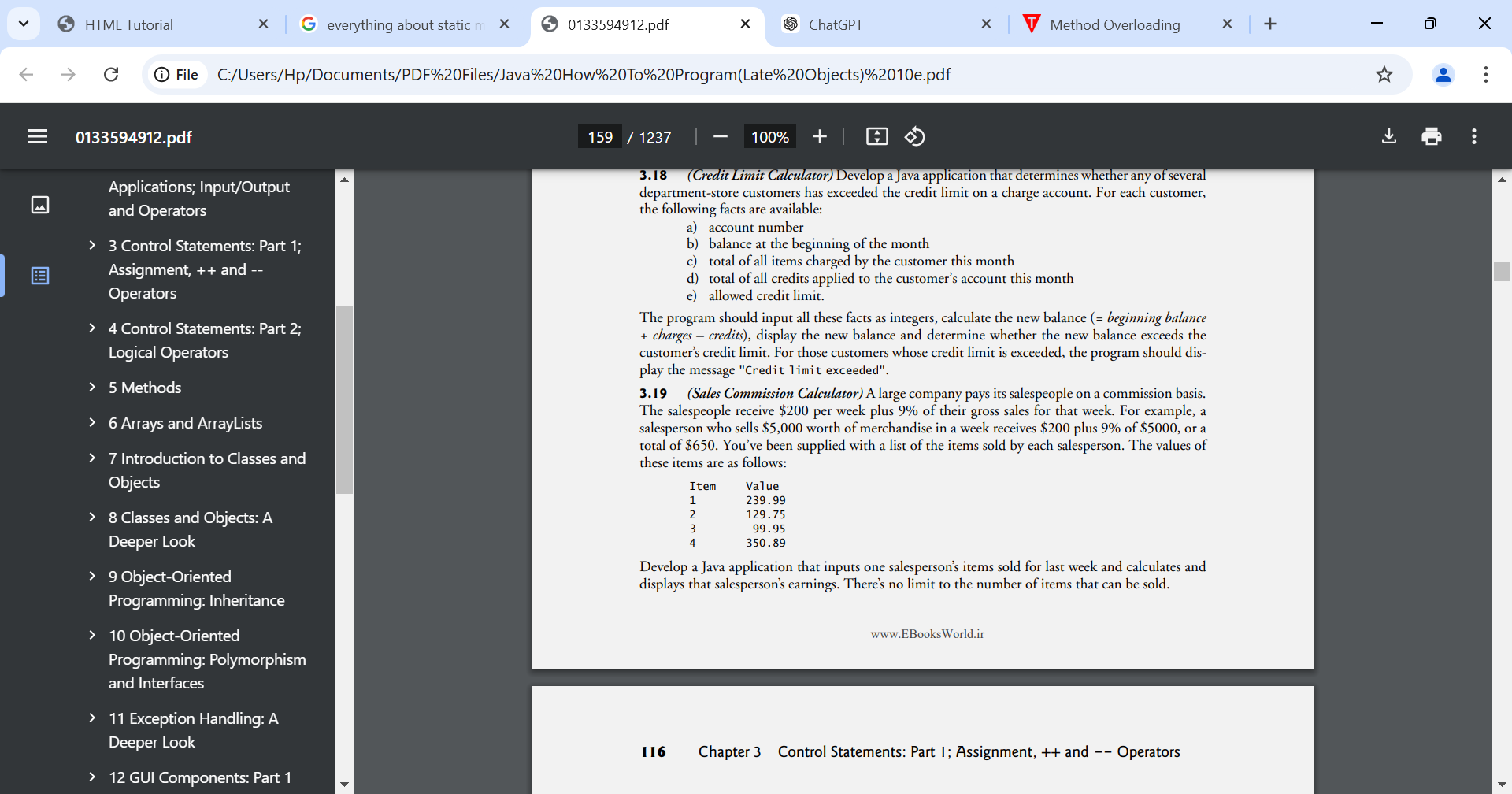
Inputs account details.

Calculates new balance: new\_balance = beginning\_balance + charges - credits.

If new\_balance > credit\_limit, displays "Credit limit exceeded".

**3.19 (Sales Commission Calculator)**

A large company pays its salespeople on a commission basis. The salespeople receive $200 per week plus 9% of their gross sales for that week. For example, a salesperson who sells $5,000 worth of merchandise in a week receives $200 plus 9% of $5000, or a total of $650. You’ve been supplied with a list of the items sold by each salesperson. The values of these items are as follows:



Develop a Java application that inputs one salesperson’s items sold for last week and calculates and displays that salesperson’s earnings. There’s no limit to the number of items that can be sold.

Inputs weekly sales.

Calculates salary: 200 + (0.09 \* sales).

Outputs total earnings.

**3.20 (Salary Calculator)**

Develop a Java application that determines the gross pay for each of three employees. The company pays straight time for the first 40 hours worked by each employee and time and a half for all hours worked in excess of 40. You’re given a list of the employees, their number of hours worked last week and their hourly rates. Your program should input this information for each employee, then determine and display the employee’s gross pay. Use class Scanner to input the data.

Inputs hours worked and hourly rate.

If hours ≤ 40, normal pay.

If hours > 40, overtime (1.5x rate).

**3.21 (Find the Largest Number)**

The process of finding the largest value is used frequently in computer applications. For example, a program that determines the winner of a sales contest would input the number of units sold by each salesperson. The salesperson who sells the most units wins the contest. Write a pseudocode program, then a Java application that inputs a series of 10 integers and determines and prints the largest integer. Your program should use at least the following three variables:

a) counter: A counter to count to 10 (i.e., to keep track of how many numbers have been input and to determine when all 10 numbers have been processed).

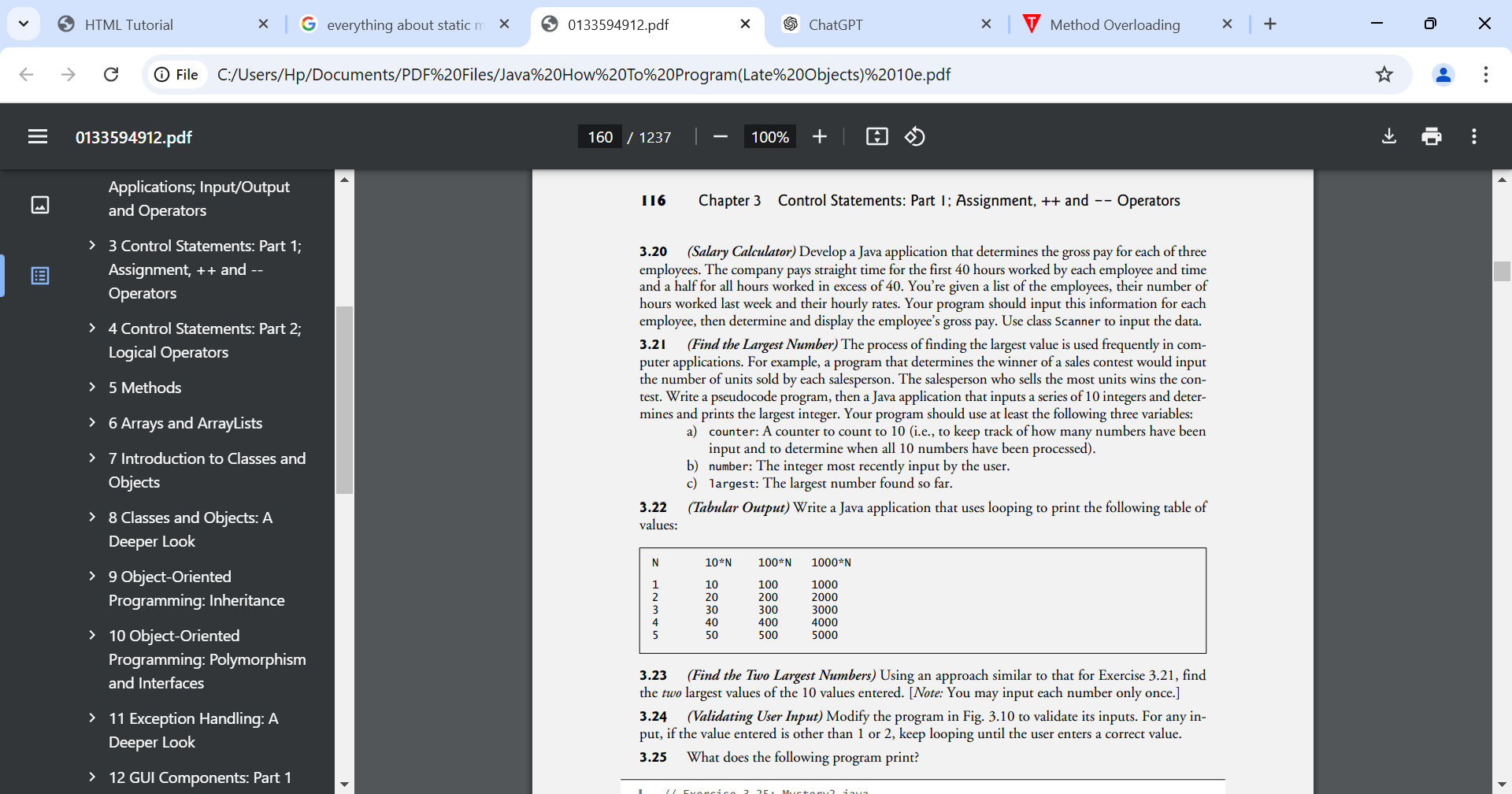
b) number: The integer most recently input by the user.

c) largest: The largest number found so far.

Inputs 10 numbers.

Uses a loop to track the largest number.

3.22 (Tabular Output) Write a Java application that uses looping to print the following table of values:



Uses a loop to print a formatted table.

3.23 (Find the Two Largest Numbers) Using an approach similar to that for Exercise 3.21, find the two largest values of the 10 values entered. [Note: You may input each number only once.]

Similar to 3.21, but tracks two highest numbers.

3.24 (Validating User Input) Modify the program in Fig. 3.10 to validate its inputs. For any input, if the value entered is other than 1 or 2, keep looping until the user enters a correct value.

Modifies the program to keep prompting until valid input (1 or 2) is entered.

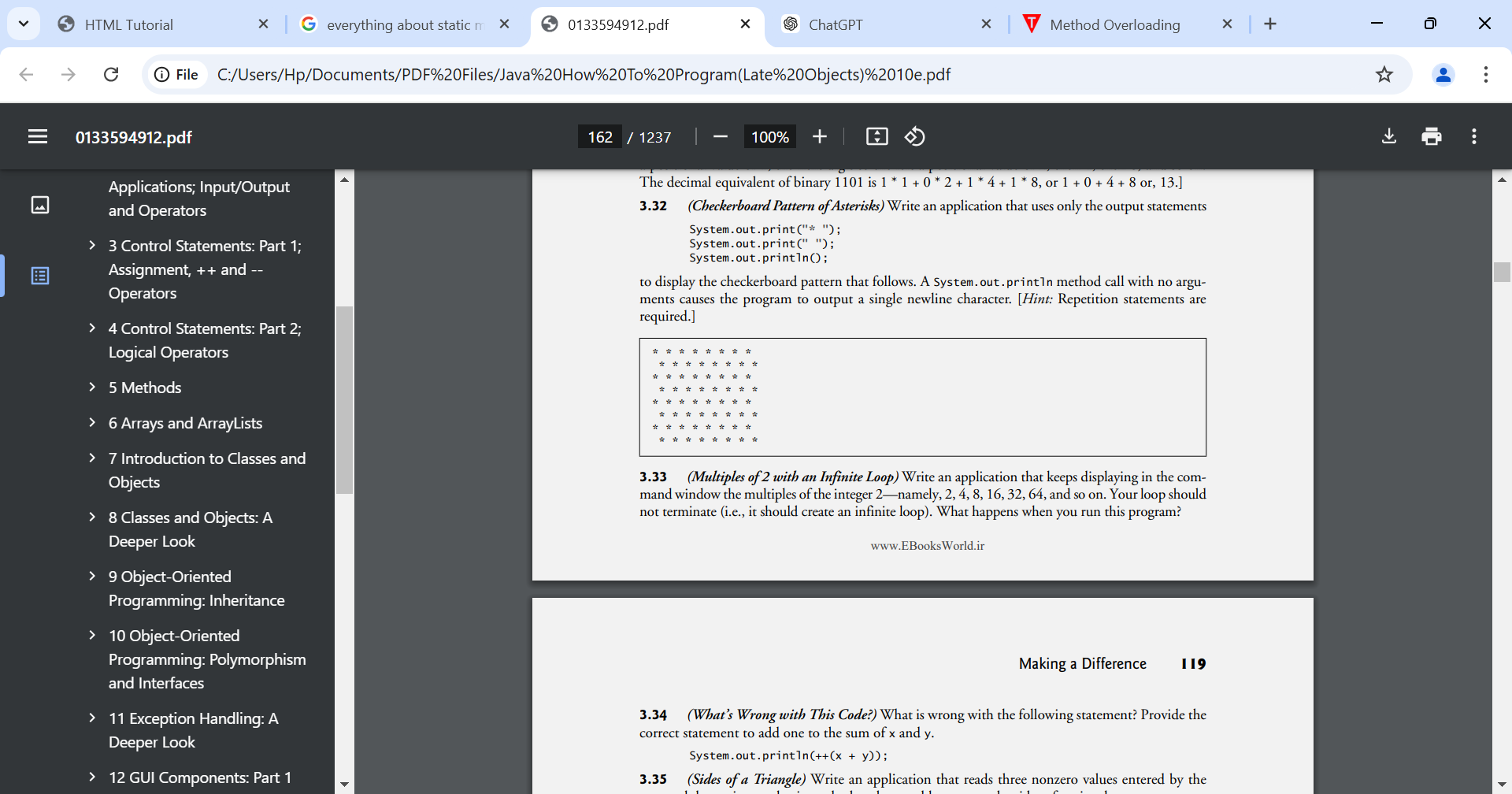
**3.25 (Checkerboard Pattern of Asterisks)**

Write an application that uses only the output statements

System.out.print("\* ");

System.out.print(" ");

System.out.println(); to display the checkerboard pattern that follows. A System.out.println method call with no arguments causes the program to output a single newline character. [Hint: Repetition statements are required.]



Uses nested loops:

for (int i = 0; i < 8; i++) {

for (int j = 0; j < 8; j++) {

System.out.print((i % 2 == j % 2) ? "\* " : " ");

}

System.out.println();

}

**Making a Difference**

**3.38 (Enforcing Privacy with Cryptography)**

The explosive growth of Internet communications and data storage on Internet-connected computers has greatly increased privacy concerns. The field of cryptography is concerned with coding data to make it difficult (and hopefully—with the most advanced schemes—impossible) for unauthorized users to read. In this exercise you’ll investigate a simple scheme for encrypting and decrypting data. A company that wants to send data over the Internet has asked you to write a program that will encrypt it so that it may be transmitted more securely. All the data is transmitted as four-digit integers. Your application should read a four-digit integer entered by the user and encrypt it as follows: Replace each digit with the result of adding 7 to the digit and getting the remainder after dividing the new value by 10. Then swap the first digit with the third, and swap the second digit with the fourth. Then print the encrypted integer. Write a separate application that inputs an encrypted four-digit integer and decrypts it (by reversing the encryption scheme) to form the original number. [Optional reading project: Research “public key cryptography” in general and the PGP (Pretty Good Privacy) specific public key scheme. You may also want to investigate the RSA scheme, which is widely used in industrial-strength applications.]

1. Adds 7 to each digit, then mod 10.

2. Swaps 1st ↔ 3rd and 2nd ↔ 4th.

Decrypts by reversing steps.

**3.39 (World Population Growth)**

World population has grown considerably over the centuries. Continued growth could eventually challenge the limits of breathable air, drinkable water, arable cropland and other limited resources. There’s evidence that growth has been slowing in recent years and that world population could peak sometime this century, then start to decline. For this exercise, research world population growth issues online. Be sure to investigate various viewpoints. Get estimates for the current world population and its growth rate (the percentage by which it’s likely to increase this year). Write a program that calculates world population growth each year for the next 75 years, using the simplifying assumption that the current growth rate will stay Print the results in a table. The first column should display the year from year 1 to year 75. The second column should display the anticipated world population at the end of that year. The third column should display the numerical increase in the world population that would occur that year. Using your results, determine the year in which the population would be double what it is today, if this year’s growth rate were to persist.

Gets current population and growth rate.

Uses a loop to project population for 75 years.

Prints year-wise growth rate and determines doubling year.