Programming Languages Homework 2

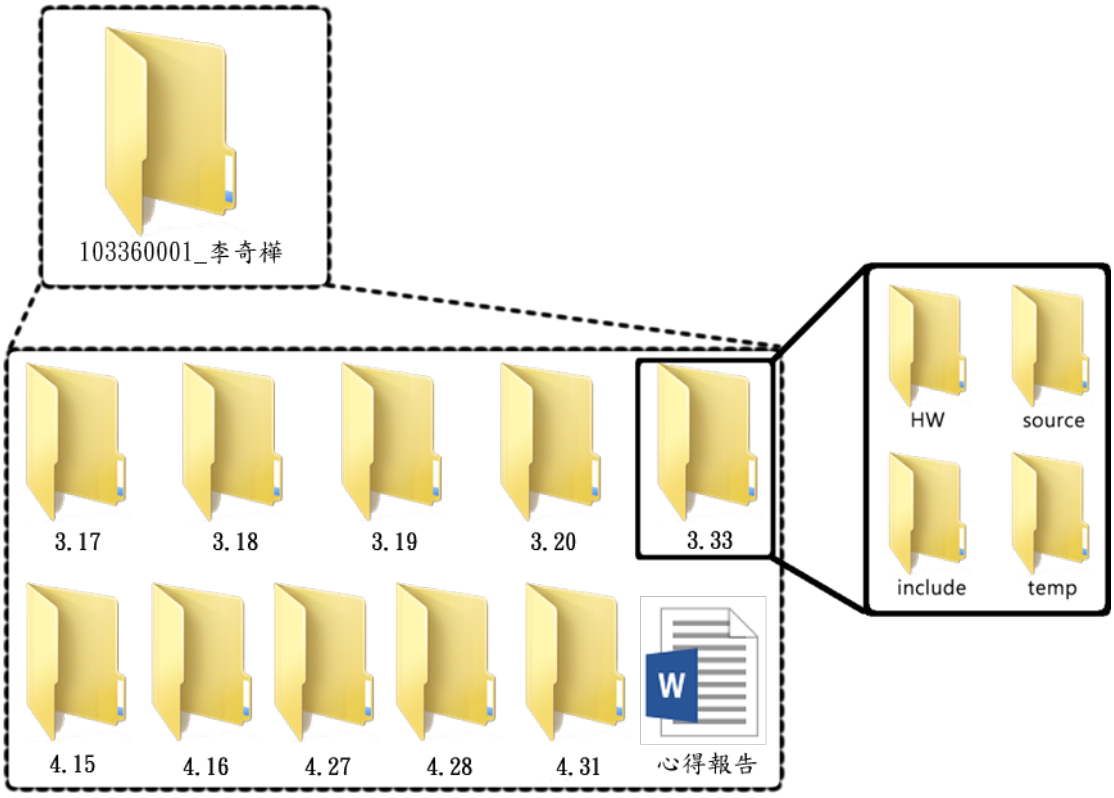
功課習題：

1. 3.17
2. 3.18
3. 3.19
4. 3.20
5. 3.33
6. 4.15
7. 4.16
8. 4.27
9. 4.28
10. 4.31

繳交期限：10/16 (一) 晚上 11:59 前

繳交格式：103360001_李奇樺.zip

繳交內容：**心得報告(包含:上傳到 GitHub 的截圖)**和 **Lab 所檢查之程式檔案**，如下圖



上傳位置：Homework\Upload

帳號、密碼：CC

如無法上傳可 Mail 繳交的作業檔案(zip)至

吳佩儒 peggyplwu1109@gmail.com

林裕軒 shane.sandy123@gmail.com

3.17：

3.17 *(Credit Limit Calculator)* Develop a C program that will determine if a department store customer 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 this customer this month
d) Total of all credits applied to this customer's account this month
e) Allowed credit limit

The program should input each fact, calculate the new balance (= *beginning balance* + *charges* – *credits*), 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 customer's account number, credit limit, new balance and the message "Credit limit exceeded." Here is a sample input/output dialog:

```
Enter account number (-1 to end): 100
Enter beginning balance: 5394.78
Enter total charges: 1000.00
Enter total credits: 500.00
Enter credit limit: 5500.00
Account:       100
Credit limit: 5500.00
Balance:      5894.78
Credit Limit Exceeded.

Enter account number (-1 to end): 200
Enter beginning balance: 1000.00
Enter total charges: 123.45
Enter total credits: 321.00
Enter credit limit: 1500.00

Enter account number (-1 to end): 300
Enter beginning balance: 500.00
Enter total charges: 274.73
Enter total credits: 100.00
Enter credit limit: 800.00

Enter account number (-1 to end): -1
```

3.18：

3.18 *(Sales Commission Calculator)* One large chemical 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 $5000 worth of chemicals in a week receives $200 plus 9% of $5000, or a total of $650. Develop a program that will input each salesperson's gross sales for last week and will calculate and display that salesperson's earnings. Process one salesperson's figures at a time. Here is a sample input/output dialog:

```
Enter sales in dollars (-1 to end): 5000.00
Salary is: $650.00

Enter sales in dollars (-1 to end): 1234.56
Salary is: $311.11

Enter sales in dollars (-1 to end): -1
```

3.19 至 3.20：

3.19 *(Interest Calculator)* The simple interest on a loan is calculated by the formula

```
interest = principal * rate * days / 365;
```

The preceding formula assumes that `rate` is the annual interest rate, and therefore includes the division by 365 (days). Develop a program that will input `principal`, `rate` and `days` for several loans, and will calculate and display the simple interest for each loan, using the preceding formula. Here is a sample input/output dialog:

```
Enter loan principal (-1 to end): 1000.00
Enter interest rate: .1
Enter term of the loan in days: 365
The interest charge is $100.00

Enter loan principal (-1 to end): 1000.00
Enter interest rate: .08375
Enter term of the loan in days: 224
The interest charge is $51.40

Enter loan principal (-1 to end): -1
```

3.20 *(Salary Calculator)* Develop a program that will determine the gross pay for each of several employees. The company pays "straight time" for the first 40 hours worked by each employee and pays "time-and-a-half" for all hours worked in excess of 40 hours. You're given a list of the employees of the company, the number of hours each employee worked last week and the hourly rate of each employee. Your program should input this information for each employee and should determine and display the employee's gross pay. Here is a sample input/output dialog:

```
Enter # of hours worked (-1 to end): 39
Enter hourly rate of the worker ($00.00): 10.00
Salary is $390.00

Enter # of hours worked (-1 to end): 40
Enter hourly rate of the worker ($00.00): 10.00
Salary is $400.00

Enter # of hours worked (-1 to end): 41
Enter hourly rate of the worker ($00.00): 10.00
```

3.33：

3.33 *(Hollow Rectangle of Plus Symbols)* Modify the program you wrote in Exercise 3.32 so that it prints a hollow rectangle. For example, if your program reads a length of 3 and a breadth of 12, it should print

```
++++++++++++
+          +
++++++++++++
```

4.15、16：

4.15 *(Modified Compound-Interest Program)* Modify the compound-interest program of Section 4.6 to repeat its steps for an investment of $5000, for 15 years, and for interest rates of 10.0%, 10.5%, 11.0%, 11.5%, and 12.0%. Use a `for` loop to vary the interest rate.

4.16 *(Triangle-Printing Program)* Write a program that prints the following patterns separately, one below the other. Use `for` loops to generate the patterns. All asterisks (`*`) should be printed by a single `printf` statement of the form `printf( "%s", "*" );` (this causes the asterisks to print side by side). [*Hint:* The last two patterns require that each line begin with an appropriate number of blanks.]

```
(A)         (B)         (C)         (D)
*           **********  **********          *
**          *********   *********          **
***         ********    ********          ***
****        *******     *******          ****
*****       ******      ******          *****
******      *****       *****          ******
*******     ****        ****          *******
********    ***         ***          ********
*********   **          **          *********
**********  *           *          **********
```

4.27、28：

4.27 *(Pythagorean Triples)* A right triangle can have sides that are all integers. The set of three integer values for the sides of a right triangle is called a Pythagorean triple. These three sides must satisfy the relationship that the sum of the squares of two of the sides is equal to the square of the hypotenuse. Find all Pythagorean triples for side1, side2, and the hypotenuse all no larger than 500. Use a triple-nested `for` loop that simply tries all possibilities. This is an example of "brute-force" computing. It's not aesthetically pleasing to many people. But there are many reasons why these techniques are important. First, with computing power increasing at such a phenomenal pace, solutions that would have taken years or even centuries of computer time to produce with the technology of just a few years ago can now be produced in hours, minutes or even seconds. Recent microprocessor chips can process a billion instructions per second! Second, as you'll learn in more advanced computer science courses, there are large numbers of interesting problems for which there's no known algorithmic approach other than sheer brute force. We investigate many kinds of problem-solving methodologies in this book. We'll consider many brute-force approaches to various interesting problems.

4.28 *(Calculating Weekly Pay)* A company pays its employees as managers (who receive a fixed weekly salary), hourly workers (who receive a fixed hourly wage for up to the first 40 hours they work and "time-and-a-half"—i.e., 1.5 times their hourly wage—for overtime hours worked), commission workers (who receive $250 plus 5.7% of their gross weekly sales), or pieceworkers (who receive a fixed amount of money for each of the items they produce—each pieceworker in this company works on only one type of item). Write a program to compute the weekly pay for each employee. You do not know the number of employees in advance. Each type of employee has its own pay code: Managers have paycode 1, hourly workers have code 2, commission workers have code 3 and pieceworkers have code 4. Use a `switch` to compute each employee's pay based on that employee's paycode. Within the `switch`, prompt the user (i.e., the payroll clerk) to enter the appropriate facts your program needs to calculate each employee's pay based on that employee's paycode. [*Note:* You can input values of type `double` using the conversion specifier `%lf` with `scanf`.]

4.31：

4.31 *(Diamond-Printing Program)* Write a program that prints the following diamond shape. You may use `printf` statements that print either a single asterisk (`*`) or a single blank. Maximize your use of repetition (with nested `for` statements) and minimize the number of `printf` statements.