NATIONAL UNIVERSITY OF SINGAPORE
SCHOOL OF COMPUTING

Practical Examination 1 (PE1) for Semester 1, AY2014/5
CS1010 Programming Methodology

20 September 2014 Time Allowed: 2 hours

INSTRUCTION TO CANDIDATES

1. You are only allowed to read this cover page and the last page. Do **not** read the questions until you are told to do so.
2. This paper consists of **2** exercises on 6 pages. Each exercise constitutes 50%.
3. This is an open-book exam. You may bring in any printed material, but **not** electronic devices, including but not limited to laptop, thumb-drive, electronic dictionary and calculator. You are to switch off/silence your mobile phone and keep it out of view.
4. You may turn to the last page (page 6) to read some advice.
5. You will be logged into a special Windows account at the beginning of the PE. Do not log off until the end of the PE. Do not use your own NUSNET account.
6. A plab account slip will be issued to you at the beginning of the PE. Bring your matriculation card for identification when you collect it. Please leave your matriculation card on the desk in front of you throughout the PE.
7. You are to write your program in the given **plab account**. The host name is **plab4** (not sunfire!). No activity should be done outside this plab account.
8. You **do not** need to submit your programs to CodeCrunch. We will retrieve your programs and submit them to CodeCrunch after the PE.
9. Skeleton programs and some test data files are already residing in your plab account. Please leave the programs in the home directory, and use the same program names as specified in the paper. Do **not** create subdirectory to put your programs there or we will not be able to find them!
10. **Only your source codes (.c programs)** from your plab account will be collected after the PE. Hence, how you name your executable files is not important.
11. Please read carefully and follow all instructions in the question. If in doubt, please ask. Raise your hand and the invigilator will attend to you.
12. Any form of communication with other students or the use of unauthorised materials is considered cheating and you are liable to disciplinary action.
13. Please save your programs regularly during the PE.
14. When you are told to stop, please do so **immediately**, or you will be penalised.
15. At the end of the PE, please **log out from your plab account**.
16. Please check and take your belongings (especially matriculation card) before you leave.
17. We will make arrangement for you to retrieve your programs after we have finished grading. Grading may take a week or more.

ALL THE BEST!

Exercise 1: Parking Fee [50 marks]

Problem Statement

Mr. Wu has been going to work every day by taxi for many years. However, the taxi fare has been increasing rather quickly in recent years. Therefore, he is considering driving to work instead.

One of the costs for driving is the parking fee. The parking rates of the car park at Mr. Wu's workplace are as shown in the table below.

	Weekday	Saturday	Sunday
4am ~ 7am	$2.00 per hour	$2.50 per hour	$5.00 per entry
7am ~ 6pm	$1.20 per 30 minutes	$1.50 per 30 minutes	
6pm ~ midnight	$5.00 per entry	$7.00 per entry	

Special note:

1. The car park opens at 4am and closes at midnight. All vehicles must leave by midnight.
2. There is a grace period of 10 minutes on any day (i.e., it is completely free to park for 10 minutes or less regardless of day and time.)
3. There is a 10% surcharge for parking more than 10 hours on a weekday and 20% for Saturday. There is no surcharge for Sunday.
4. There is an additional $3.00 fee for exiting after 10pm on any day. (Surcharge is not applicable on this fee.)

Here are a few examples on how the total fee is calculated:

Example 1: Tuesday, 4:29am to 7:50am.

- 4:29am to 7am is charged as 3 1-hour slots: $2.00 * 3 = $6.00
- 7am to 7:50am is charged as 2 30-minute slots: $1.20 * 2 = $2.40
- Total fee = $6.00 + $2.40 = $8.40

Example 2: Saturday, 7:01am to 7:49pm.

- 7:01am to 6pm is charged as 22 30-minute slots: $1.50 * 22 = $33.00
- 6pm to 7:49pm is charged as one entry: $7.00
- 20% Surcharge for parking more than 10 hours: ($33.00 + $7.00) * 20% = $8.00
- Total fee = $33.00 + $7.00 + $8.00 = $48.00

Example 3: Sunday, 3pm to 10:01pm.

- 3pm to 10:01pm is charged as one entry: $5.00
- Additional fee for exiting after 10pm: $3.00
- Total fee = $5.00 + $3.00 = $8.00

Example 4: Thursday, 11:49pm to 11:59pm.

- Grace period
- Total fee = $0.00

Example 5: Monday, 12pm to 10:01pm.

- 12pm to 6pm is charged as 12 30-minute slots: $1.20 * 12 = $14.40
- 6pm to 10:01pm is charged as one entry: $5.00
- 10% Surcharge for parking more than 10 hours: ($14.40 + $5.00) * 10% = $1.94
- Additional fee for exiting after 10pm: $3.00
- Total fee = $14.40 + $5.00 + $1.94 + $3.00 = $24.34

In this exercise, you are to write a program to calculate the parking fee so that Mr. Wu can have a good idea of how much the parking is going to cost him.

Your program should read in one integer, which is an integer between 1 and 7 representing the day of the week (1 being Monday and 7 being Sunday). It should also read in two numbers representing the time-in and time-out in 24-hour format. It should then calculate and display the parking fee (with two decimal places).

You may assume that the inputs are valid (*i.e.*, the day is within the specified range, both time-in and time-out are between 4am and midnight in 24-hour format, and time-out is no earlier than time-in).

Write on the skeleton file **parking.c** given to you. You need to include one function:

- **`double computeFee(int day, int timeIn, int timeOut)`**

  which takes in day (an integer between 1 and 7), time-in (in 24-hour format) and time-out (in 24-hour format), and returns parking fee accordingly.

You may define additional functions as needed. Check sample runs for input and output format.

Sample Runs

Five sample runs are shown below with user input highlighted in **bold**.

```
Enter day: 2
Enter time-in: 429
Enter time-out: 750
Parking fee is $8.40.
```

```
Enter day: 6
Enter time-in: 701
Enter time-out: 1949
Parking fee is $48.00.
```

```
Enter day: 7
Enter time-in: 1500
Enter time-out: 2201
Parking fee is $8.00.
```

```
Enter day: 4
Enter time-in: 2349
Enter time-out: 2359
Parking fee is $0.00.
```

```
Enter day: 1
Enter time-in: 1200
Enter time-out: 2201
Parking fee is $24.34.
```

Exercise 2: Happy Number [50 marks]

Problem Statement

For a positive integer S, if we sum up the squares of all the digits in S, we get another (possibly different) integer S_1. If again, we sum up the squares of all the digits in S_1, we get yet another integer S_2. We can repeat this process as many times as we want to get more integers. It has been proven that the integers generated in this way always eventually reach one of the 10 numbers 0, 1, 4, 16, 20, 37, 42, 58, 89, or 145.

Particularly, a positive integer S is said to be *happy* if one of the integers generated this way is 1. For example, starting with 7 gives the sequence {7, 49, 97, 130, 10, 1}, so 7 is a happy number.

In this exercise, you are to write a program to compute and compare the number of happy numbers in two given ranges.

For example, given two ranges [1, 10] and [2, 11], your program should be able to compute that there are 3 happy numbers (1, 7 and 10) in the first range and 2 (7 and 10) in the second. It should also be able to tell that there are more happy numbers in the first range than the second.

Your program should read in four integers, which represent the lower bounds and upper bounds of the two ranges (both inclusive), compute the numbers of happy numbers in each range, and then print messages stating the numbers of happy numbers as well as which range has more happy numbers.

You may assume that the input is valid (*i.e.*, the integers are all positive and the lower bounds are no bigger than the upper bounds).

Write on the skeleton file **happy.c** given to you. You need to include one function:

- **`int computeHappyNumbers(int lower, int upper)`**

  which takes in two integers `lower` and `upper`, and returns the number of happy numbers in the range [`lower`, `upper`].

You may define additional functions as needed. Check sample runs for input and output format.

Sample Runs

Two sample runs are shown below with user input highlighted in **bold**.

```
Enter 1st range: 1 1
Enter 2nd range: 1 1
The numbers of happy numbers in the two ranges are: 1 1
The numbers of happy numbers in both ranges are the same.
```

```
Enter 1st range: 1 10
Enter 2nd range: 11 100
The numbers of happy numbers in the two ranges are: 3 17
There are more happy numbers in the second range.
```

CS1010 AY2014/5 Semester 1
Practical Exam 1 (PE1)

Advice – Please read!

- You are advised to spend the first 10 minutes for each exercise thinking and designing your algorithm, instead of writing the programs right away.
- You are **not** allowed to use recursion or string functions from string.h. If in doubt, please check with the lecturer.
- If you write a function, you must have a function prototype, and you must put the function definition after the main() function.
- You may write additional function(s) not mentioned in the question, if you think it is necessary.
- Any variable you use must be declared in some function. You are **not** allowed to use global variables (variables that are declared outside all the functions).
- You may assume that all inputs are valid, that is, you do not need to perform input validity check.
- Manage your time well! Do not spend excessive time on any exercise.
- Be careful in naming your executable code. Do **not** overwrite your source code with your executable code, especially if you are using the –o option in gcc!
- The rough marking scheme for both exercises is given below.

Rough Marking Scheme For Each Exercise [50 marks]

1. Style: 10 marks
   - Are name, matriculation number, plab-id, DG and description filled at the top of the program?
   - Is there a description written at the top of every function (apart from the main() function)?
   - Are there proper indentation and naming of variables?
   - Are there appropriate comments wherever necessary?
2. Design: 10 marks
   - Are there correct definition and use of functions?
   - Are function prototypes present?
   - Is the right construct used?
   - Is algorithm not unnecessarily complicated?
3. Correctness: 30 marks
4. Deductions (not restricted to the following):
   - Program cannot be compiled: Deduct 10 marks
   - Compiler issues warning with –Wall: Deduct 5 marks.
   - Use of recursion, built-in string functions from string.h: Deduct 10 marks
   - Use of global variables: Deduct 10 marks

--- END OF PAPER ---