CCN2042 Computer Programming
Assignment 1 – Individual assignment
Submission deadline: 18:00, 17 Mar 2017 (Friday)

Expected Learning Outcomes

- Familiarise themselves with at least one high level language programming environment.
- Develop a structured and documented computer program.
- Apply the computer programming techniques to solve practical problems.

Introduction

This is an **individual assignment**. You are given a C++ program template file called *A1Template.cpp*. You are required to insert C++ codes into the template file according to the given instructions. The final program should be able to satisfy all the requirements in this specification.

Instruction

- In the given program template file, you should insert code in the *showInfo* function, so the program displays your personal particulars (e.g., your English name, your lecture and tutorial group number, and your student ID) when it begins the execution.
- To answer the following questions, insert codes into the functions *Q1*, *Q2* and *Q3*. You can also include more header files in the program if necessary.
- You are **NOT** allowed to modify any codes in the existing functions in the template file, such as the *main* function and the *compareFiles* function. Follow the requirements of each question below to decide whether you should add other functions for each question.
- You are ONLY allowed to use the C++ programming techniques under the topics up to "L5: Functions (Part 1)" from CCN2042 for the inserted codes in this assignment, unless stated otherwise in each question below.

Question 1: Permutation/Combination

Below shows the formula for calculating permutation and combination in Mathematics (for *n* larger than or equal to *r*, and both *n* and *r* larger than 0):

1. Permutation formula: $nPr = \frac{n!}{(n-r)!}$
2. Combination formula: $nCr = \frac{n!}{r!(n-r)!}$

Manually decide which formula above should be used by you in this question as follow. If the **6th** digit of your student ID number is **even**, use (1) Permutation formula. Otherwise, use (2) Combination formula.

6th digit of student ID number	The program uses …
Even	Permutation formula
Odd	Combination formula

For example, if the student ID number is 15472906A, the 6th digit is 9 (odd). Therefore, (2) Combination formula should be used in your program for question 1.

Requirements for Question 1:

1R1 When the user enters 1 in the **Program Selection Menu** (from *main*), the program displays a table showing the values of *nPr* (or *nCr*) for different values of *n* and *r* ranged from 1 to 6. See below for the table output and format:

Output for (1) Permutation formula

```
Values of nPr for different n and r
===================================
n \ r    1    2    3    4    5    6
-----------------------------------
    1    1   --   --   --   --   --
    2    2    2   --   --   --   --
    3    3    6    6   --   --   --
    4    4   12   24   24   --   --
    5    5   20   60  120  120   --
    6    6   30  120  360  720  720
```

Output for (2) Combination formula

```
Values of nCr for different n and r
===================================
n \ r    1    2    3    4    5    6
-----------------------------------
    1    1   --   --   --   --   --
    2    2    1   --   --   --   --
    3    3    3    1   --   --   --
    4    4    6    4    1   --   --
    5    5   10   10    5    1   --
    6    6   15   20   15    6    1
```

1R2 After showing the table, the console displays messages to prompt for user's input of two integer values, *n* and *r*.

1R3 Following 1R2 above, the program then displays the calculation result:-

Under inputs which *nPr* (or *nCr*) gives a valid answer, the sample output is shown below:

```
Please enter the value of n: 5
Please enter the value of r: 3
The value of nPr is 60
```

Under inputs which *nPr* (or *nCr*) gives an invalid answer, the sample output is shown below:

```
Please enter the value of n: 3
Please enter the value of r: 5
No solution, bye bye!
```

1R4 The program should keep asking for user's input of *n* and *r* and calculating the result whenever *nPr* (or *nCr*) gives a valid answer. When *nPr* (or *nCr*) gives an invalid answer, the program returns to the **Program Selection Menu**.

1R5 Write the codes for this question by writing and calling **TWO** user-defined functions in the program: the *factorial* function and the *nPr* function (or the *nCr* function). The function prototypes of these functions are shown below:

```
int factorial(int);
int nPr(int, int);
int nCr(int, int);
```

1R6 Make sure that your program supports the operation of the correct formula (based on your student ID number), and the calculation is correct.

1R7 Make sure that your program uses stream manipulators appropriately to show the table with format the same as the one shown in 1R1 above.

1R8 You can assume that the user always enter integral values larger than 0 when being asked for values of *n* and *r*. You can also assume that any inputs of *n* and *r* that results in a too large (overflowing the 32-bit integer type) factorial value, *nPr* value or *nCr* value will not be tested.

1R9 Write your codes in good style, and insert appropriate comments as necessary.

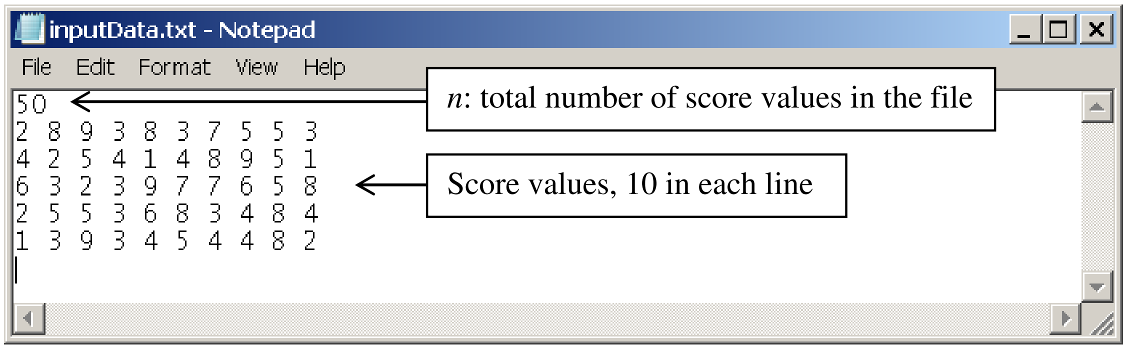
Question 2: Showing statistics

In Statistics, source data are usually stored in a file. It is common to perform data analysis by writing C++ program, which (1) reads data from a source file, (2) performs analysis on useful data, and (3) outputs statistics about the useful data either in the console or into another file.

Complete the function *Q2*, so that when the user enters 2 in the **Program Selection Menu** (from *main*), the program finds some useful statistics based on the data from an input text file.

Input text file format

Some score values are used as the source data in this question. They are stored in an input text file named as "**inputData.txt**" in a format as follow. The file contains a number (*n*) in the first line, followed by *n* different integer values (the *scores*), 10 in each line. The score values are ranged from 1 to 9 inclusively. Below shows one example.


inputData.txt - Notepad
File Edit Format View Help
n: total number of score values in the file
Score values, 10 in each line

```
50
2 8 9 3 8 3 7 5 5 3
4 2 5 4 1 4 8 9 5 1
6 3 2 3 9 7 7 6 5 8
2 5 5 3 6 8 3 4 8 4
1 3 9 3 4 5 4 4 8 2
```

Operation

In the input text file, only the scores within a particular range are considered "useful". If the **7th** digit of your student ID number is **even**, only the score values between 2 and 8 inclusively are useful. Otherwise, only the score values between 3 and 7 inclusively are useful. That is:

7th digit of student ID number	**The program considers the following as useful data …**
Even	Any score values between 2 and 8 inclusively
Odd	Any score values between 3 and 7 inclusively

Only data which are useful should be considered in calculating the statistics. The program should find the following: (1) the count of useful score values (i.e., scores between 2 and 8 (or 3 and 7) inclusively), (2) the mean (i.e., average) score of the useful score values, and (3) the count of score **5** in the source data.

In addition, the program allows the user to choose either to show the statistics results in the console (see requirement 2R2 below), or write the statistics results into an output file named **outputData.txt** (see requirement 2R3 below).

Requirements for Question 2:

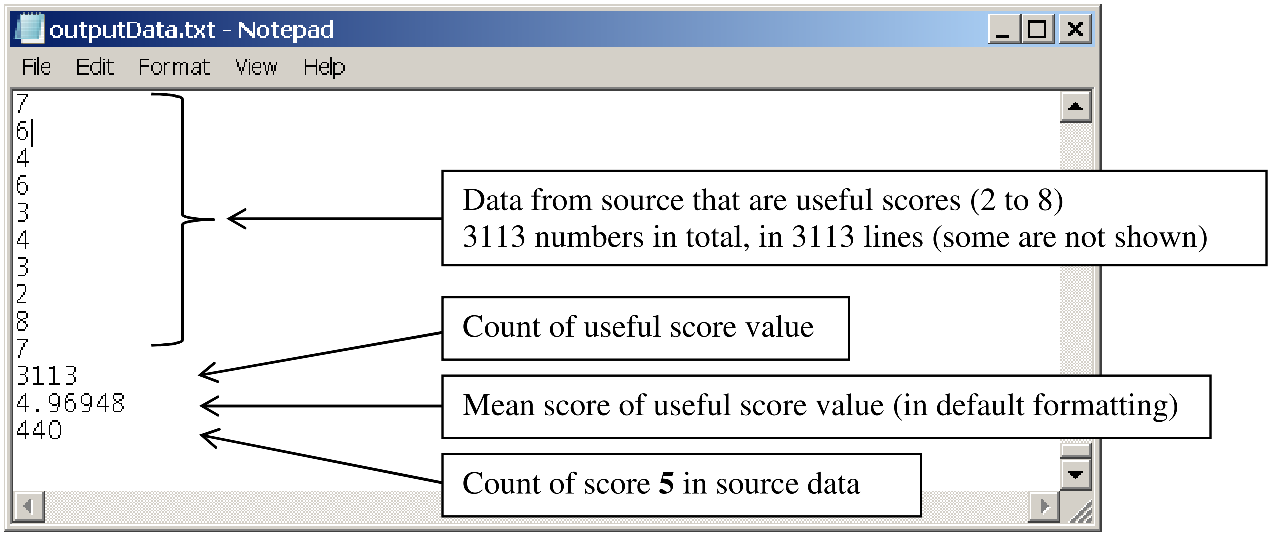
2R1 When the user enters 2 in the **Program Selection Menu** (from *main*), the program shows a prompt message to ask for the user's input option of 1 or 2. If the user chooses option 1, the program will show results in the console; if the user chooses option 2, the program will write results into an output file named **outputData.txt**.

2R2 Following 2R1 above, under option 1, the program then displays the statistics, as in the sample output below (assume that 7th digit of student ID number is even):

```
Output result to: (1) console / (2) file (outputData.txt): 1
Number of values from 2 to 8 in the source is 3113
Average of these values is 4.969
Frequency of 5 is 440
```

* In the above, the source data (from **inputData.txt**) contains 3113 score values which are from 2 to 8. The mean of these values is 4.969 (**correct to 3 d.p.**), and there are 440 counts of score 5 in the source data.

2R3 Following 2R1 above, under option 2, the program then writes the following results into an output file named **outputData.txt**: (1) EVERY useful score value from the source data, (2) the count of useful score value, (3) the mean score of the useful score values, and (4) the count of score **5** in the source data. All these values are put one in each line (see sample below, assume that 7th digit of student ID number is even):



```
7
6
4
6
3
4
3
2
8
7
3113
4.96948
440
```

outputData.txt - Notepad
File Edit Format View Help
Data from source that are useful scores (2 to 8)
3113 numbers in total, in 3113 lines (some are not shown)
Count of useful score value
Mean score of useful score value (in default formatting)
Count of score **5** in source data

2R4 Following 2R3 above, the console should display a message to notify the user that results are written into the file **outputData.txt**, as in the sample output below:

```
Output result to: (1) console / (2) file (outputData.txt): 2
Data is written into file outputData.txt
```

2R5 Basic program codes for file I/O such as declaring file stream variables, opening file and closing file are provided in the template file. A user defined function *compareFiles()* is also written. These codes should **NOT** be modified. You can assume that the input text file to the program is always named as **inputData.txt**. However, to allow writing contents into an output file, you will need to write codes to open the file **outputData.txt** when the user chooses option 2 at the beginning.

2R6 You can assume that the input text files to be tested are always in correct format.

2R7 Appropriate use of loop is required in your codes. **DO NOT** add other user-defined functions in this question.

2R8 Make sure that your program considers the correct score values as the useful data based on your student ID number.

2R9 Three sets of testing files (in folders **10_1000**, **20_500** and **30_4000**, inside the file **sample.zip**) are given. 30% of marks in this Question 2 will be awarded if your program can produce output files with the same contents as the given sample output files (**sampleOutput.txt**) using these input text files. No marks will be given if any error is found (see Grading Criteria on page 9).

2R10 Following 2R9, your program should display the following message after your output in 2R2 or 2R4, as a result of the function call *compareFiles* at the end of function *Q2*, if your codes are correct:

Under input option 1:

```
No file to open (outputData.txt)
```

Under input option 2:

```
outputData.txt <---> sampleOutput.txt ........... [OK]
```

2R11 Write your codes in good style, and insert appropriate comments as necessary.

Hints on file I/O

You need to add `#include <fstream>` in order to use the file I/O related codes. Similar to *cin* that uses >> to read data from keyboard, reading data from file follows a similar way by replacing *cin* with the **file stream variable**.

Writing contents to file can be done similarly by replacing *cout* with the file stream variable.

```
// Assume "example.txt" stores four numbers
//   12 and 34 in the first line (separated by space)
//   20 and 40 in the next line (separated by space)

ifstream inFile;      // Declare input file stream variable
ofstream outFile;     // Declare output file stream variable

inFile.open("example.txt"); // Open the file named example.txt
outFile.open("output.txt"); // Open the file named output.txt

int a, b;             // Declare variables to store file data ...
inFile >> a;          // a stores value 12
inFile >> b;          // b stores value 34

outFile << a + b << endl;  // write 46 (in one line) to output file

inFile >> a;          // a stores value 20
inFile >> b;          // b stores value 40

outFile << a + b << endl;  // write 60 (in one line) to output file

inFile.close();
outFile.close();
```

Placing your files

Files to be read should be put in the **same folder as your ".cpp" source code by using file explorer**. Any output files will be written into this folder too. Therefore, in your program testing, make sure that source files **inputData.txt** and **sampleOutput.txt** are put in the correct folder.

<u>Question 3: Parking charge</u>

The table below shows the parking charge of two carparks, Carpark A and Carpark B.

Carpark A			**Carpark B**		
Vehicle type	First rate (first 3 hours)	Second rate (after 3rd hour)	Vehicle type	First rate (first 3 hours)	Second rate (after 3rd hour)
Car	\$10 / hour	\$13 / hour	Car	\$12 / hour	\$15 / hour
Bus	\$6 / hour	\$8 / hour	Bus	\$8 / hour	\$10 / hour

Manually decide which carpark to use with the **<u>8th</u>** digit of your student ID number as follow:

8th digit of student ID number	**The program uses …**
Even	Carpark A
Odd	Carpark B

For example, if the student ID number is 15472906A, the 8th digit is 6. Therefore, Carpark A should be used in your program.

Requirements for Question 3:

3R1 When the user enters 3 in the **Program Selection Menu** (from *main*), the program displays prompt messages to ask for the following user inputs, <u>in the same sequence below</u>:

(i) A character showing the type of vehicle: ‘c’ for Car, ‘b’ for Bus;
(ii) An integer between 0 to 23 showing the hour the vehicle entered the carpark;
(iii) An integer between 0 to 59 showing the minute the vehicle entered the carpark;
(iv) An integer between 0 to 23 showing the hour the vehicle left the carpark;
(v) An integer between 0 to 59 showing the minute the vehicle left the carpark.

3R2 Following 3R1 above, the program then displays the parking charge. It shows which carpark (A or B) is used in the first line, followed by the parking charge summary:

```
PARKING CHARGE (Carpark A)
--------------------------
Type of vehicle:         Car
Time In:               00:50
Time Out:              01:59
                       -----
Parking Time:          01:09
Round Total (hours):       2
                       -----
Total Charge:          $  20
```

* In the above, the user inputs (i) ‘c’, (ii), 0, (iii) 50, (iv) 1, (v) 59.

3R3 Following 3R2 above, after showing the parking charge, the program displays message to prompt for a character input by the user to see if the user wants to perform another calculation. If the user enters 'y', the program should ask for user inputs of (i) to (v) in 3R1 again. Otherwise, the program returns to the **Program Selection Menu**. You can assume the user always enters a single character here.

3R4 There is no fractional hour charge. Therefore, in the calculation, your program needs to round the parking time up to the next hour for any parking period less than an hour.

3R5 You can assume the user would not enter any leaving time earlier than the entering time. You can also assume the vehicle is not parked overnight.

3R6 You can assume the user always enter 'c' or 'b' in input (i) in 3R1, and integer values that are within the valid range in inputs (ii) to (v) in 3R1.

3R7 The parking charge summary should be displayed according to the format shown in 3R2 above. Use appropriate stream manipulators in your answer. To show the hour or minute value in a two-digit format, you may find the stream manipulator *setfill( )* useful. For example:

```
cout << setfill('0');     // A sticky manipulator
                          // Set the fill character as '0'
                          // Filling '0' in empty space
                          // when field is wider than output
```

3R8 Make sure that your program calculates using the correct carpark charge based on your student ID number, and considers the first and second rate in the calculation.

3R9 Appropriate use of loop is required in your codes. **DO NOT** add other user-defined functions in this question.

3R10 Write your codes in good style, and insert appropriate comments as necessary.

Submission

This is an individual assignment. You are required to submit the **final** C++ source code file, which contains your codes inserted into the template file provided. Use your student name and ID as the filename: *StudentName_ID.cpp*. Submit the file to Moodle before the deadline: **18:00, 17 Mar 2017 (Friday) (Week 7)**.

Late submission is subject to 20% deduction in your final marks for each day (including public holiday and Sundays). No late submission is allowed **4 days** after the due date.

Grading Criteria

	Marks allocation (out of **100** marks)	Grading aspects	Percentage
ShowInfo	3	Correctness	100%
Question 1	30	Correctness	85%
		Program design	10%
		User interface[1]	5%
Question 2	30	Correctness on file outputs (based on given input files)	30%
		Correctness on file output	20%
		Correctness on input and console output	40%
		Program design	5%
		User interface	5%
Question 3	30	Correctness	85%
		Program design	5%
		User interface	10%
Program standard, comments, line spacing, program clarity, indentation	7	n/a	n/a

Marks deduction

Marks will be deducted if the program fails to be compiled. The deduction is from 5 to 20 marks, depending on how serious the compilation error is. Note that if the program contains unacceptably too many serious compilation errors, your program will score 0 marks. Compilation errors also lead to failure in the program correctness if the function cannot be tested.

Failure to decide the program requirements correctly based on your student ID would lead to marks deduction under program correctness.

You should make sure your program can be compiled and run properly in our standard program testing environment, i.e., by Microsoft Visual Studio 2010 Professional.

Ensure the originality of your work. Plagiarism in any form is highly prohibited.

- End -

[1] E.g., proper messages to users, clear display of output, etc.