COS30008 Data Structures and Design
Patterns
Problem Set 1

Objectives

- To practice solving problems using array
- To practice solving problems using class
- To practice writing solutions to problems in a clear and succinct way

Question 1

Write a program using array for the following problem statement:

PTR Company has three different branches selling various car dash cam. The manager would like to calculate monthly sales figure for four different dash cam models for all the branches. Create a program that could store all sales figures (unit sold) by each branch and selling price for each model of dash cam. Refer to the following tables:

Example of Unit Sold (monthly) (**Unit Sold** array):

Dash Cam model	**SJ Branch**	**PJ Branch**	**KL Branch**
RS Pro with GPS	5	4	3
Transcend Drive Pro	2	2	3
H203 1080P	3	2	5
Pioneer	4	5	3

Dash Cam selling price (**Price** array):

Dash Cam model	**Price (RM)**
RS Pro with GPS	730
Transcend Drive Pro	850
H203 1080P	150
Pioneer	350

The program stores the unit sold into a two-dimensional array named **UnitSold**. Every month, the user will enter all sales figures (unit sold) while the price of each model is fixed (refer to the **Price** array above).

After all the input process completed, the program will display the following output (based on the table above):

Total gross sales for Branch SJ is: RM …
Total gross sales for Branch PJ is: RM …
Total gross sales for Branch KL is: RM …

The highest sales figure is RM … by Branch ..
The most popular dash cam model is …… with unit sold of .. units.

Marking Criteria
Correctness (5 marks)
Efficiency (2 marks)
Validation (2 marks)
Output presentation (3 marks)

Question 2

In this task, we define a simple data type to represent combinations

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

A combination is a selection of items from a collection, such that the order of selection does not matter.
Here, we are interested only in the binomial coefficient

$$\frac{n!}{k!(n-k)!}$$

Whenever $k \leq n$, and which is zero when $k > n$.

Define a C++ class that implements combination. The specification of class Combination is shown below:

```
#pragma once
class Combination
{
private:
   unsigned int fN;
   unsigned int fK;

public:
   // constructor for combination n over k Combination(
   unsigned int aN, unsigned int aK );
```

```
    // getters
    unsigned int getN() const;
    unsigned int getK() const;

    // call operator to calculate n over k
    // We do not want to evaluate factorials.
    // Rather, we use this method
    //
    //       n          (n-0)     (n-1)              (n - (k - 1))
    // (         ) =     -----  *  -----  * …..... * -------------
    //       k             1         2                     k
    //
    // which maps to a simple for-loop over 64-bit values.
    //   https://en.wikipedia.org/wiki/Combination
    unsigned long long operator()() const;
};
```

The class Combination has two instance variables representing n and k. The constructor initializes those instance variables and the getters provide read-only access to their values. The call operator() returns the value of a combination. We use the method described on WikiPedia: https://en.wikipedia.org/wiki/Combination rather than calculating the factorials, which can become very big numbers. You need to use the type **unsigned long long** for calculation in order to work with 64-bit values.
To test your implementation of class Combination, we can use the following test driver.

```
#include <iostream>
#include "Combination.h"
void runProblem2()
{
Combination a(6, 2);
Combination b(5, 2);
Combination c(28, 14);
Combination d(52, 5);
Combination e(5, 5);
cout << a.getN() << " over " << a.getK() << " = " << a() << endl;
cout << b.getN() << " over " << b.getK() << " = " << b() << endl;
cout << c.getN() << " over " << c.getK() << " = " << c() << endl;
cout << d.getN() << " over " << d.getK() << " = " << d() << endl;
cout << e.getN() << " over " << e.getK() << " = " << e() << endl;
```

Marking Criteria
Correctness (5 marks)
Efficiency (2 marks)
Validation (3 marks)
Output presentation (3 marks)

Submission

Upload ZIP/RAR document titled yourname_W04_PS1 into the correct submission folder in Canvas before 11.59pm Malaysian time. Late submission will be penalized / not accepted.

Your document should contain your source code, screen shots and other necessary files. Show your code and output during lab to receive your marks.