CIS 200 - Lab 2

Fall 2016

Note: in order to complete these tasks involving programming, you must write your own main() function to exercise the code.

# Task 1

Given that A[MAX\_ROWS][MAX\_COLUMNS] is a 2 dimensional array of integers, write a C++ function *Even* to find the total number of even elements in the array.

It should **have 3 input parameters**:

* A[*length*][*width*]
* *Length*
* *Width*

The function should **return an integer**. The constants MAX\_ROWS and MAX\_COLUMNS may be accessed globally by defining the following global constants:

const int MAX\_ROWS=3;

const int MAX\_COLUMNS=2;

The array A is initialized by the following statement in your main():

int A[MAX\_ROWS][ MAX\_COLUMNS] = { {3 , 2}, {4, 5}, {2, 2} };

# Task 2

Given that Sale[NUM\_STORES][NUM\_MONTHS][NUM\_DEPTS] is a three dimensional array of double type.

Write a C++ function to calculate and print the total value of sales during a specific month by each department and in each store. *Hint: You must write the function such that the month is specified*.

The output will be a value for each possible department and each possible store for a total of 4 outputs.

The return type of the function is *void* and the function should have 2 input parameters: Sale array and the *month* user specified. The constants NUM\_STORES, NUM\_MONTHS, and NUM\_DEPTS may be accessed globally by defining the following global constants:

const int NUM\_DEPTS = 2;

const int NUM\_STORES= 2;

const int NUM\_MONTHS=12;

The array Sale is initialized by the following statement in your main():

float Sale[NUM\_STORES][NUM\_MONTHS][NUM\_DEPTS] =

{ 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2,

2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, 3.0, 3.1, 3.2,

3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, 4.0, 4.1, 4.2,

2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, 3.0, 3.1, 3.2

};

# Task 3

Consider the following question:

“How many combination of X items can I make out of Y items?”

Let the definition of Combinations in this context be:

Combinations(Y, X) =   
Y if X = 1   
1 if X = Y   
Combinations(Y-1, X-1) + Combinations(Y-1, X) if Y > X > 1

Using the above definition, **write a recursive function** Combination(y, x) to calculate the number of combinations of X items made out of a total of Y items.

# Task 4

Write a test (driver) function to test the previously defined function, and have it print out the number of combinations of 4 items made out of a total of 8 items. [i.e., Combinations(8, 4)]

Let it print out, “The number of combinations of 4 items made out of a total of 8 items is: “, followed by the total calculated from the invocation of function Combinations.

# Task 5

Use cout to print out a trace of ***each recursive level*** for the function call, Combination(8, 4). Through this print out, you will have a better understanding on how the recursion works.

At each recursive level, print out the values for parameter Y and X as well as the recursive level. Assume the recursive level starts from 1. The output of your code should look like:

Recursive level 1:  
Y = 8 X = 4  
Recursive level 2:  
Y = 7 X = 3

**Hint: You may modify the recursive function’s signature to Combinations(Y, X, level) where level indicates the recursive level you’re at.**

Note that a **higher** (greater number) recursive level indicates that you are further down into the recursive calls.