Updated Readme *Group 16*

-A brief identification of the team members.

Wu Yuxi, UID: 3035534957: year 2 science student. I am responsible for game design, calculating results and storing records.

Liang Zhuo, UID:3035608849: year 1 science student. I am responsible for game design and basic frame build-up.

- A game description with basic game rules.

This is a 24 points game that allows several players to compete. Each player will be given four randomly generated integers from 1 to 9 and player may follow the instructions to try to get the outcome of 24. The player can determine to perform factorial or square root on a single number or add, subtract, multiple, divide, combination or permutation between two numbers. If he used simple mathematical operations (e.g. add, subtract, multiple, divide), we add one to his score. If complex operations (e.g. factorial, square root, combination or permutation) are used, his score will plus three. Several additional conditions need to be focus: for clearer game flow, we round up the square root and division to avoid long decimal places; moreover, subtraction can only be done in one direction, which avoid negative results; If the player want to do factorial on a integer bigger than 9, the system will return -999 because we regard player’s choice as giving up (it’s already so hard that 362880 can turn into 24) and remind him not to do so as well. When all the players finished their game, a file named “All\_Possible\_Results.txt” will be generated showing all the possible solutions for each player’s case. In addition, another file named “Players\_record.txt” will write in players’ names and their score if they chose to keep the record at the beginning. The record will be displayed in the order that higher score shown first.

- A list of features / functions that you **have implemented**, vis-a-vis each of the items 1 to 5 listed under **coding requirements** above.

No.1 Generation of random game sets or events

The four numbers are randomly generated using srand( ).

No.2 Data structures for storing game status

We use an array to store the four numbers. We created two structs for linked list building.

No.3 Dynamic memory management

We use linked list to keep track of players’ score and sort them as well. One for players who would disclose their scores(sorted), and one for players who would not.

No. 4 File input/output (e.g., for loading/saving game status)

“All\_Possible\_Results.txt” is the output file to tell the player all the possible answers.

“Players\_record.txt” stores all the players’ names and scores if permitted, and only the names for players who hided their scores.

No.5 Program codes in multiple files

We have written makefile and source code.

-Any non-standard C/C++ libraries, if any, that are used in your work, and what features/functions in your game are supported by these libraries.

<cstdlib> to generate random numbers

<fstream> to write in an output file

<cmath> to use round( ) and sqrt( )

-Compilation and execution instructions. Simply put, this serves like a manual to your program.

First input your name. Next input “yes” or “no” to decide whether to keep your score. Then the system will give you four numbers. Follow the words to try your best to obtain 24. Then we will ask you whether there is another player. If yes, the game will start again. If no, the game will end.