Project 1: A Simple Game on Addition/Subtraction
Due: 11:59PM, Friday October 1st, 2004.

Project Objective:
1. get familiar with the process of completing a programming project.
2. learn the basics of printf() and scanf().
3. declare and use of variables of the basic data types: int, char, float.
4. use basic relational and logical operators.
5. use basic if and/or if-else statements for program selection.
6. use array and basic loops (while, do-while, for) for repetition.
7. do simple data management.
8. generate random numbers in UNIX with rand().

Project Description:
In this project, you will implement an interactive game for first graders to practice addition and subtraction problems on single-digit numbers. The game starts with a brief description of the rules. After the game, the number of appearances for each operator (+/-) and each operand (0,1,…,9), the most and least frequent operand, as well as the total number of problems, and the average number of tries the player took to get the answer will be displayed. The three parts of the project are described in detail as follows:

1. Print out a welcome message and explain how to play the game. Use your own language, you don’t have to copy the following example message:

   ***************************************************
   ** WELCOME TO THE MATH GAME **
   ***************************************************

   How to play: This is a single player game that drills you on addition and subtraction problems. Simply enter your answer and you will be asked to repeat if your answer is incorrect. 10 problems form a drill and at the end of each drill, you can decide to continue the game or not. Good luck!

   ***************************************************

2. Game section:
   Because this game is designed for first graders, it should be kept simple. Thus, we restrict the problems to single digit addition and subtraction. For subtraction problems, the program should make sure that it is always the larger number minus the smaller number (How many first graders know negative numbers?).

   Your program should use the rand() function to randomly generate the two operands in your addition or subtraction math problem. The operand should be between 0 and 9 (However, the answer may not be. For example, 5 + 6 = 11.). You should also determine the operator (+ or -) with rand(). You then print the problem on the screen.
Next, your program reads in the answer from the screen and check whether it is correct. If the player gives the correct answer, you should reprint the problem and the answer using a non-trivial format such as any one of the following or something of your choice:

```
  _             _   _
 | |     |     _|_       ___       --
 |_|     |    |_    _|      |              --
0000  4  4  5555  6666     +
  0  0  4  4  5  6       +    =====
0  0  4444  5555  6666     ++++++
  0  0  4  5  6  6     +    =====
0000  4  5555  6666     +
```

Here is an example of how the game should work:

**Problem 1**

4 + 3 = 7

That’s the correct answer. Good job!

```
|   _ _ _ _ |
|   _ _ _ _ |
```

**Problem 2**

3 + 3 = 6

That’s the correct answer. Good job!

```
|_ _ _ _ _ | |
|_ _ _ _ _ |
```

**Problem 3**

2 + 8 = 9

Sorry, that’s not the correct answer. Try again.
2 + 8 = 8

Sorry, that’s not the correct answer. Try again.
2 + 8 = 10

That’s the correct answer. Good job!

```
|_ _ _ _ _ |
|_ _ _ _ _ |
```

**Problem 4**

8 - 3 = 5

That’s the correct answer. Good job!

```
|_ _ _ _ _ |
|_ _ _ _ _ |
```

After each set of 10 problems (or a drill), you must prompt the user asking whether to continue by, for example, the following message:
Do you want to keep playing? [Y/N]

If the user chooses ‘y’ or ‘Y’, then continue a new drill of 10 problems with continuous problem number. That is, the first problem in the second drill should be Problem 11.

If the user chooses ‘n’ or ‘N’, then print out a thank you message such as:

Thanks for playing!

and move on to the last stage of the program.

3. Game summary and statistics:
   Once the player decides to stop the game (see the above paragraph), display the following information:

   Game summary:
   1 drill(s) of 10 problems have been completed.
   4 problem(s) are on addition and 6 problem(s) are on subtraction.
   You took on average 1.23 tries to get the correct answer. Congratulations!

   Occurrence statistics
   #0 = 2
   #1 = 0
   #2 = 1
   #3 = 4
   #4 = 3
   #5 = 1
   #6 = 4
   #7 = 2
   #8 = 2
   #9 = 1

   The most frequent number: 3, 6 (4 times).
The least frequent number: 1 (0 times).

The average number of tries to get the correct answer is defined as the ratio of the total number of answers (both correct and incorrect) and the total number of problems. For example, if a player plays 3 drills and out of the 30 problems, 25 were answered correctly on the first try, 4 on the second try, and 1 on the third try, then the average is 
\[
(1 \times 25 + 2 \times 4 + 3 \times 1) / 30 = 37 / 30 = 1.23.
\]
Note: print out only two digits after the decimal point.

Project Requirements:
1. You must program using C under GLUE UNIX system and name your program p1.c.
2. Restrict yourself to what we have discussed in lecture and recitation. Using advanced features of C that will not be covered by Tuesday, September 28th will result in lost of points.
3. Your program must be properly documented.
4. The following is expected as the output of your program:
   - Welcome message and description of the game: Refer to section 1 in this project description.
   - Game section: Refer to section 2 in this project description. You do not have to use the 7-segment digit format. You can use whatever way you want as long as it is non-trivial [i.e. you can’t just print out the number as is].
   - Game summary and statistics: Refer to section 3 in this project description.
5. Submit your program `p1.c` electronically before the due time. How to do this will be discussed in the recitation sections.

Grading Criteria:
Correctness: 80%
Good coding style: 10%
Proper documentation: 10%

Project Template:

```
#include <stdio.h>
#define DrillSize 10

int main(void)
{
    int probNum, operand1,  operand2,  result;
    char operator, decision;
    float precision;
    int stat[12];

    <print out the welcome message and simple rules of the game>

    <generate a problem and print it out in plain format>
    <read in result, check its correctness and act accordingly>
    <print out the final problem and answer in any fancy format of your choice.>

    <repeat the above two steps for DrillSize (10) times>

    <ask the user if he/she wants to continue playing>
    <start a new drill if decision == “Y” or ‘y’ and stop if decision ==“N” or ‘n’>

    <print out the game statistics>

    return 0;
}
```