EE231002 Introduction to Programming

Lab12. Poker Dice

Due: Dec. 31, 2021

The classic poker dice game is played with 5 dice and two or more players. Players take turns to roll 5 dice, and then compare their dice according to the table below which lists the ranking from high to low.

Hands	Condition	Example
Five of a kind	All five dice have the same face	⚄ ⚄ ⚄ ⚄ ⚄
Four of a kind	Four dice have the same face.	⚁ ⚁ ⚁ ⚁ ⚄
Full house	Three dice have the same face; with the other two a different face.	⚁ ⚁ ⚁ ⚄ ⚄
Straight	Continuous faces from 1 to 5 or 2 to 6.	⚀ ⚁ ⚂ ⚃ ⚄
Three of a kind	Three dice with the same face.	⚁ ⚁ ⚁ ⚂ ⚃
Two pair	Two dice have the same face, with two other same face dice.	⚁ ⚁ ⚄ ⚄ ⚂
One pair	Two dice with the same face.	⚁ ⚁ ⚂ ⚄ ⚅
Bust	None of the above	⚀ ⚁ ⚂ ⚄ ⚅

Your assignment is to write a `C` program to complete the following tasks:

1. Implement a `rollDiceFair(int dice[5])` function that uses the built-in pseudo random number generator function **`rand()`** from the `stdlib` library to simulate rolling 5 fair dice that each face of a die has equal probability.

2. Test this `rollDiceFair(int dice[5])` function by calling it 390,000 times to observe the occurrence of each face of each die.

3. A function `rollDiceUnfair(int dice[5])` is provided to you by the object file **`roll.o`**. This function simulate the rolling of the 5 dice but with two dice having higher probability of getting ⚅ (50% higher probability). Please verify this function again by calling it 390,000 time.

4. Implement a function `rank(int dice[5])` that takes the 5 dice rolled either by `rollDiceFair` or `rollDiceUnfair` function as input and returns the ranking as shown in the table above. The integer values of those rankings are defined in the **`roll.h`** header file as macros.

5. Then your program simulates two players playing the Poker Dice game 10,000,000 times. Player 1 rolls the dice using `rollDiceFair` function, while Player 2 rolls the dice using `rollDiceUnfair` function. If rankings of these two rolls are equal, then both players need to repeat rolling dice until a winner is determined. After 10,000,000 games, print out the winning and losing percentages of Player 2.

The output format of your program should look like the following:

```
$ ./a.out
RollDiceFair 390000 times
Occurrences  1     2     3     4     5     6
  die 1:   65060 64806 64570 65152 65062 65350
  die 2:   65066 65350 65117 64572 64936 64959
  die 3:   64947 64970 64951 65124 65146 64862
  die 4:   65048 65267 65072 64831 64877 64905
  die 5:   64724 64987 65562 64735 65014 64978

RollDiceUnfair 390000 times
Occurrences  1     2     3     4     5     6
  die 1:   60209 59903 60239 59984 59690 89975
  die 2:   60403 60128 60015 59910 60010 89534
  die 3:   64812 64921 64837 65333 64966 65131
  die 4:   64923 64839 65141 65109 64924 65064
  die 5:   64920 64993 64968 65126 64985 65008

Player 2 using unfair dice playing 10000000 games:
  Winning percentage: xx.xx%
  Losing percentage: yy.yy%
```

Overall, 3 functions, in addition to the **main** function, are needed. Their declarations are shown below and can be found in the **roll.h** file.

```
void rollDiceFair(int dice[5]);   // roll fair dice
void rollDiceUnfair(int dice[5]); // roll unfair dice
int rank(int dice[5]);            // Given dice, return corresponding rank
```

The **rank** function takes the **dice** rolled as input and output the ranking. To make the program more legible, the following macros are defined in **roll.h**:

```
#define FiveKind 0
#define FourKind 1
#define FullHouse 2
#define Straight 3
#define ThreeKind 4
```

```
#define TwoPair 5
#define OnePair 6
#define Bust 7
```

In this way, the higher ranking hands have smaller values. The input **`dice`** is an **`int array`** of 5 elements. Each element can have values from `0` to `5` to represent ⚀ to ⚅.

For the lab, your program should be compiled using the following command:

```
$ gcc lab12.c roll.o
```

Also, in addition to the `roll.h` header file, you need to include `stdlib.h` for the **`rand`** function.

Notes.

1. Create a directory **`lab12`** and use it as the working directory.
2. Name your program source file as **`lab12.c`**.
3. The first few lines of your program should be comments as the following.

   ```
   // EE231002 Lab12. Poker Dice
   // ID, Name
   // Date:
   ```

4. After you finish verifying your program, you can submit your source code by

   ```
   $ ~ee2310/bin/submit lab12 lab12.c
   ```

   If you see a "submitted successfully" message, then you are done. In case you want to check which file and at what time you submitted your labs, you can type in the following command:

   ```
   $ ~ee2310/bin/subrec lab12
   ```

   It will show the submission records of lab12.

5. You should try to write the program as efficient as possible. The format of your program should be compact and easy to understand. These are part of the grading criteria.