# lab08

```
$ gcc lab08.c
$ ./a.out
Categories      Probability
Straight flush    0.0014%
Straight flush   0.0014%
Four of a kind    0.0236%
Four of a kind   0.0236%
Full house        0.1443%
Full house       0.1443%
Flush             0.1961%
Flush            0.1961%
Straight          0.3520%
Straight         0.3520%
Three of a kind   2.1138%
Three of a kind  2.1138%
Two pair          4.7533%
Two pair         4.7533%
One pair         42.2540%
One pair         42.254%
High card        50.1616%
High card        50.162%

Program output is incorrect
CPU time: 2.12501 sec
score: 95
o. [Output] Program output is incorrect
o. [Format] Program format can be improved
```

# lab08.c

```c
// EE231002 Lab08. Poker Hands
// 109061158, 簡佳吟
// Date: 2020/11/23


#include <stdio.h>
#include <stdlib.h>
#define N 10000000      // number of total trials

int category(int card[5]);
    // This function categorizes the card,and return numbers represent its type
    //  input: card[5]
    //  output:
    //   0 straight flush
    //   1 four of a kind
    //   2 full house
    //   3 flush
    //   4 straight
    //   5 three of a kind
    //   6 two pair
    //   7 one pair
    //   8 high card

int main (void)
int main(void)
{
    int card[5];                // array of card
    int i, k;                   // index for loop
    long int j;                 // index for loop
    int card_type;              // type of card
    float type[9] = {0};        // count each type

    for (j = 1; j < N; j++) {
        for (i = 0; i < 5; i++) {
             card[i] = (rand() / 1000) % 52;    // shuffle
            for (k = 0; k < i; k++) {
                if (card[k] == card[i]) i--;    // if the card has same number
                                                // shuffle again
            }
        }
```

```
40
41          card_type = category(card);             // call category function
42
43          switch (card_type) {                    // select card type
44              case 0: type[0]++; break;           // and count the number
45              case 1: type[1]++; break;
46              case 2: type[2]++; break;
47              case 3: type[3]++; break;
48              case 4: type[4]++; break;
49              case 5: type[5]++; break;
50              case 6: type[6]++; break;
51              case 7: type[7]++; break;
52              case 8: type[8]++;
53          }
54
55      }
56
57      printf("Categories      Probability\n");                    // prompt
58      printf("Straight flush    %.4f%%\n", type[0] / N * 100);    // prompt
59      printf("Four of a kind    %.4f%%\n", type[1] / N * 100);    // and
60      printf("Full house        %.4f%%\n", type[2] / N * 100);    // calculate
61      printf("Flush             %.4f%%\n", type[3] / N * 100);    // probability
62      printf("Straight          %.4f%%\n", type[4] / N * 100);
63      printf("Three of a kind   %.4f%%\n", type[5] / N * 100);
64      printf("Two pair          %.4f%%\n", type[6] / N * 100);
65      printf("One pair         %.4f%%\n", type[7] / N * 100);
66      printf("High card        %.4f%%\n", type[8] / N * 100);
67      return 0;                                                  // done and return
68  }
69
70
71  // This function categorizes the card,and return numbers represent its type
72  //  input: card[5]
73  //  output: 0 straight flush
74  //          1 four of a kind
75  //          2 full house
76  //          3 flush
77  //          4 straight
78  //          5 three of a kind
79  //          6 two pair
80  //          7 one pair
```

```
81 //          8 high card
82
83 int category(int card[5])
84 {
85
86     int rank[13] = {0};     // array for card's rank
87     int suit[4] = {0};      // array for card's suit
88
89     int straight = 0;       // number for scanning whether it is straight
90     int flush = 0;          // number for scanning whether it is flush
91     int pair = 0;           // number for scanning whether it has pair
92     int three_kind = 0;     // number for scanning
93                             // whether it has three of a kind
94     int four_kind = 0;      // number for scanning
95                             // whether it is four of a kind
96
97     int i;                  // index for loop
98     int num_consec = 0;     // number for checking whether it is straight
99
100    for (i = 0; i < 5; i++) {   // initialize rank array
101        rank[card[i] % 13]++;   // rank 0 is A and rank 1 is 2,and so on
102    }
103
104    for (i = 0; i < 5; i++) {               // card 0 ~ 12 is spade
105        if (card[i] < 13) suit[0]++;        // card 13 ~ 25 is heart
106        else if (card[i] < 26) suit[1]++;   // card 26 ~ 38 is diamond
107        else if (card[i] < 39) suit[2]++;   // card 39 ~ 51 is club
108        else  suit[3]++;
109    }
110                                                // scan for straight
111    for (i = 0; i < 13 && rank[i] == 0; i++) ;  // find the smallest index s.t.
112                                                // rank[i] is not 0
113    for (; i < 13 && rank[i] > 0; i++) {
114        num_consec++;                           // check whether rank[i] to
115                                                // rank[i + 5] is not 0
116    }
117    if (num_consec == 5) straight = 1;          // if num_consec is 5
118                                                // is straight
119
120
121    for (i = 0; i < 4; i++) {                   // scan for flush
```

```
122         if (suit[i] == 5) flush = 1;            // if there exists suit[i] is 5
123                                                 // the card has the same suit
124     }
125     for (i = 0; i < 13; i++) {                  // scan for pair,
126                                                 // three of a kind,
127                                                 // and four of a kind
128         if (rank[i] == 2) pair += 1;            // if rank[i] = 2
129                                                 // there exists two card
130                                                 // with same value
131         if (rank[i] == 3) three_kind = 1;       // if rank[i] = 3, there exists
132                                                 // three of a kind
133         if (rank[i] == 4) four_kind = 1;        // if rank[i] = 4,
134                                                 // is four of a kind
135     }
136
137                                                     // select for each type

    This line has more than 80 characters
138     if (straight && flush) return 0;                // straight flush
139     else if (four_kind) return 1;                   // four of a kind
140     else if (three_kind && pair == 1) return 2;     // full house
141     else if (flush && !straight) return 3;          // flush
142     else if (straight && !flush) return 4;          // straight
143     else if (three_kind && pair == 0) return 5;     // three of a kind
144     else if (pair == 2) return 6;                   // two pair
145     else if (pair == 1 && !three_kind) return 7;    // one pair
146     else return 8;                                  // high card
147 }
148
```