LAB 11/12

In this experiment, we suppose rand() can generate data with uniform distribution from 0 to RAND_MAX

1. **Homework Problem I:**

   A. Please enter one positive integer x from the keyboard and find out the square root (r) of this positive integer. **(Using Newton's method only)**.

   Please repeat the process:

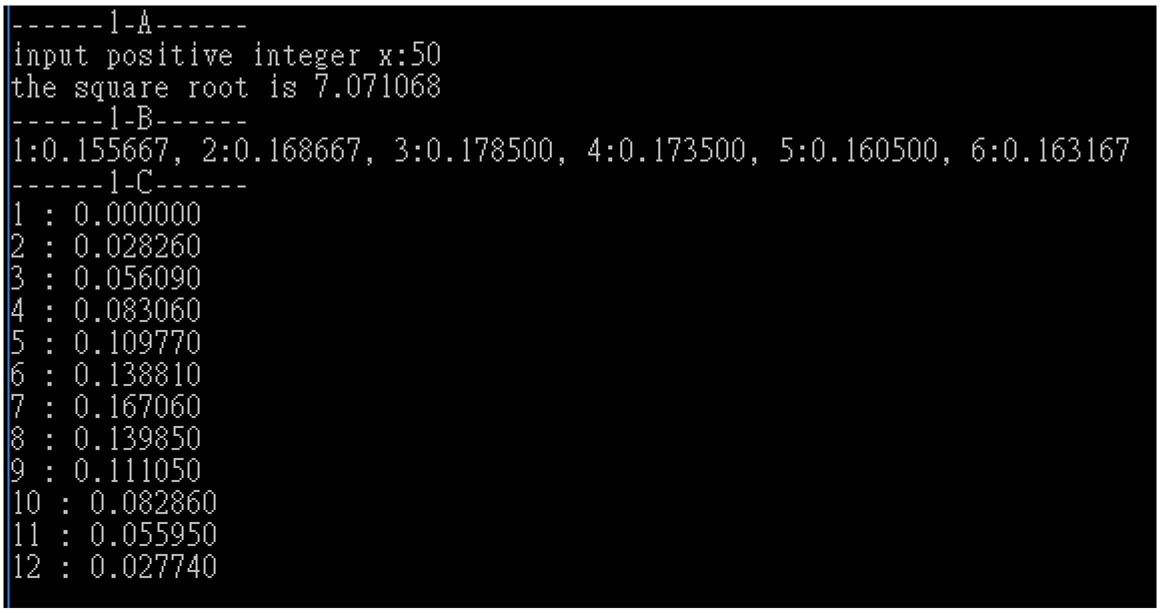
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

   until $|x_{n+1} - x_n| \leq 10^{-8}$.

   B. Write a program to simulate 5000 rolls of six-sided die and output the frequency of each number.

   C. Charlie tosses a pair of six-sided dice. What number **(sum of the face value of both dice)** is **most likely to thrown**? Please write a program to simulate the process of the toss.

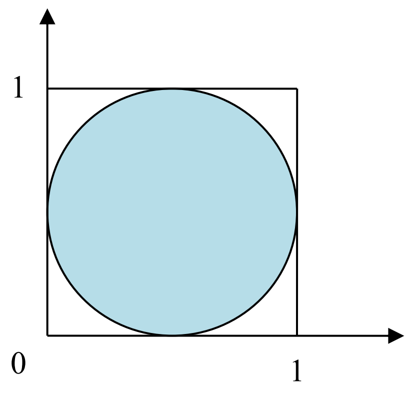
   Sample output:



```
------1-A------
input positive integer x:50
the square root is 7.071068
------1-B------
1:0.155667, 2:0.168667, 3:0.178500, 4:0.173500, 5:0.160500, 6:0.163167
------1-C------
1 : 0.000000
2 : 0.028260
3 : 0.056090
4 : 0.083060
5 : 0.109770
6 : 0.138810
7 : 0.167060
8 : 0.139850
9 : 0.111050
10 : 0.082860
11 : 0.055950
12 : 0.027740
```

2. **Homework Problem II:**

A. Write a program to simulate throwing darts. (射鏢遊戲)



Use a random number generator to obtain 1,000,000 pairs of floating-point numbers (x, y) satisfying 0< x<1, 0< y<1.

Print the proportion P of throws that hit the dart board, that is, the proportion of pairs (x, y) that are inside the circle. Also print 4* P.

Notice that the geometry of the problem leads us to expect P to be about $\frac{\pi}{4}$. Thus 4* P provides an approximation of π.

Note: You can use the following process to generate random number **between 0~1**:

double seed;
const double mpy = 25173.0;
const double inc =13849.0;
const double mod =65535.0;
input variable "seed" then calculate the following formula:

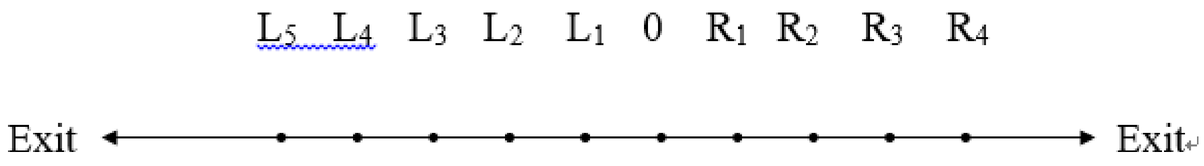
seed = (seed *mpy + inc) % mod ; // **fmode(seed *mpy + inc, mod)**

then **get one random number** between 0~1 by using: **seed/mod**

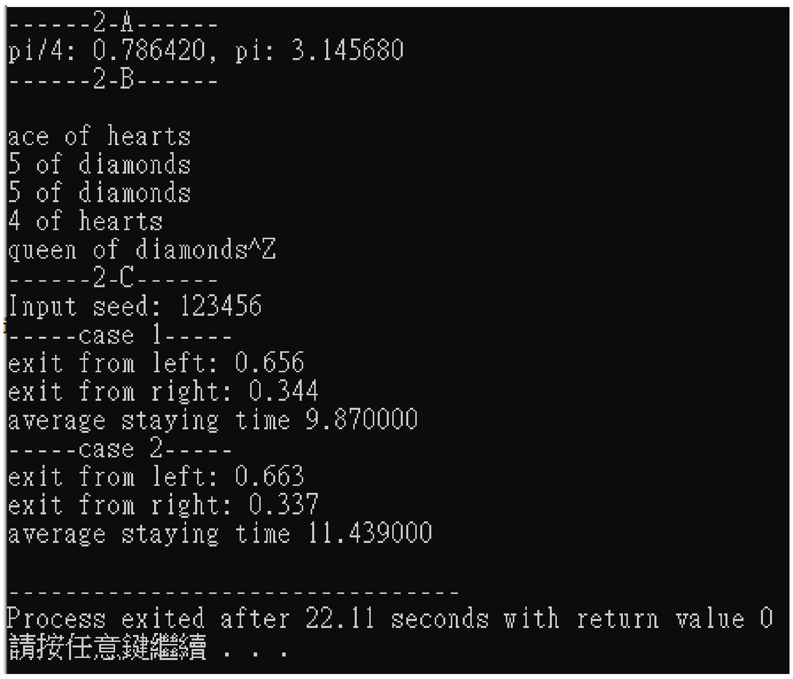
B. Write a program that displays the name of a card **randomly chosen** from a complete deck of 52 playing cards. Each card consists of a rank (ace, 2,3,4,5,6,7,8,9,10,jack, queen, king) and suit (clubs, diamond, hearts, spades).
You should refer to following program to let user continuously press ENTER and see a random result until typing CTRL+D / CTRL+Z.

```
char t;
while(scanf("%c", &t) != EOF)
{
        …
}
```

C. A dog is lost in a tunnel at node 0 (see diagram). It can move one node at one time in either direction right or left with equal probability (1 = right, 2 = left). When the dog hits L2 however, a force always propels it directly to node L4.
   The dog escapes from the tunnel when it either hits L5 or R4.
   Restart the dog at node 0 for **one thousand times**, please write a program to answer following questions.
   (1) What are the two odds that the dog escapes from L5 and R4?
       How long on average does the dog stay in the tunnel? (each node takes one minute to cover)
   (2) Do (1) and again, but now let node L2 propels the dog to L4 only when traveling in a left direction. If node L2 is reached when traveling to the right, the force has no effect.



Sample output:



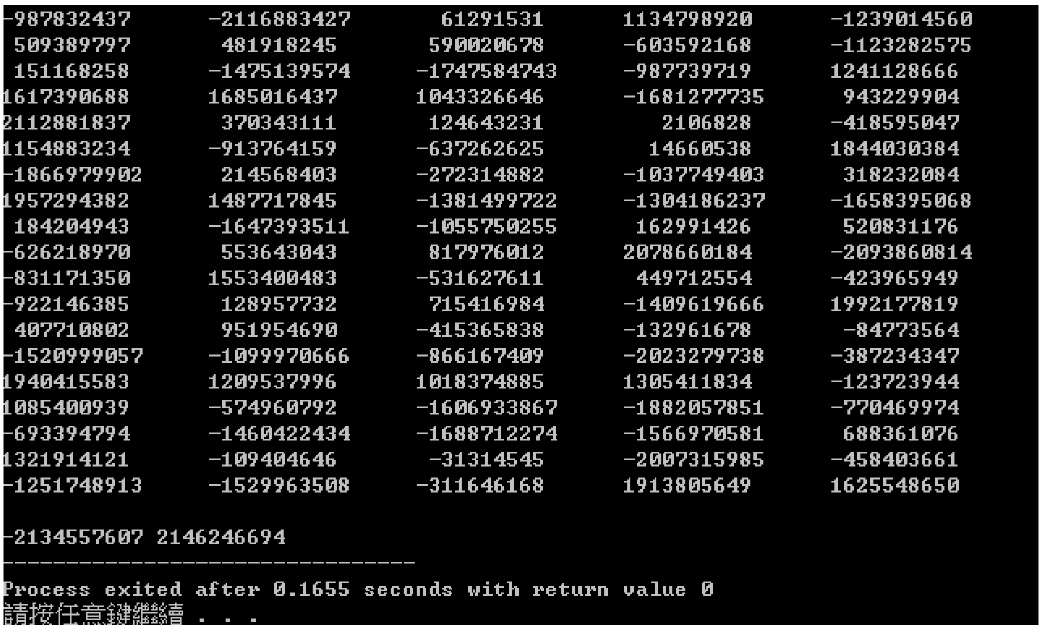
```
------2-A------
pi/4: 0.786420, pi: 3.145680
------2-B------

ace of hearts
5 of diamonds
5 of diamonds
4 of hearts
queen of diamonds^Z
------2-C------
Input seed: 123456
-----case 1-----
exit from left: 0.656
exit from right: 0.344
average staying time 9.870000
-----case 2-----
exit from left: 0.663
exit from right: 0.337
average staying time 11.439000

--------------------------------
Process exited after 22.11 seconds with return value 0
請按任意鍵繼續 . . .
```

3. Generate 1000 random numbers with uniform distribution in the **range of integer (-2147483678~2147483647).** Also print out the max and min number in the 1000 numbers generated.

   Hint1: the range of integer is from -2^{31} to 2^{31}-1, and rand() generate random numbers from 0 to 32767 (2^{15}-1)

   Hint2: try and understand **rand() * (RAND_MAX + 1) + rand()**

   Hint3: try replacing your rand() into RAND_MAX and observe whether the result is INT_MAX or INT_MIN or neither. If neither, your code probably goes wrong.

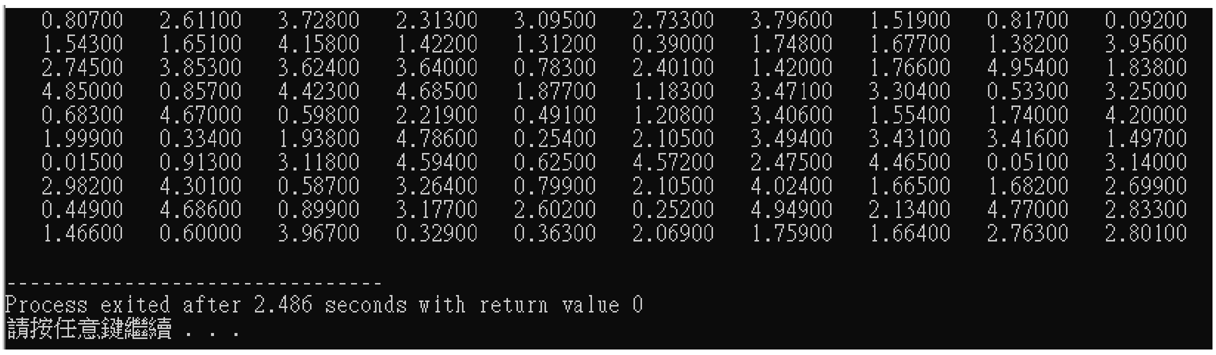


```
-987832437      -2116883427      61291531       1134798920      -1239014560
 509389797       481918245      590020678       -603592168      -1123282575
 151168258      -1475139574    -1747584743      -987739719       1241128666
1617390688      1685016437     1043326646      -1681277735       943229904
2112881837       370343111      124643231          2106828      -418595047
1154883234      -913764159     -637262625         14660538      1844030384
-1866979902      214568403     -272314882      -1037749403       318232084
1957294382      1487717845    -1381499722      -1304186237     -1658395068
 184204943      -1647393511    -1055750255       162991426       520831176
-626218970       553643043      817976012       2078660184     -2093860814
-831171350      1553400483     -531627611        449712554      -423965949
-922146385       128957732      715416984      -1409619666      1992177819
 407710802       951954690     -415365838       -132961678       -84773564
-1520999057     -1099970666    -866167409      -2023279738      -387234347
1940415583      1209537996     1018374885       1305411834      -123723944
1085400939      -574960792    -1606933867      -1882057851      -770469974
-693394794      -1460422434   -1688712274      -1566970581       688361076
1321914121      -109404646      -31314545      -2007315985      -458403661
-1251748913     -1529963508    -311646168       1913805649      1625548650

-2134557607 2146246694
--------------------------------
Process exited after 0.1655 seconds with return value 0
請按任意鍵繼續 . . .
```

4. **Understand the following method for generating random numbers**

```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
int main()
{
    srand(time(NULL));
    int range=10000, x;
    int bound=(RAND_MAX+1)/range;
    do{
        x=rand();
    }while(x>=bound*range);
    printf("%d\n",x%range);
    return 0;
}
```

Using similar concept, Please generate 100 random numbers in the range from [0, 5], the numbers should be accurate to 3 digits after decimal point, and they must be uniformly distributed. For example, 1.000 and 4.777 each is generated in the same probability.



```
   0.80700   2.61100   3.72800   2.31300   3.09500   2.73300   3.79600   1.51900   0.81700   0.09200
   1.54300   1.65100   4.15800   1.42200   1.31200   0.39000   1.74800   1.67700   1.38200   3.95600
   2.74500   3.85300   3.62400   3.64000   0.78300   2.40100   1.42000   1.76600   4.95400   1.83800
   4.85000   0.85700   4.42300   4.68500   1.87700   1.18300   3.47100   3.30400   0.53300   3.25000
   0.68300   4.67000   0.59800   2.21900   0.49100   1.20800   3.40600   1.55400   1.74000   4.20000
   1.99900   0.33400   1.93800   4.78600   0.25400   2.10500   3.49400   3.43100   3.41600   1.49700
   0.01500   0.91300   3.11800   4.59400   0.62500   4.57200   2.47500   4.46500   0.05100   3.14000
   2.98200   4.30100   0.58700   3.26400   0.79900   2.10500   4.02400   1.66500   1.68200   2.69900
   0.44900   4.68600   0.89900   3.17700   2.60200   0.25200   4.94900   2.13400   4.77000   2.83300
   1.46600   0.60000   3.96700   0.32900   0.36300   2.06900   1.75900   1.66400   2.76300   2.80100


--------------------------------
Process exited after 2.486 seconds with return value 0
請按任意鍵繼續 . . .
```