追加課題 2a（数当てゲームを作成する）

ランダムな 0 から 10 までの数字を作成し，その数字をキーボードを用いて当てるゲームを作成せよ．また，入力された数値に対し，大きい，小さい，当たりであるかを出力せよ．最後に，何回で当てられたかを表示せよ．

- ランダムな 10 までの数は rand 関数を用いて作成せよ．
  - ただし，rand 関数は 0 から RAND_MAX(2147483647) の範囲の整数を生成する
- キーボードの入力は scanf 関数を用いよ．

提出用のファイル名は `problem_2a.c` とすること．

ゲームの出力例は以下の通りである．

```
数字を入力してください
1
1 よりも大きい数です．数字を入力してください
8
1 よりも小さい数です．数字を入力してください
6
当たりです．3 回で当てられました．
```

追加課題 2b（積が最大となる組み合わせ）

配列の長さが n の 2 つの配列 a,b をキーボードから入力し，配列の要素の中で `a[i]*b[j]` が最大となるような添え字 i,j を 1 つ求めよ．複数ある場合もあるが，その場合はどの候補を返しても良いとする．なお，プログラムの作成の際には関数

```
void find_max_pair(int a[], int b[], int n, int *index_a, int *index_b)
```

を作成し，`a[i]*b[j]` が最大となるような添え字 i,j を index_a と index_b に格納し，`main` 関数でどちらの添え字か分かるように表示し，`a[i]*b[j]` の値も表示せよ．プログラムの作成には以下の手順で行うこと．

- 最初にデータ数 n を `main` 関数内で scanf 関数を用いて入力せよ．(n の最大は 8 としてよい．)
- 次に `main` 関数内で配列 a,b という順番で整数データを入力せよ．
- 最後に `main` 関数内で `find_max_pair` 関数を呼び出し，`main` 関数内で最大となるような添え字を分かるように表示し，`a[i]*b[j]` の値も表示せよ．提出用のファイル名は `problem_2b.c` とすること．

【注意】

- n=2,a={-2,3}, b={-2,3}の場合，最大となる a と b の組み合わせは a[1] と b[1] である．
- n=2,a={-3,-2}, b={-3,-2}の場合，最大となる a と b の組み合わせは a[0] と b[0] である．
- n=2,a={-2,-1}, b={1,2}の場合，最大となる a と b の組み合わせは a[1] と b[0] である．
- n=2,a={2,4}, b={-2,-1}の場合，最大となる a と b の組み合わせは a[0] と b[1] である．

追加課題 2c（秘書問題）

秘書を 1 人雇いたいとする．そして，m 人が応募してきている．最初の m/e 人 (小数点は切り捨て) の応募者を雇わず，それ以降に面接した応募者がそれまでより良いと判断したら採用する，という戦略をとる．もし，最後まで採用しなかった場合は最後の応募者を採用する．このとき，n 回試行して最も能力の高い秘書を雇った確率を求めよ．提出用のファイル名は `problem_2c.c` とすること．

ただし，$e = 2.7182818284$ とし，秘書の能力は課題 7a で作成した make_rand_r 関数で与えた値とする．また大きい値の方が優秀であるとする．プログラムは以下を参考にせよ．

Step 1 応募してくる人数である m を scanf 関数で入力する

Step 2 試行回数 n を scanf 関数で入力する

Step 3 make_rand_r 関数で応募してきた人の能力を決定する

Step 4 この戦略で選んだ秘書と最も優秀な秘書かどうかを調べる

Step 5 Step 3- 4 を n 回試行して最も能力の高い秘書を雇った確率を求める

追加課題 2d（3 角形の面積）

3-次元空間上の 3 点 $A = (a_1, a_2, a_3)$, $B = (b_1, b_2, b_3)$,$C = (c_1, c_2, c_3)$ で囲まれた面積を計算せよ．提出用のファイル名は `problem_2d.c` とすること．

3-次元空間上の 3 点 A, B,C で囲まれた面積は，ベクトル表記を用いて以下で表せる．

$$\frac{1}{2}(\sqrt{|\overrightarrow{AB}|^2|\overrightarrow{AC}|^2 - (\overrightarrow{AB} \cdot \overrightarrow{AC})^2}$$

追加課題 2e（コッホ曲線を計算する）

コッホ曲線とは以下の操作を繰り返しながら描かれるフラクタル図形である．

Step 1 与えられた線分 p1,p2，を 3 等分する．三等分する 2 点を a と b とする

Step 2 中央の線分 a,b を 1 辺とする正三角形 a,b,c を描く (このとき，線分 a,b には線を引かない．図 2 を参照)

- 線分 p1,a，線分 a,b，線分 b,c，線分 c,p2 の得られた 4 つの線分に対しても同じ操作をする ($n = 2$)
- 得られた 16 つの線分に対しても同じ操作を繰り返す ($n = 3$)
- 指定の n まで繰り返す

指定の n まで繰り返し，そのすべての座標を出力するプログラムを作成せよ． このとき，

```
void Koch(double p1_x,double p1_y,double p2_x,double p2_y,int n)
```

Koch 関数の中で Koch 関数を呼び出す (再帰) ようにプログラムを作成せよ．また，gnuplot を用いて画像を作成するので，線を繋ぐ順番に座標を出力すること．

入力は n および p1,p2 とする．次にリダイレクトを用いて座標ファイルを作成せよ．

```
./problem_2e > problem_2e.txt
```

problem_2e.plot を以下のように作成し，png ファイルを作成せよ．

```
set size square
set xrange [0:1]
set yrange [-0.5:0.5]
set xzeroaxis
set yzeroaxis
plot "problem_2e.txt" with lines
set terminal png size 800, 800
set output "problem_2e.png"
replot
```

• n=6，p1 の座標が (0,0)，p2 の座標が (1,0) の場合の png ファイルを作成し，提出せよ．提出用のファイル名は `problem_2e.png` とすること．

n=1，p1 の座標が (0,0)，p2 の座標が (1,0) の場合は以下のような出力となる．

```
0.000000 0.000000
0.333333 0.000000
0.500000 0.288675
0.666667 0.000000
1.000000 0.000000
```

n=2，p1 の座標が (0,0)，p2 の座標が (1,0) の場合は以下のような出力となる．

```
0.000000 0.000000
0.111111 0.000000
0.166667 0.096225
0.222222 0.000000
0.333333 0.000000
0.388889 0.096225
0.333333 0.192450
0.444444 0.192450
0.500000 0.288675
0.555556 0.192450
0.666667 0.192450
0.611111 0.096225
0.666667 0.000000
0.777778 0.000000
0.833333 0.096225
0.888889 0.000000
1.000000 0.000000
```

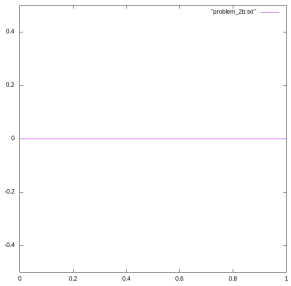

図 1: $n = 0$ のコッホ曲線

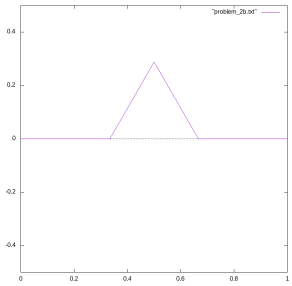

図 2: $n = 1$ のコッホ曲線

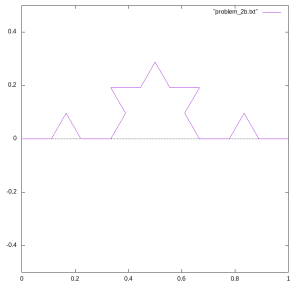

図 3: $n = 2$ のコッホ曲線

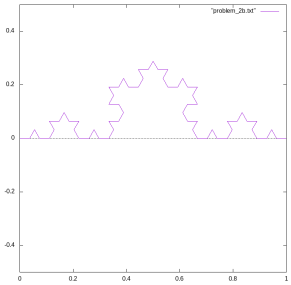

図 4: $n = 3$ のコッホ曲線

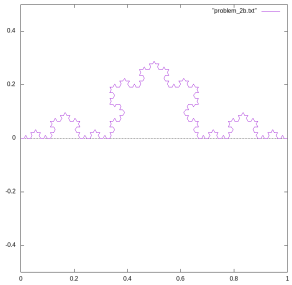

図 5: $n = 4$ のコッホ曲線

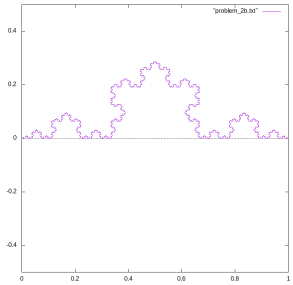

図 6: $n = 5$ のコッホ曲線

追加課題 2f（行列の入力と出力を作る）

二次元配列を用いて行列の入出力用の関数 input_matrix 関数と print_matrix 関数を作成せよ．入力は scanf 関数を用いよ．提出用のファイル名は `problem_2f.c` とすること．

C 言語では一次元配列を用いて作成することが多い．

```
void input_matrix(int rows, int columns, double *mat){
}
```

高速な行列演算ルーチンの BLAS などは一次元配列でのインターフェースとなっており，基本的には一次元配列で作成することを推奨する．二次元配列にすると，

```
#define COL 3
void input_matrix(int rows, double mat[][COL]){
}
```

のように書く必要がある．二次元配列を用いるのであれば，C99 の可変長配列を用いて書く方法もある．

```
#include<stdio.h>
#define N 100
#define M 100
```

```
void input_matrix(int rows, int columns,double mat[rows][columns]){

}

void print_matrix(int rows,int columns,double mat[rows][columns]){

}

int main(void){

        int n,m;
        double mat[N][M];
        scanf("%d",&n);
        scanf("%d",&m);
        input_matrix(n,m,mat);
        print_matrix(n,m,mat);
}
```

追加課題 2g（行列積のプログラムを作成する）

行列積を行う matrix_multiplication 関数を作成せよ．引数は課題 2f で作成したプログラムに合わせて自分で決定せよ．提出用のファイル名は problem_2g.c とすること．