Lab04/02

1. **Play Cards! (I)**

   This question is almost the same as your HW3-6. Instead of using words to represent the suits, here we use the symbols.

   On our workstation we can use Unicode for printing symbols. For example, printing the string “\xE2\x99\xA0” shows a blade on the screen.

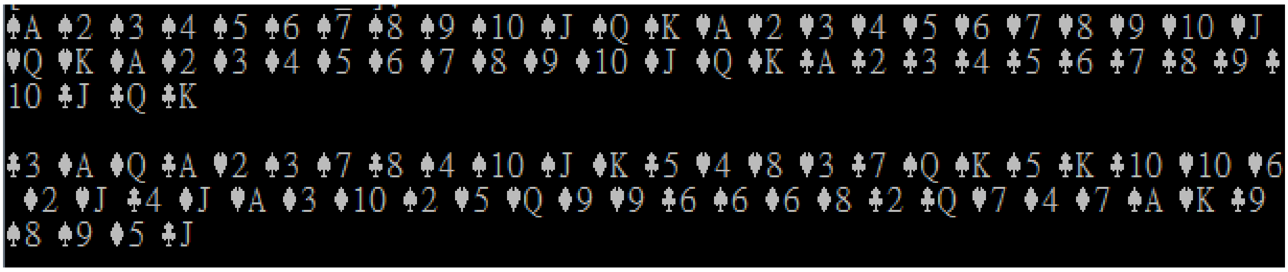
   Please define following functions.

   - void displayCards(...): Show each card’s number and suit.
   - void getCards(...): Create a deck of cards
   - void swapCards(...): Swap two cards
   - void shuffleCards(...): Shuffle the cards.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>

// Definition of string array to store poker suits
// Blade: \xE2\x99\xA0, Heart: \xE2\x99\xA5
// Diamond: \xE2\x99\xA6, Club: \xE2\x99\xA3
……

// Definition of string array to store poker number including A, 2, 3, …, K
……

// Declaration of structure Card
……

int main(){
    srand(time(NULL));

    struct Card *cards;

    getCards(...);
    displayCards(...);

    shuffleCards(...);
```

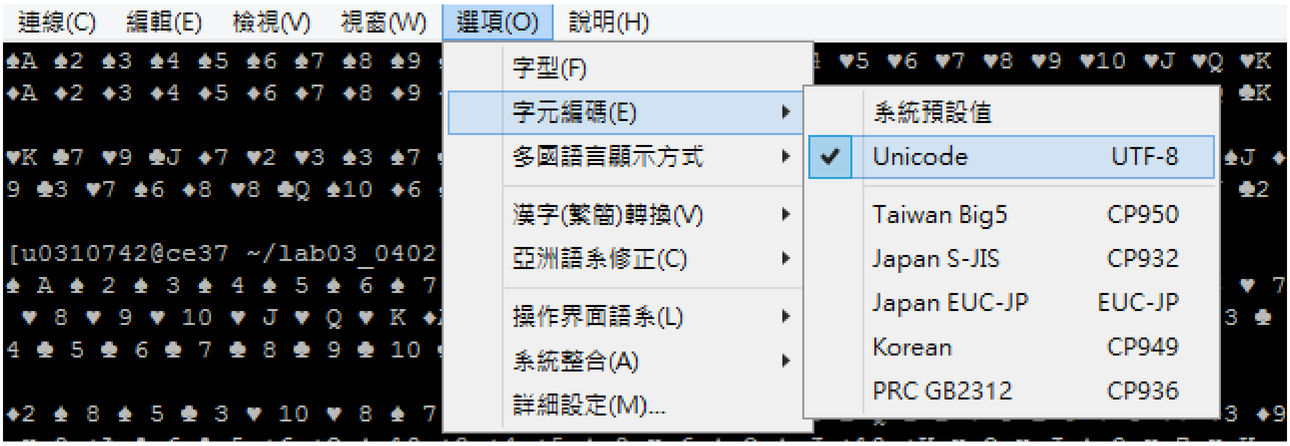
```
        displayCards(...);

        return 0;
}
```

The result is as follows.

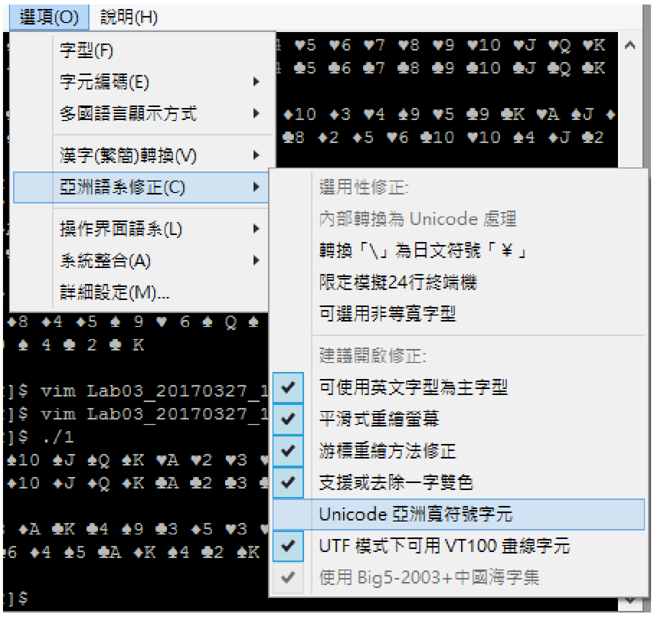


```
♠A ♠2 ♠3 ♠4 ♠5 ♠6 ♠7 ♠8 ♠9 ♠10 ♠J ♠Q ♠K ♥A ♥2 ♥3 ♥4 ♥5 ♥6 ♥7 ♥8 ♥9 ♥10 ♥J
♥Q ♥K ♦A ♦2 ♦3 ♦4 ♦5 ♦6 ♦7 ♦8 ♦9 ♦10 ♦J ♦Q ♦K ♣A ♣2 ♣3 ♣4 ♣5 ♣6 ♣7 ♣8 ♣9 ♣
10 ♣J ♣Q ♣K

♣3 ♦A ♦Q ♣A ♥2 ♠3 ♠7 ♣8 ♠4 ♠10 ♠J ♦K ♣5 ♥4 ♥8 ♥3 ♣7 ♠Q ♠K ♠5 ♣K ♣10 ♥10 ♥6
 ♦2 ♥J ♣4 ♦J ♥A ♦3 ♦10 ♠2 ♥5 ♥Q ♦9 ♥9 ♣6 ♠6 ♦6 ♦8 ♣2 ♣Q ♥7 ♦4 ♦7 ♠A ♥K ♣9
♠8 ♠9 ♦5 ♣J
```

如花色無法正常顯示，勾選使用”Unicode”編碼



如花色後有異常空格，取消”Unicode 亞洲寬符號字元”



2. Complex Number

A complex number can be represented by real part and imaginary part, i.e. $a+bi$. Another way of encoding points in the complex plane is to use polar form $r(\cos\theta + i\sin\theta)$ or $re^{i\theta}$. Therefore, we can represent a complex by a structure shown as follows. (You can refer to Wikipedia to get more information about complex number.

(**http://en.wikipedia.org/wiki/Complex_number#Polar_form**)

Hint: you can use atan2 to obtain angle $\varphi = \mathrm{atan2}(\mathrm{imaginary}, \mathrm{real})$, the unit is radian.

Please construct a complex number type by using a nested structure. The format of the complex number is as follows. Please declare a struct type by following table which r and i stand for real and image parts.

Complex			
r	i	polar	
float	float	mag	arg
		double	double

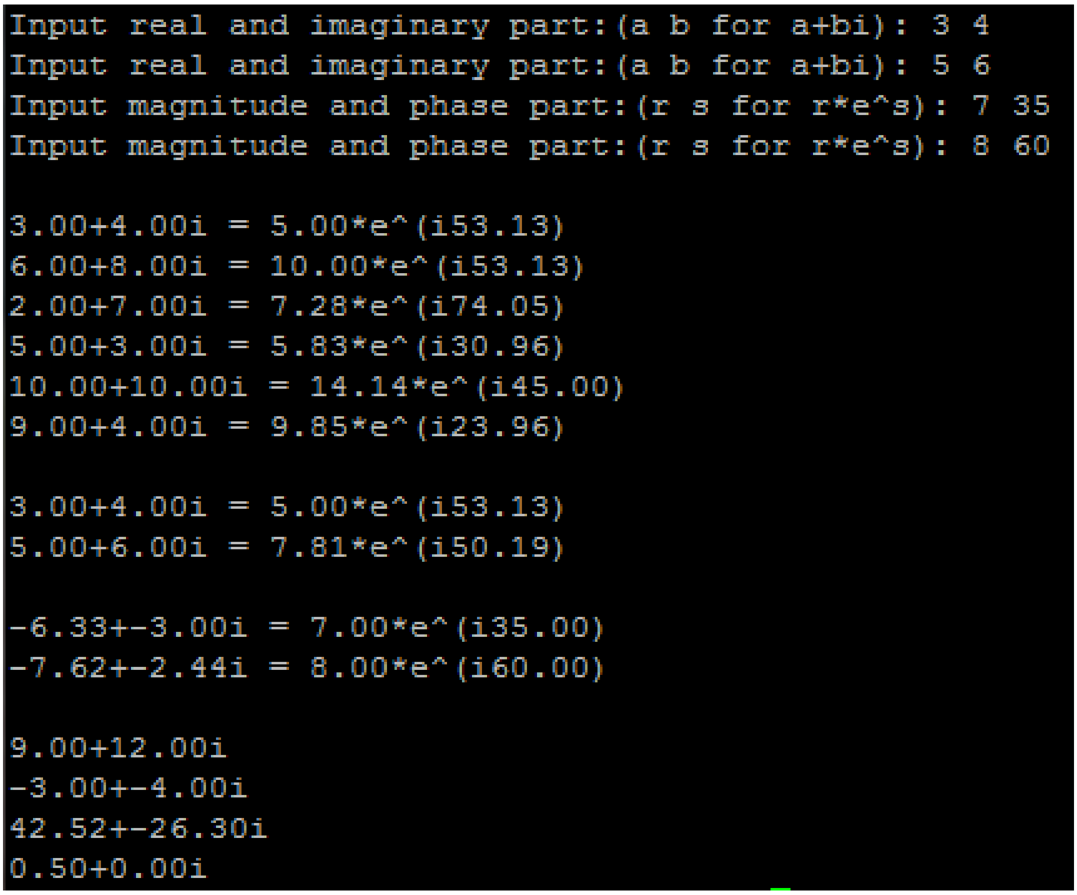
Complete the following functions.

```
void C_print (const Complex z)
{
        // printout (a + bi)
}

void P_print (const Complex z)
{
        // printout (r * e^ (i sita)◦ )
}

void convertC2P (Complex *z)
{
    // complete polar form transform
    // Hint: use atan2(), unit is degree
}

void convertP2C (Complex *z)
{
    // complete general form transform
    // Hint: use cos, sin
}

void readInGeneral (Complex &z)
{
    // read real part and imaginary part from user
    // and then convert to polar form by convertC2P function
}

void readInPolar (Complex &z)
{
    // read magnitude part and phase part from user
    // and then convert to general form by convertP2C function
}

Complex C_add (const Complex z1, const Complex z2)
{
    // return (z1 + z2)
}

Complex C_sub (const Complex *z1, const Complex *z2)
{
   // return (z1 - z2)
}

Complex C_mul (const Complex &z1, const Complex &z2)
{
    // return (z1 * z2)
}

Complex C_div (const Complex *z1, const Complex *z2)
{
    // return (z1 / z2)
}
```

You can use following code to verify your functions implemented in problem 2.

```
int main(void)
{
    Complex a[N] = {{3.0, 4.0}, {6.0, 8.0}, {2,7}, {5, 3}, {10, 10}, {9, 4}};
    Complex b[2]; // for general form input
    Complex c[2]; // for polar form input
    int i;

    // read data from user into b[i] and c[i]
    for (i=0; i<2; i++)
        readInGeneral(b[i]);

    for (i=0; i<2; i++)
        readInPolar(c[i]);
    printf("\n");

    // transform complex number a[i] into polar form
    for (i=0; i<N; i++) {
        convertC2P(&a[i]);
        C_print(a[i]);
        printf(" = ");
        P_print(a[i]);
        printf("\n");
    }
    printf("\n");

    // print out a[i] and b[i] and c[i] with general form and polar form
    for (i=0; i<2; i++) {
        C_print(b[i]);
        printf(" = ");
        P_print(b[i]);
        printf("\n");
    }
    printf("\n");

    for (i=0; i<2; i++) {
        C_print(c[i]);
        printf(" = ");
        P_print(c[i]);
        printf("\n");
    }
    printf("\n");

    // verify add, sub, mul, and div functions here
    C_print(C_add(a[0], a[1]));
    printf("\n");
    C_print(C_sub(&a[0], &a[1]));
    printf("\n");
    C_print(C_mul(a[0], a[1]));
    printf("\n");
    C_print(C_div(&a[0], &a[1]));
    printf("\n");

    return 0;
}
```

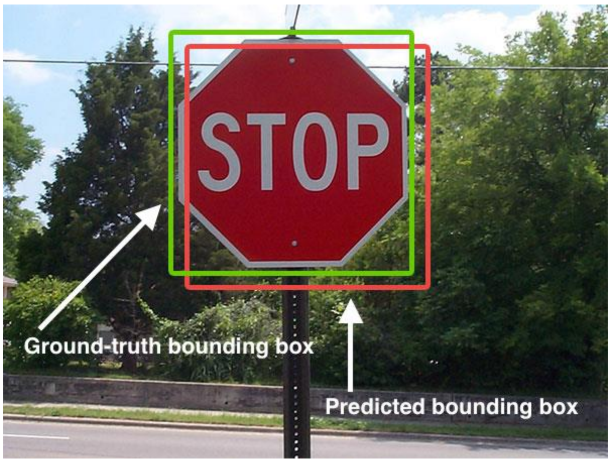
The example result is as follows.



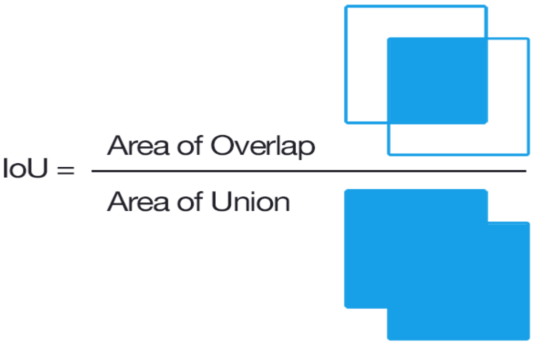
```
Input real and imaginary part:(a b for a+bi): 3 4
Input real and imaginary part:(a b for a+bi): 5 6
Input magnitude and phase part:(r s for r*e^s): 7 35
Input magnitude and phase part:(r s for r*e^s): 8 60

3.00+4.00i = 5.00*e^(i53.13)
6.00+8.00i = 10.00*e^(i53.13)
2.00+7.00i = 7.28*e^(i74.05)
5.00+3.00i = 5.83*e^(i30.96)
10.00+10.00i = 14.14*e^(i45.00)
9.00+4.00i = 9.85*e^(i23.96)

3.00+4.00i = 5.00*e^(i53.13)
5.00+6.00i = 7.81*e^(i50.19)

-6.33+-3.00i = 7.00*e^(i35.00)
-7.62+-2.44i = 8.00*e^(i60.00)

9.00+12.00i
-3.00+-4.00i
42.52+-26.30i
0.50+0.00i
```

3. Intersection over Union(IoU) (Using Graphics Mode on Windows)

Intersection of Union is one method for determining the accuracy of object detection. In this field we often use rectangle to specify an object, for example: the red rectangle is determined by computer, and the green one is determined by experts. We can determine the accuracy of this detection result by IoU.



https://www.pyimagesearch.com

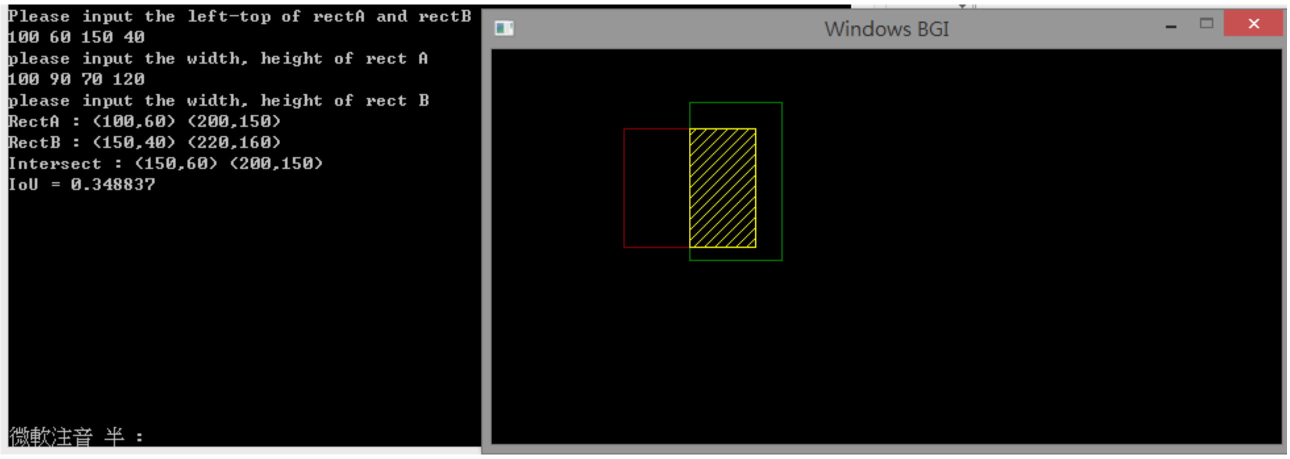


Please do the following tasks:

1. Construct a structure point and then use point to construct a structure Rect which represents rectangle.
2. Let users input 2 rectangles(by the form you decide).
3. Output the coordinates of these rectangles
4. Draw the 2 rectangles.
5. Compute the IoU of these 2 rectangles.
6. Repeat 2~5 until user inputs EOF

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<time.h>

#define W 600
#define H 300

// Structure Declaration
……

int main()
{
    srand(time(NULL));
    initwindow(W,H);

    // Some variables definition
    ……

    while(……){

        setcolor(RED);
        rectangle(…);     //rectangle(x1,y1,x2,y2)
        setcolor(GREEN);
        rectangle(…);

        //find the IoU of a and b
        ...
        getch();
        cleardevice();
    }
    closegraph();
}
```

The output coule be like following (You don’t need to deal with the Yellow part):



```
Please input the left-top of rectA and rectB
100 60 150 40
please input the width, height of rect A
100 90 70 120
please input the width, height of rect B
RectA : (100,60) (200,150)
RectB : (150,40) (220,160)
Intersect : (150,60) (200,150)
IoU = 0.348837
```

Hint: You can find the left-top point if you have the left-top points of both red and green rectangles. So does the right-bot point.