Lab10

1. Homework Problem I

part 1.

Please use recursive function to compute the following:

(a). $S = \frac{1}{1*2} + \frac{1}{2*3} + \frac{1}{3*4} + \cdots + \frac{1}{n*(n-1)}$

(Think about when n becomes to 2 what the value of this term)

(b). $\pi = 4 * \frac{2}{3} * \frac{4}{3} * \frac{4}{5} * \frac{6}{5} * \frac{6}{7} * \frac{8}{7} * \cdots * \frac{2*n}{2*n+1} * \frac{2*(n+1)}{2*n+1}$

(You may consider ($\frac{2}{3}$ and $\frac{4}{3}$), ($\frac{4}{5}$ and $\frac{6}{5}$), ($\frac{6}{7}$ and $\frac{8}{7}$), ⋯ as one pair and let n=0 as 4)

(c). From (b) and using the constant M_PI in the math.h to find the approximated value of π till the error between M_PI and your approximated value π reduces to 0.005. Please print both M_PI and your approximated value π.

The program (a)&(b) should be able to execute repeatedly until user entering Ctrl-D/Ctrl-Z.

part2.

Write a program that inputs two numbers: x and y (data are all integers) in the main program and passes these two numbers (pass by value) to the recursive function: power that returns the x^y.

If y ≧ 0,

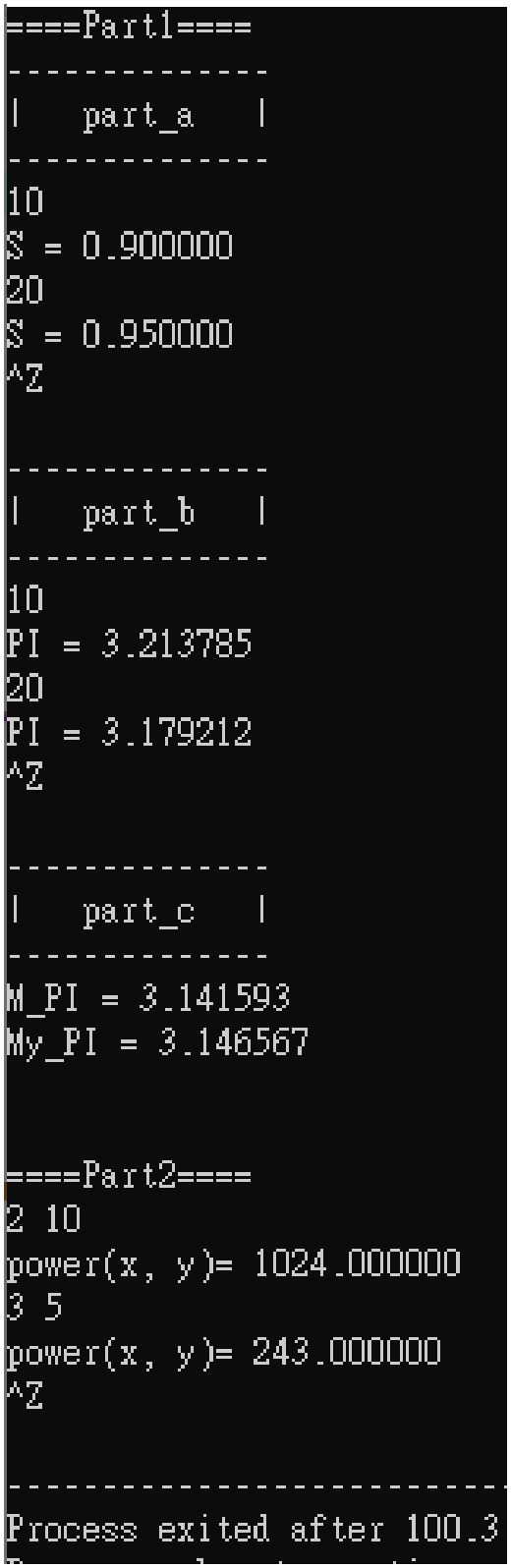
$$\text{power}(x, y) = \begin{cases} 1 & \text{if } y=0 \\ x & \text{if } y=1 \\ x * \text{power}(x, y-1) & \text{if } y>1 \end{cases}$$

If y<0,

$$\text{power}(x, y) = \frac{1}{\text{power}(x,-y)}$$

Print the result in the main program. The program should be able to execute repeatedly until user entering Ctrl-D/Ctrl-Z.

Sample Output:

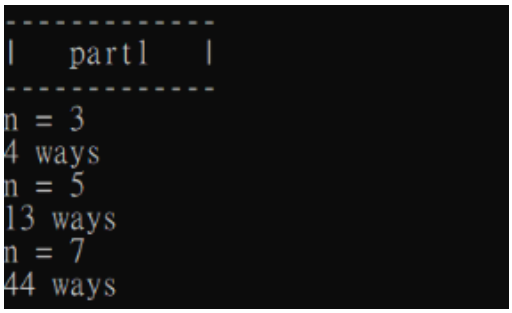


2. Homework Problem II

part1.

A robot can take steps of 1meter, 2 meters and 3 meters. Write a recursive function to evaluate the number of ways the robot can walk n meters. The program should be able to execute repeatedly until user entering Ctrl-D/Ctrl-Z.

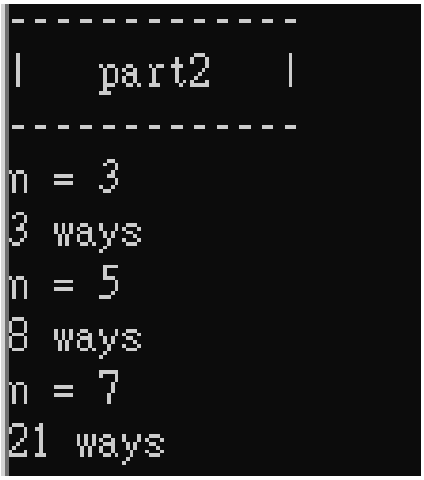
Sample output:



part2.

Suppose that we have a 2 × n rectangular board divided into 2n squares. Write a function that computes the number of ways to cover this board exactly by 1 × 2 dominoes. The program should be able to execute repeatedly until user entering Ctrl-D/Ctrl-Z.

Sample output:

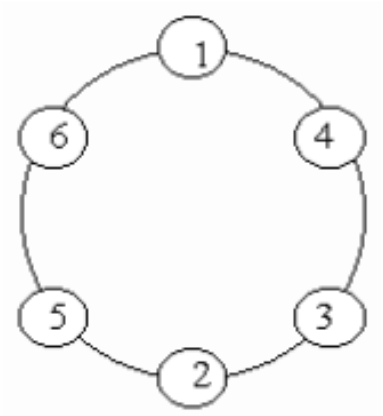


3.

Prime ring is a ring that is composed of n (even number, $0 < n \leq 16$) circles as shown in diagram. Put natural numbers 1, 2,⋯,n into each circle separately, and the sum of numbers in two adjacent circles should be a prime. Please write a program to find out all possible series that can composed a prime ring.

Note: The number of first circle should always be 1.
Hint: You can build up the prime map first to speedup.

Input:
6
8



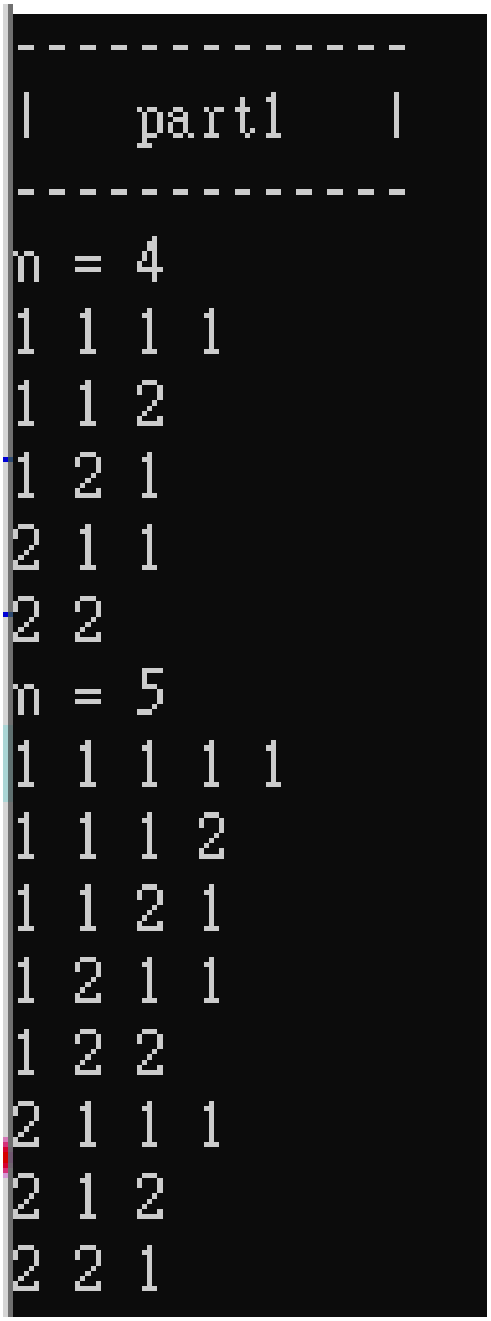
Output:
Case 1:
1 4 3 2 5 6
1 6 5 2 3 4
Number: 2

Case 2:
1 2 3 8 5 6 7 4
1 2 5 8 3 4 7 6
1 4 7 6 5 8 3 2
1 6 7 4 3 8 5 2
Number: 4

4.

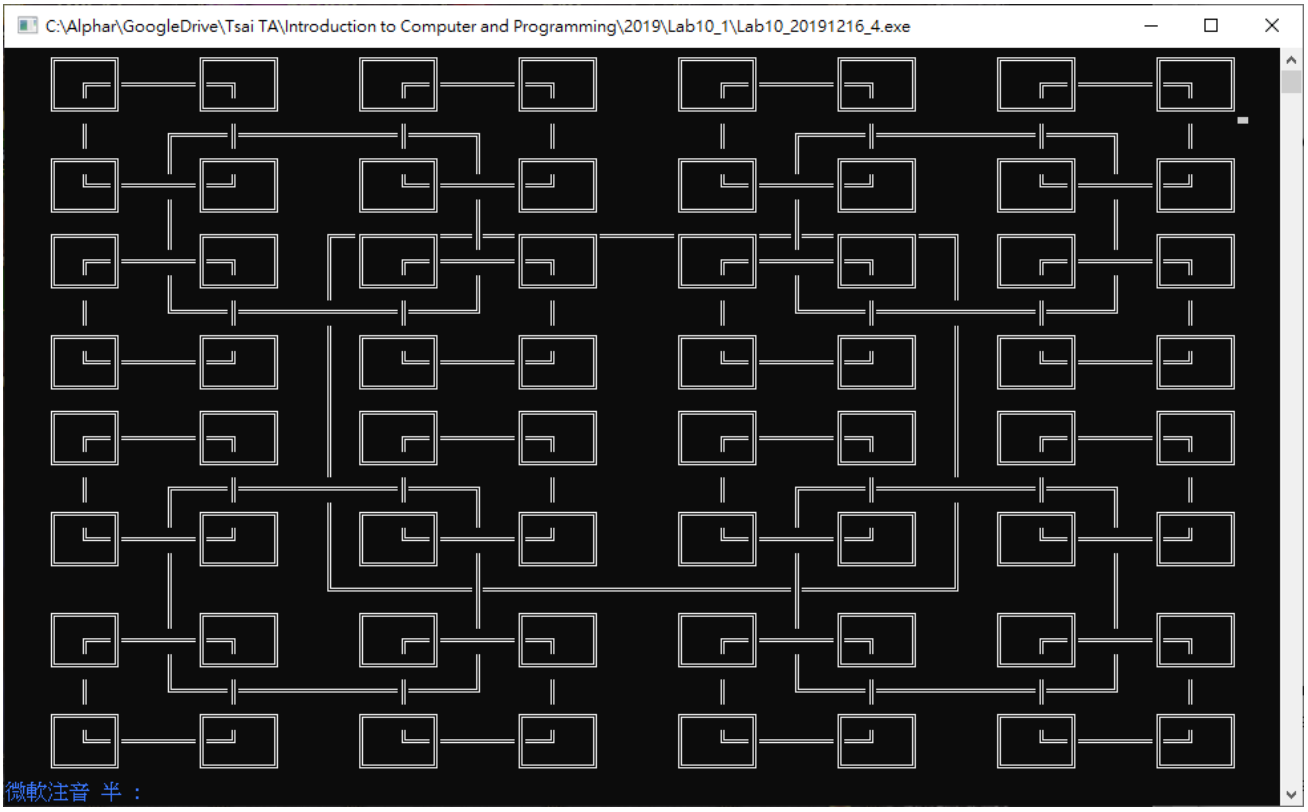
part1.

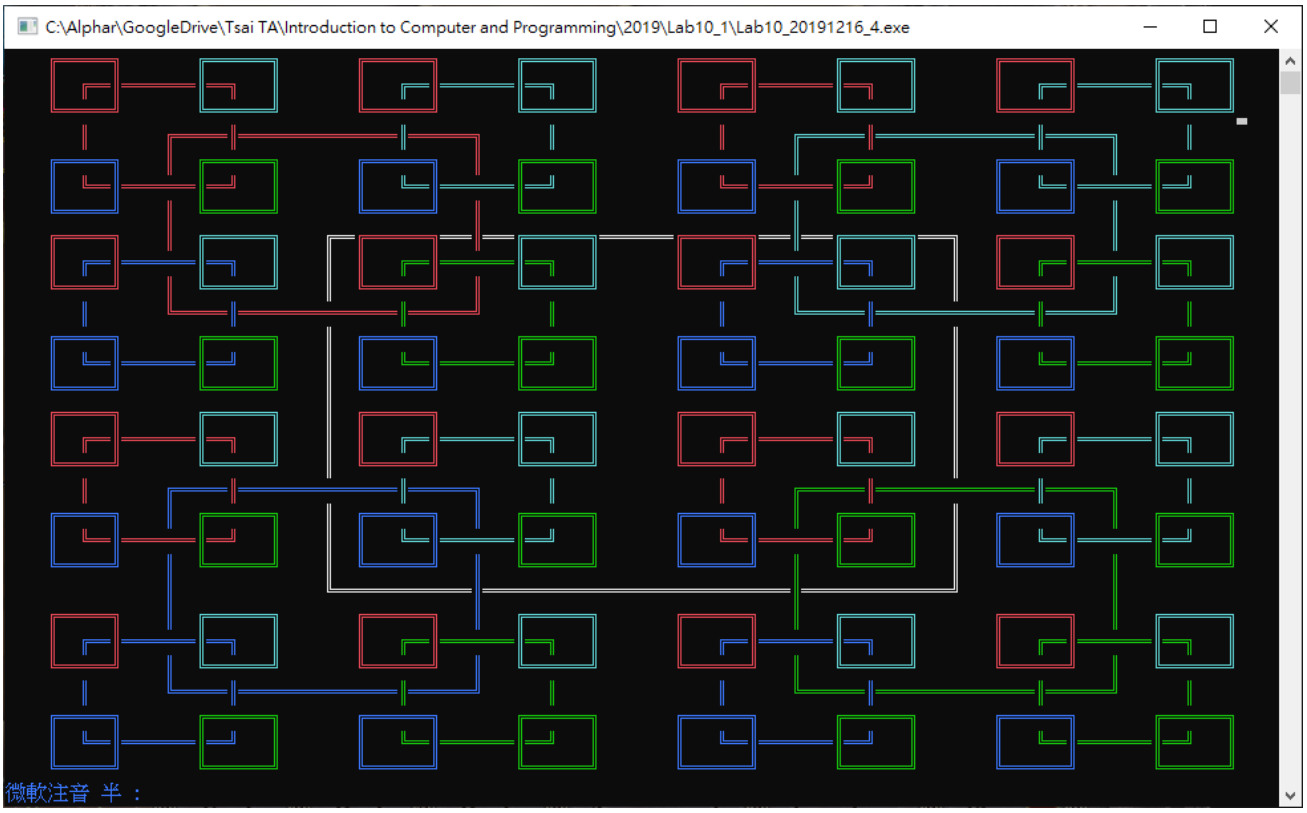
A robot can take steps of 1meter, 2 meters. Write a function that lists all of the ways that the robot can walk n meters. The program should be able to execute repeatedly until user entering Ctrl-D/Ctrl-Z.



part2.

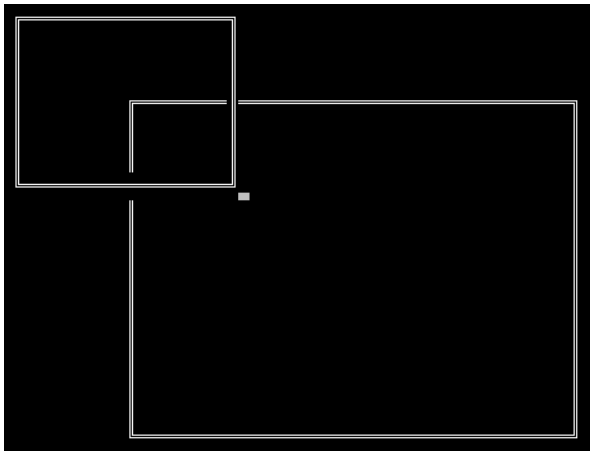
Write a program that uses recursive function void drawSquares(int n, int x1, int y1, int x2, int y2) to draw squares with one square on the corner of each square. i.e the results of this program after 4 recursions should look like these:



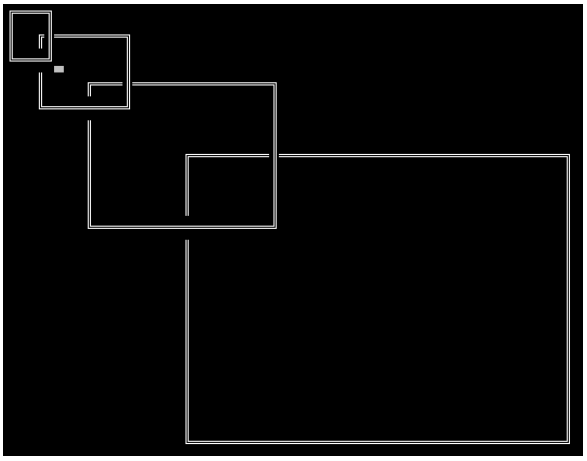


The algorithm would recursively draw up-left square, and then draw bottom-left square in the final recursion layer. For instance:

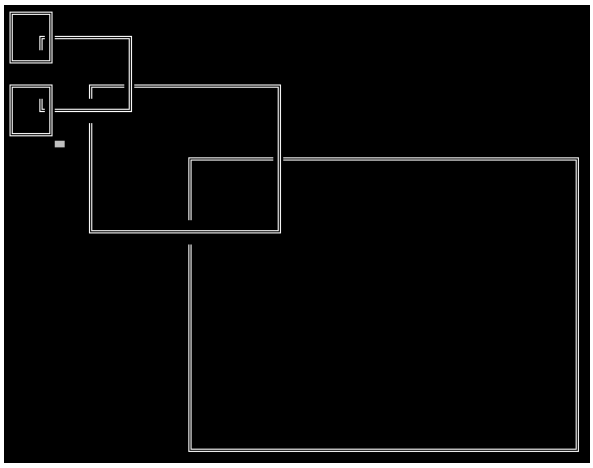
(1)



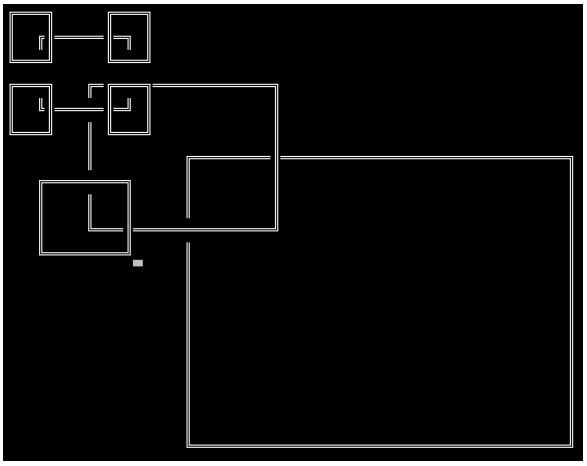
(2)



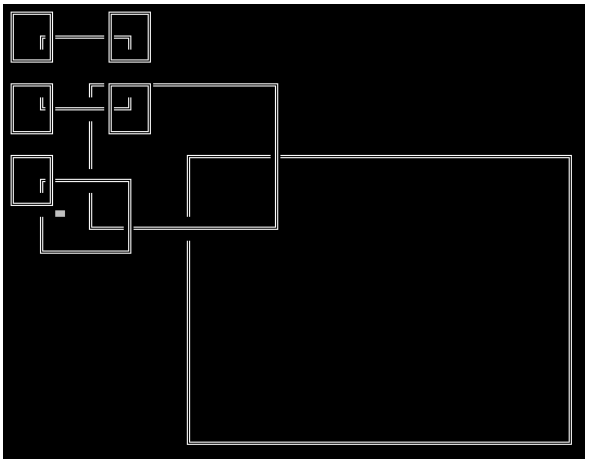
(3)



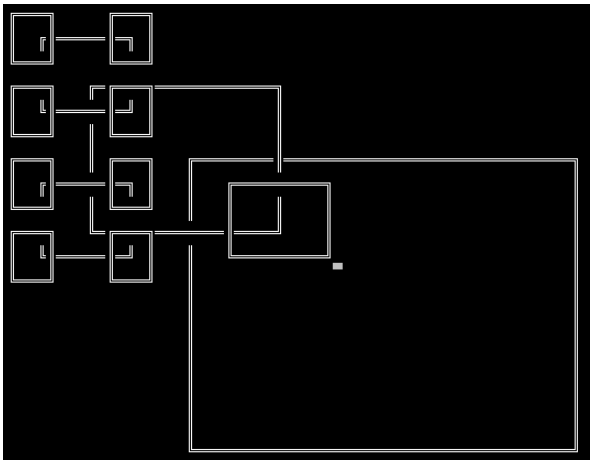
(4)



(5)



(6)



You only need to complete the following program:

```
#include <stdio.h>
#include <conio2.h>
#include <stdlib.h>
#define MAX 4 // define the max recursions, try different numbers!

void drawRect(int x1, int y1, int x2, int y2)
{    int x, y;
     // ╔
     gotoxy(x1,y1);
     printf("%c", 1);
     //=
    for (x=x1+1; x<x2; x++)
        printf("%c", 6);
     //╗
     gotoxy(x2,y1);
     printf("%c", 2);
     //║
     for (y=y1+1; y<y2; y++) {
        gotoxy(x1,y);
        printf("%c", 5);
        gotoxy(x2,y);
        printf("%c", 5);
     }
     //╚
     gotoxy(x1,y2);
     printf("%c", 3);
     //=
     for (x=x1+1; x<x2; x++)
        printf("%c", 6);
     //╝
     gotoxy(x2,y2);
     printf("%c", 4);
}

void drawSquares(int n, int x1, int y1, int x2, int y2)
{     //draw the rectangle
```

```
    drawRect( int((x2-x1)/4.+x1+0.5), int((y2-y1)/4+y1+0.5), int((x2-
x1)*3/4+x1+0.5),
                int((y2-y1)*3/4+y1+0.5));
    if (n < MAX)
    {
        //Something missed here…
    }
}

int main()
{
    system("chcp 950");
    textbackground(0);
    clrscr();
    textcolor(WHITE);
    drawSquares(1, 1, 1, 120, 30);  // try different sizes!
    return 0;
}
```