LAB 12/19

***Please use recursive function to solve the following problems.**

1. (a) Write a recursive function "long long int d2o(long long int)", which can convert the decimal to octal. Write a program to verify your function. Please let user input decimal number continuously until inputting 0 for stopping program.

Ex: 15→17 39→47 64→100 100→144

(b) Write a recursive function "long long int o2d(long long int)", which can convert the octal to decimal. Write a program to verify your function. Please let user input octal number continuously until inputting 0 for stopping program.

Ex: 17→15 47→39 100→64 144→100

2. The Hamming sequence *S* is a sequence of distinct integers in ascending order defined as follows:

(1). $1 \in S$

(2). If $x \in S$, then $2x \in S$, $3x \in S$, and $5x \in S$

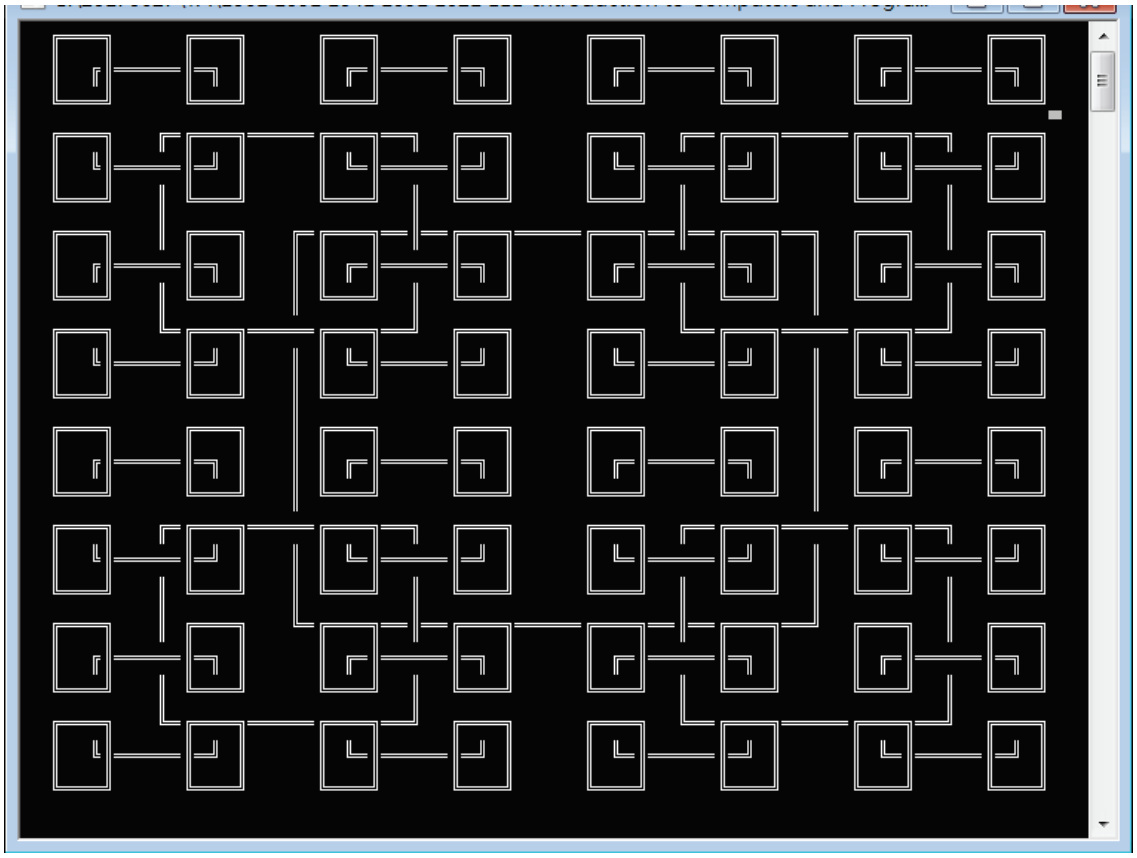
(3). Nothing else belongs to *S*

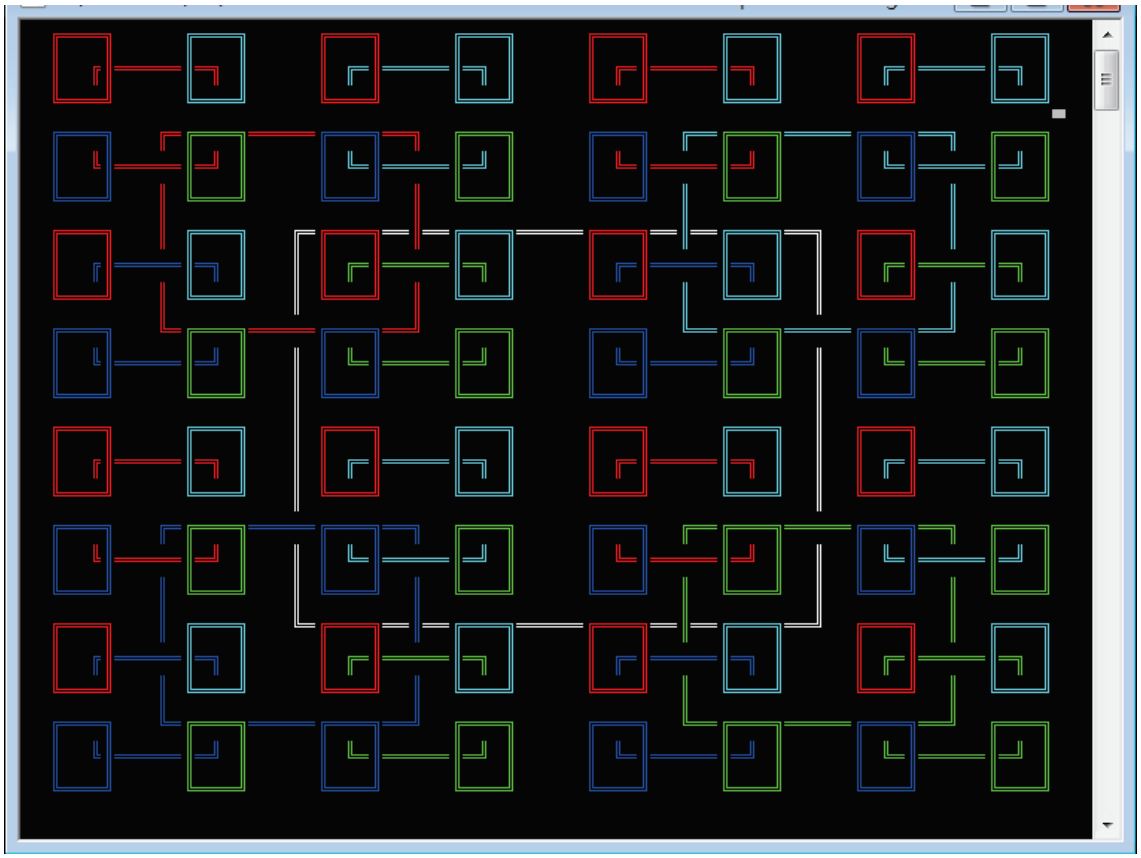
The first 20 elements of the Hamming sequence are
1,2,3,4,5,6,8,9,10,12,15,16,18,20,24,25,27,30,32,36

Write a recursive function "bool isHamming(int n)", which can determine whether n belongs to Hamming sequence. In this problem you are required to output the Kth number of Hamming sequence. Please let user input an integer K continuously until inputting 0 for stopping program.

Ex: 1→ 1 20→ 36 50→ 243

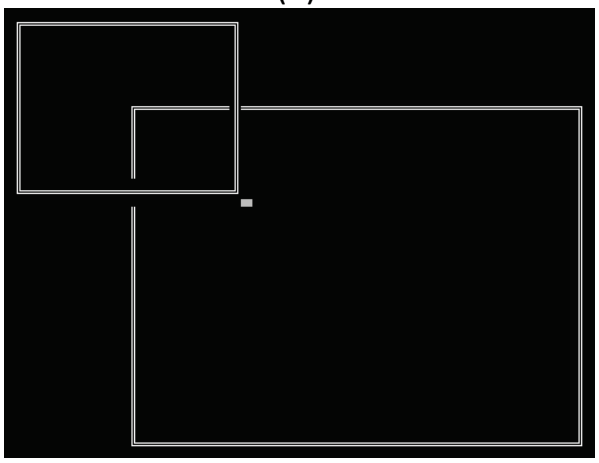
3. Write a program that uses recursive function void drawSquares(int n, int x1, int y1, int x2, int y2) to draw squares with one square on the corner of each square.
i.e the results of this program after 4 recursions should look like these:



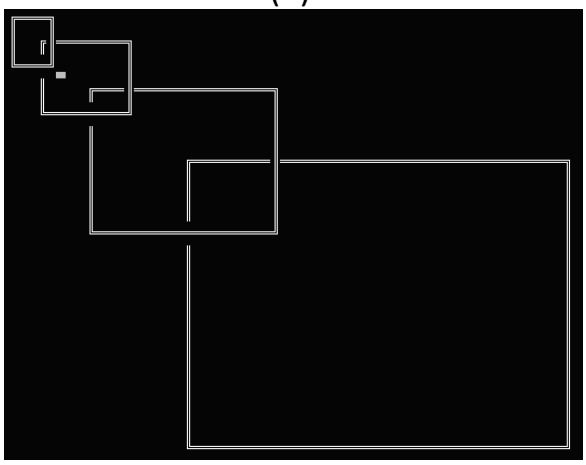


The algorithm would recursively draw up-left square, and then draw bottom-left square in the final recursion layer. For instance:

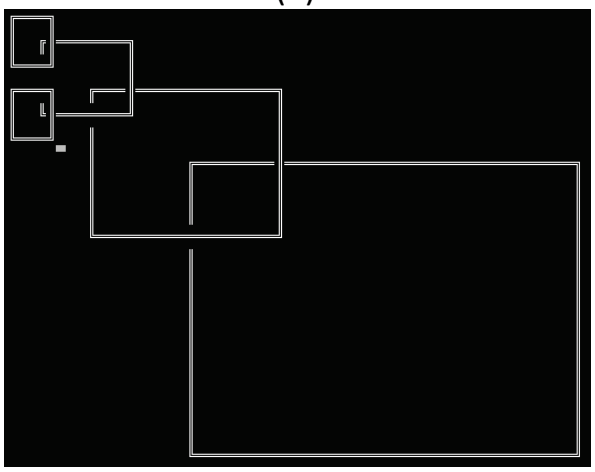
(1)



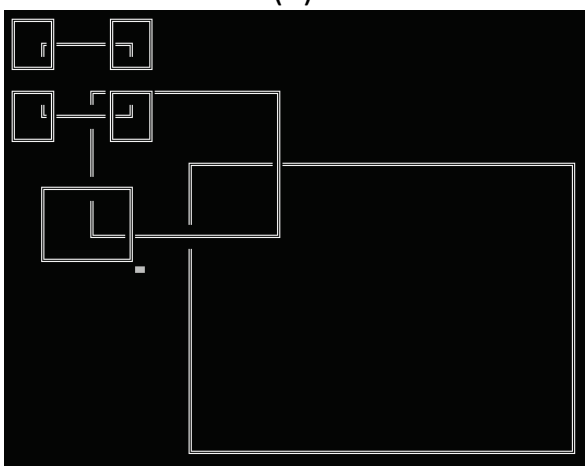
(2)



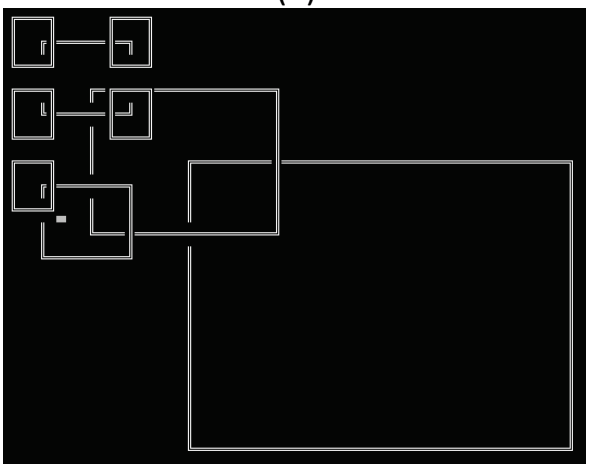
(3)



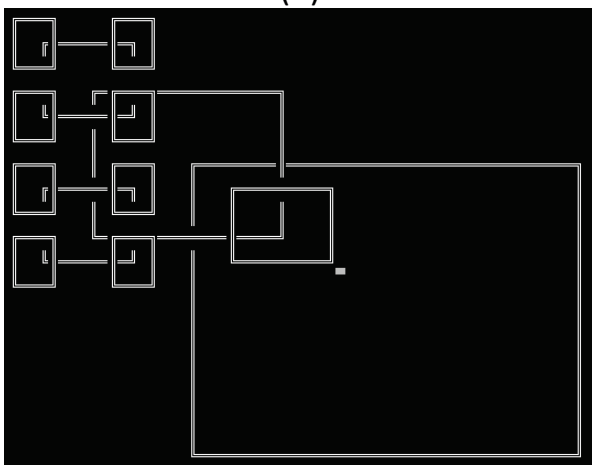
(4)



(5)



(6)



You only need to complete the following program:

```
#include <stdio.h>

#include <conio2.h>

#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()
{       textbackground(0);
        clrscr();
        textcolor(WHITE);
        drawSquares(1, 1, 1, 80, 25);    // try different sizes!
        return 0;
}
```