University of Zurich[UZH]

Department of Informatics
Database Technology Group
Prof. Dr. M. Böhlen

Informatik II
Exercise 2

Feb 25, 2019

Recursion

Task 1. The least common multiple LCM(x,y) is the smallest positive integer that is divisible by both given numbers x and y. For example, LCM(3,4) = 12, LCM(9,14) = 126, and LCM(15,15) = 15.
The greatest common divisor GCD(x,y) is the largest positive integer that divides each of the integers, x and y. For example, GCD(3,4) = 1, GCD(8,12) = 4, and GCD(15,15) = 15.

(a) Given two integers x and y, write a C program that uses the function **int LCM(int x, int y, int result)** to calculate the *least common multiple (LCM)* of x and y, recursively.

(b) Given two integers x and y, write a C program that uses the function **int GCD(int x, int y)** to calculate the *greatest common divisor (GCD)* of x and y, recursively.
Hint: The **GCD** can be implemented using Euclid's algorithm, which works as follows:

   (a) If the two numbers are identical i.e. x = y, GCD(x,y) = x
   (b) If one number is larger, subtract the smaller number from the larger number, and then compute the GCD of the difference and the smaller number recursively until condition (a) is met.

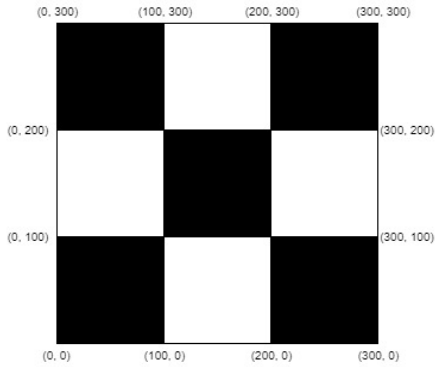
   Then modify the solution a) to find LCM using GCD. LCM(x,y) = (x*y)/GCD(x,y).

Task 2. Given integer n, write a C program that plots out a triangle of dimension n, recursively. An input/output example is illustrated below (input is typeset in bold):

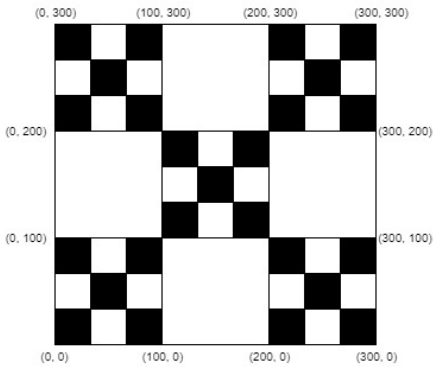
Enter n: **6**
Output:

```
     *
    ***
   *****
  *******
 *********
***********
```

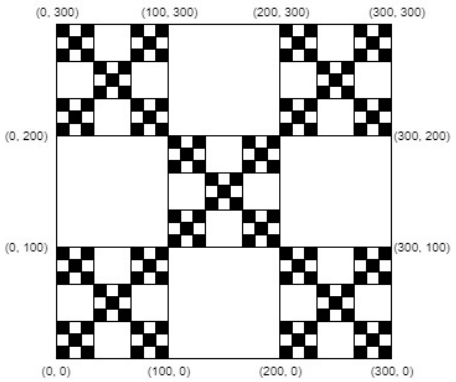
Task 3. The Box Fractal is fractal made by using the "method of successive removals". Iteration starts with a solid (filled) square. Then it is divided into 9 smaller congruent squares. Keep the four corner squares and the middle square but remove the other four squares. The Box Fractal of the i-th iteration, BF_i, is constructed by subdividing each of the five remaining solid squares of BF_{i-1} iteration into 9 congruent squares and removing the corresponding set of 4 squares in each of the 9 congruent squares. The Box Fractal of the first 3 iterations is shown on Figure 1.
A square is defined by three parameters. The coordinates (`x`, `y`) of its lower-left corner and its side length (`l`).



(a) Iter=1



(b) Iter=2



(c) Iter=3

Figure 1: Box Fractal

(a) Create the C function **drawBoxFractal(double x, double y, double l, int iter)**. The function should calculate the coordinates of the lowerleft corner and the side length that define each of the 5 subsquares, do appropriate recursive calls for each of the 5 subsquares if the number of iterations left is greater than one or print the coordinates and side length of the subsquares that must be drawn.

Example: The invocation **drawBoxFractal(0, 0, 300, 1)** prints the coordinates of the five remaining subsquares in order for the Box Fractal of

the first iteration to be drawn. The output would be:

```
(0.00, 0.00), 100.00
(200.00, 0.00), 100.00
(100.00, 100.00), 100.00
(0.00, 200.00), 100.00
(200.00, 200.00), 100.00
```

(b) Write a program in C that adapts the function **drawBoxFractal(double x, double y, double l, int iter)** to draw the Box Fractal using SDL Library.
Note: A skeleton of the solution of Task 3(b) is provided.

Task 4. What is the output of the following program and what does the function fun() do?

```
1
2  #include <stdio.h>
3  int cnt = 0;
4  void fun(char s[], int i)
5  {
6    if (s[i] == '\0')
7      return;
8    if (s[i] == 'a' || s[i] == 'e' || s[i] == 'i' || s[i] == 'o' || s[i] == 'u' ||
9      s[i] == 'A' || s[i] == 'E' || s[i] == 'I' || s[i] == 'O' || s[i] == 'U')
10     cnt++;
11   fun(s, i + 1);
12 }
13 int main()
14 {
15   fun("Recursion Exercise", 0);
16   printf("%d\n", cnt);
17   return 0;
18 }
```

Task 5. What is the output of the following program and what does the function fun() do?

```
1  #include<stdio.h>
2
3  int fun(int n)
4  {
5    if (n ≤ 100) { return fun(fun(n + 11)); }
6    else { return n − 10; }
7  }
8
9  int main()
10 {
11   printf("%d\n", fun(89));
12   return 0;
13 }
```