Programming & Data Structure

CS 11002

Partha Bhowmick

http://cse.iitkgp.ac.in/~pb

CSE Department
IIT Kharagpur

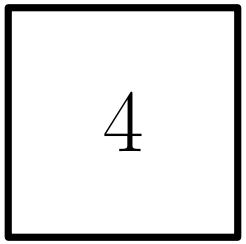
Spring 2012-2013

Structure

Why

A Math Puzzle

Subdivide an integer square by a *minimum* number of smaller integer squares such that their *GCD of their lengths is unity.*



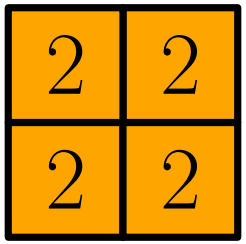
Program Requirement

Report each square by its *center coordinates* (x, y) and *length* a. (Assume that the bottom-left corner of the given square is at $(0, 0)$.)

Why

A Math Puzzle

Subdivide an integer square by a *minimum* number of smaller integer squares such that their *GCD of their lengths is unity.*



correct

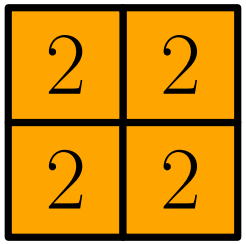
Program Requirement

Report each square by its *center coordinates* (x, y) and *length* a. (Assume that the bottom-left corner of the given square is at $(0, 0)$.)

Why

A Math Puzzle

Subdivide an integer square by a *minimum* number of smaller integer squares such that their *GCD of their lengths is unity.*



not correct

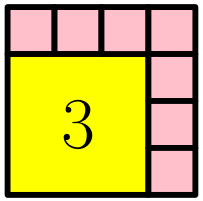
Program Requirement

Report each square by its *center coordinates* (x, y) and *length* a. (Assume that the bottom-left corner of the given square is at $(0, 0)$.)

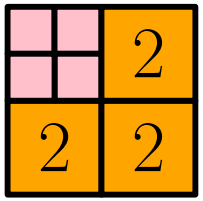
Why

A Math Puzzle

Subdivide an integer square by a *minimum* number of smaller integer squares such that their *GCD of their lengths is unity.*



correct
but not minimum



correct & minimum

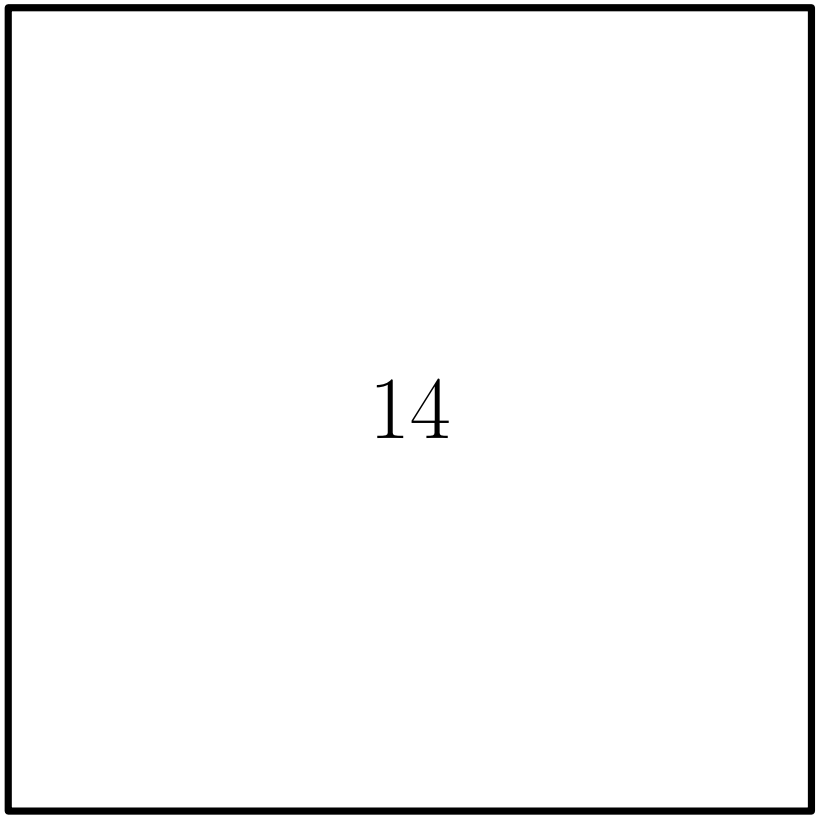
Program Requirement

Report each square by its *center coordinates* (x, y) and *length* a. (Assume that the bottom-left corner of the given square is at $(0, 0)$.)

Why

A Math Puzzle

Subdivide an integer square by a *minimum* number of smaller integer squares such that their *GCD of their lengths is unity.*



Program Requirement

Report each square by its *center coordinates* (x, y) and *length* a. (Assume that the bottom-left corner of the given square is at $(0, 0)$.)

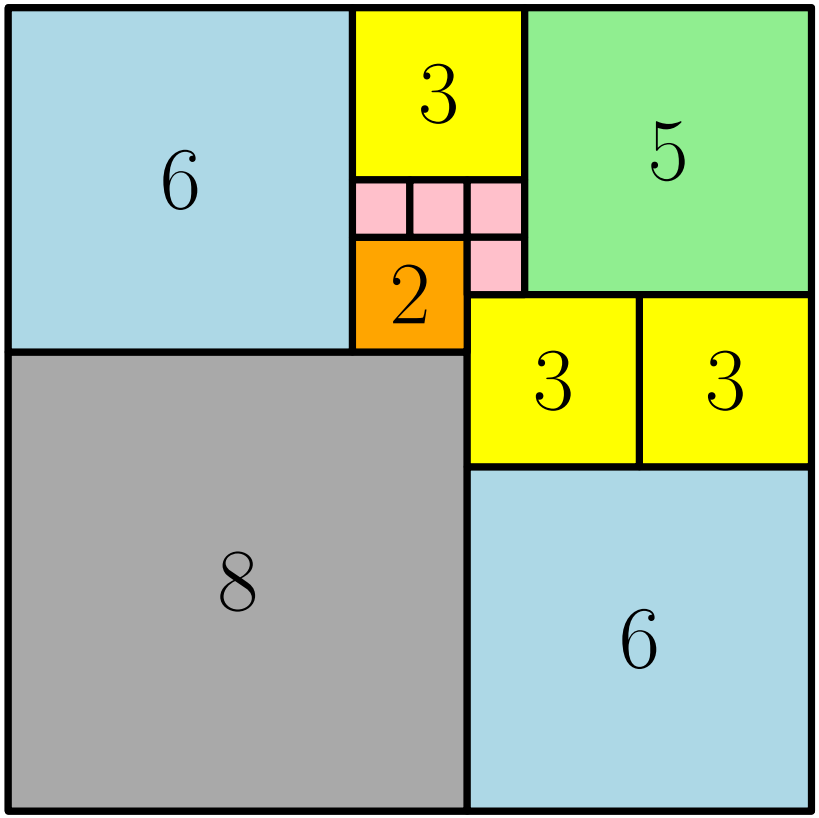
Why

A Math Puzzle

Subdivide an integer square by a *minimum* number of smaller integer squares such that their *GCD of their lengths is unity.*

Program Requirement

Report each square by its *center coordinates* (x, y) and *length* a. (Assume that the bottom-left corner of the given square is at $(0, 0)$.)



correct & minimum

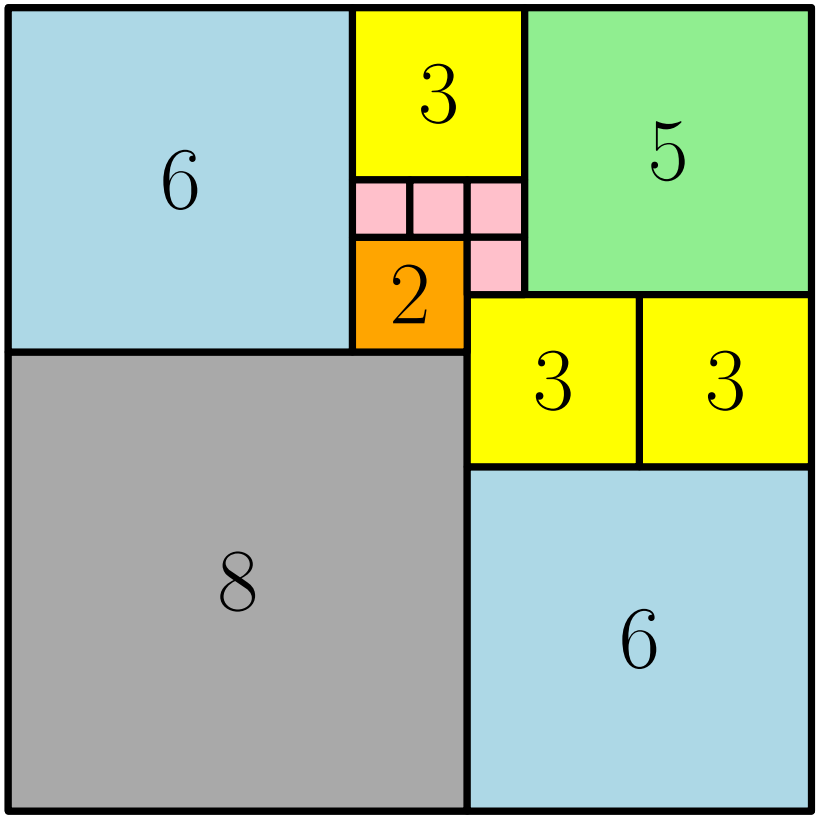
Why

A Math Puzzle

Subdivide an integer square by a *minimum* number of smaller integer squares such that their *GCD of their lengths is unity.*

Program Requirement

Report each square by its *center coordinates* (x, y) and *length* a. (Assume that the bottom-left corner of the given square is at $(0, 0)$.)



correct & minimum

Representation by struct (1)

In the square puzzle, for each square, we have three variables: x, y, a. These three variables can be represented by a *newly defined data type*, called *C structure*. It helps in handling a group of logically related data items.

```
struct square {
        int x, y, a; };
```

`struct square` is the new data type.
Two data items of this data type can be declared as:

```
struct square sq1, sq2;
```

Representation by struct (2)

Alternative ways of defining and declaring

```
typedef struct {
        int x, y, a;
} square;
square sq1={4,4,8}, sq2, sq3[5], *sq4;
```

sq1 is an initialized square type data;
sq2 is uninitialized;
sq3[] is an 1D array of 5 elements of type square; sq4 is a pointer to square, can hold an address, and its pointer arithmetic is determined by the sizeof(square).

Representation by struct (3)

We can also define it as follows:

```
struct squareType {
        double real, imag;
};
typedef struct squareType square;
```

Operations (1)

Complex Numbers

```
typedef struct {
        double real, imag;
} complex;
```

Example

```
complex a[5], b;
b.real = 3.0; b.imag = 4.0;
a[0].real = 2.0*b.real + 3.0/b.imag;
```

A field *of structure can be accessed by the '.' (projection) operator.*

Operations (2)

Group operations

Unlike arrays, group operations can be performed with structure variables. A structure variable can be directly assigned to another structure variable of the same type.

Example

```
complex a, b;
b.real = 3.0; b.imag = 4.0;
a=b;
```

All the individual members of `a` get assigned.

Operations

Two structure variables can be compared for equality or inequality.

Example

```
if (a1 == a2) printf("Equal");
```

Operations

Operation with pointer

```
complex a = {1.0,2.0}, *p;
```

- `p = &a;` makes `p` point to `a`.
- `p = (complex *)malloc(sizeof(complex));` allocates space for type `complex`.
- `(*p).real = 5.0;` assigns `5.0` to `a.real`, as `*p` is the object `a`. Since '`.`' has higher precedence than '`*`', a parenthesis is essential.
- `p -> real = 5.0;` is equivalent.

Functions with struct (1)

```
// dataComplex1.c

#include <stdio.h>
#include <stdlib.h>

typedef struct {
        double real, imag;
} complex;

complex addComplex(complex, complex);
void printComplex(complex);
```

Functions with struct (2)

```
int main(){
    complex a = {1.0, 2.0}, b, c[5], *p;
    b.real = 3.0, b.imag = 4.0;
    p = (complex *)malloc(sizeof(complex));
    (*p).real = 5.0;  p -> imag = 6.0;
    c[0].real = 7.0, c[0].imag = 8.0;
    c[1] = addComplex(a,b);
    c[2] = addComplex(*p, *p);
    printComplex(c[1]); putchar('\n');
    p = &c[2];
    printComplex(*p);   putchar('\n');
    return 0;
}
```

Functions with struct (3)

```
complex addComplex(complex x, complex y){
    complex t;
    t.real = x.real + y.real;
    t.imag = x.imag + y.imag;
    return t;
}
void printComplex(complex x){
     printf("%f + j%f", x.real, x.imag);
}
```

```
$ cc -Wall dataComplex1.c
$ ./a.out
4.000000 + j6.000000
10.000000 + j12.000000
```

Nested struct (1)

```
#define NAME 50
#define ROLL 9
typedef struct {
        char name[NAME], roll[ROLL];
        int sem;
        struct {
           float sgpa; int gp;
        } sgpa[10];
        float cgpa;
} student, studentIIT;
```

The field name `float sgpa` *does not clash with the type name* `struct sgpa[10]`.

More than one type name can be given using `typedef`.

Exercise (1)

1. Extend the complex number program to include functions for addition, subtraction, multiplication, and division.
2. Define a structure for representing a point in two-dimensional Cartesian co-ordinate system. Then write functions to compute:
    1. the distance between two given points.
    2. the midpoint of the line segment joining two given points.
    3. the area of a triangle, given the coordinates of its three vertices.