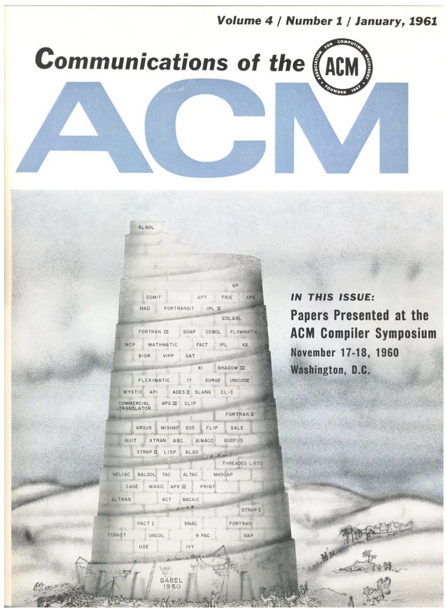
Programming Languages

CSE 3302, Summer 2019
Homework 05


Volume 4 / Number 1 / January, 1961
Communications of the ACM
ACM
IN THIS ISSUE:
Papers Presented at the ACM Compiler Symposium
November 17-18, 1960
Washington, D.C.
BABEL 1960

PA 05

Programming Assignment 05
Part A

Programming Assignment 05a

- Write three Java functions to *recursively* (***not** iteratively*) calculate these functions ➔

- Print the results for
  - `sumOfCubes`: n = 0 .. 15
  - `pseudoPell`: n = 0 .. 15
  - `GCD`:

    m = 1 .. 10, n = 1 .. 10

`sumOfCubes( n ) ≡`
  `0,` *for $n = 0$*
  n^3 `+ sumOfCubes( n-1 ),` *for $n > 0$*

`pseudoPell( n ) ≡`
  `0,` *for $n = 0$*
  `1,` *for $n = 1$*
  `3 × pseudoPell( n-1 ) + 2 × pseudoPell( n-2 ),` *for $n > 1$*

`GCD( m, n ) ≡`
  `m,` *for $n = 0$*
  `GCD( n, m%n ),` *for $n > 0$*

Programming Assignment 05a

`sumOfCubes( n ) ≡`
  `0,` *for $n = 0$*
  n^3 `+ sumOfCubes( n-1 ),` *for $n > 0$*

- Could hardly be more straightforward.
- The function definition will follow the specification exactly.

```
sumOfCubes(0) is 0
sumOfCubes(1) is 1
sumOfCubes(2) is 9
sumOfCubes(3) is 36
sumOfCubes(4) is 100
sumOfCubes(5) is 225
sumOfCubes(6) is 441
sumOfCubes(7) is 784
sumOfCubes(8) is 1296
sumOfCubes(9) is 2025
sumOfCubes(10) is 3025
sumOfCubes(11) is 4356
sumOfCubes(12) is 6084
sumOfCubes(13) is 8281
sumOfCubes(14) is 11025
sumOfCubes(15) is 14400
```

Programming Assignment 05a

```
pseudoPell( n ) ≡
  0, for n = 0
  1, for n = 1
  3 × pseudoPell( n-1 ) + 2 × pseudoPell( n-2 ), for n > 1
```

- Again, you should have no problems writing this function.
- The clauses in the specification correspond to the function definition.

```
pseudoPell(0) is 0
pseudoPell(1) is 1
pseudoPell(2) is 3
pseudoPell(3) is 11
pseudoPell(4) is 39
pseudoPell(5) is 139
pseudoPell(6) is 495
pseudoPell(7) is 1763
pseudoPell(8) is 6279
pseudoPell(9) is 22363
pseudoPell(10) is 79647
pseudoPell(11) is 283667
pseudoPell(12) is 1010295
pseudoPell(13) is 3598219
pseudoPell(14) is 12815247
pseudoPell(15) is 45642179
```

Programming Assignment 05a

```
GCD( m, n ) ≡
  m, for n = 0
  GCD( n, m%n ), for n > 0
```

- Yawn. Could this get any easier?
- Specification ⟺ Function.

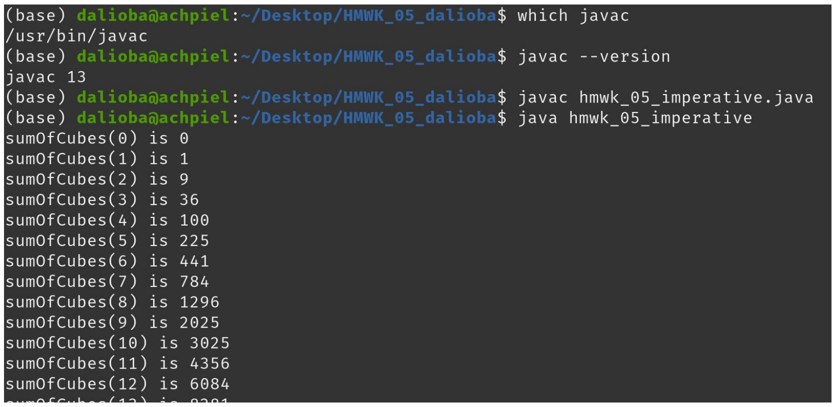
```
. . .
GCD(5, 5) is 5
GCD(5, 6) is 1
GCD(5, 7) is 1
GCD(5, 8) is 1
GCD(5, 9) is 1
GCD(5, 10) is 5
GCD(6, 1) is 1
GCD(6, 2) is 2
GCD(6, 3) is 3
GCD(6, 4) is 2
GCD(6, 5) is 1
. . .
```

Programming Assignment 05a

- OK, here's a skeleton …
- Geez, it's just three functions and a `main` that calls them.
- In my solution, each function is *3* lines of code in total!

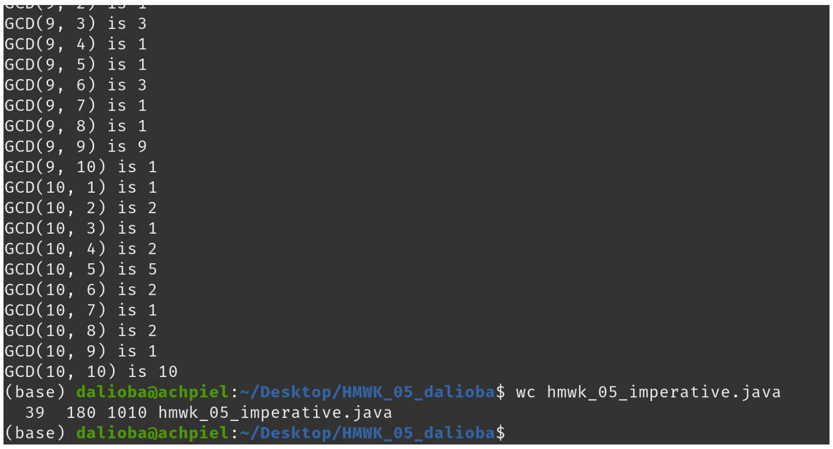
```java
public class hmwk_05_imperative {
  static Long sumOfCubes( Long n ) {
    // Put something here.
  }

  static Long pseudoPell( Long n ) {
    // Put something here.
  }

  static Long GCD( Long m, Long n ) {
    // Put something here.
  }

  public static void main( String[] args ) {
    // Put for loop here that calls sumOfCubes().

    // Put for loop here that calls pseudoPell().

    // Put nested for loops here that call GCD().
  }
}
```

Programming Assignment 05a—First Few Lines



```
(base) dalioba@achpiel:~/Desktop/HMWK_05_dalioba$ which javac
/usr/bin/javac
(base) dalioba@achpiel:~/Desktop/HMWK_05_dalioba$ javac --version
javac 13
(base) dalioba@achpiel:~/Desktop/HMWK_05_dalioba$ javac hmwk_05_imperative.java
(base) dalioba@achpiel:~/Desktop/HMWK_05_dalioba$ java hmwk_05_imperative
sumOfCubes(0) is 0
sumOfCubes(1) is 1
sumOfCubes(2) is 9
sumOfCubes(3) is 36
sumOfCubes(4) is 100
sumOfCubes(5) is 225
sumOfCubes(6) is 441
sumOfCubes(7) is 784
sumOfCubes(8) is 1296
sumOfCubes(9) is 2025
sumOfCubes(10) is 3025
sumOfCubes(11) is 4356
sumOfCubes(12) is 6084
```

Programming Assignment 05a—Last Few Lines



```
GCD(9, 3) is 3
GCD(9, 4) is 1
GCD(9, 5) is 1
GCD(9, 6) is 3
GCD(9, 7) is 1
GCD(9, 8) is 1
GCD(9, 9) is 9
GCD(9, 10) is 1
GCD(10, 1) is 1
GCD(10, 2) is 2
GCD(10, 3) is 1
GCD(10, 4) is 2
GCD(10, 5) is 5
GCD(10, 6) is 2
GCD(10, 7) is 1
GCD(10, 8) is 2
GCD(10, 9) is 1
GCD(10, 10) is 10
(base) dalioba@achpiel:~/Desktop/HMWK_05_dalioba$ wc hmwk_05_imperative.java
  39  180 1010 hmwk_05_imperative.java
(base) dalioba@achpiel:~/Desktop/HMWK_05_dalioba$
```

PA 05

Programming Assignment 05

Part B

Programming Assignment 05b

- Rewrite the three functions `sumOfCubes`, `pseudoPell`, and `GCD` from PA05a to be recursive *lambda* expressions instead of an expression and a `return` statement.
  - This is incredibly easy.
  - If you take more than a few minutes to do this, you are certainly overthinking the problem and probably going down a way-wrong path.
- The output of PA05b should be *identical* to that of PA05a.
  - Use *almost* the same `main` function. (Remember you have to `apply` lambdas to argument lists).

Programming Assignment 04b

- Do ***not*** use a **code block** in your definitions of the lambdas for the three definitions.
  - Code blocks are for more complex lambdas. You don't need them for this introductory example.
- Recursive lambdas run into problems with name visibility.
  - For example, `error: self-reference in initializer`
  - Remember our discussion about when names become visible in declarations. In Java, one can't use a name in its own initializer.
  - Use a *qualified name* in the initializer when referring to the name that's being defined.

Programming Assignment 05b

- OK, here's a skeleton …
- Geez, it's just three lambdas and a `main` that calls them.
- In my solution, each lambda is *3 or 4* lines of code.

```java
import java.util.function.UnaryOperator;
import java.util.function.BinaryOperator;

public class hmwk_05_functional {
  // sumOfCubes lambda goes here.

  // pseudoPell lambda goes here.

  // GCD lambda goes here.

  public static void main( String[] args ) {
    // Put for loop here that applies the sumOfSquares lambda.

    // Put for loop here that applies the pseudoPell lambda.

    // Put for loop here that applies the GCD lambda.
  }
}
```

Programming Assignment 05b

- The output of PA05b should be *identical* to that of PA05a.
- The lambdas should *not* use code blocks.
  - ***No*** `return` statements allowed!