```
Faculteit der Bètawetenschappen     exam Introduction to Programming (Java)
Vrije Universiteit                     December 16th, 2019 time: 2:45 hours
--------------------------------------------------------------------------
```

Problem 1.

a) Let the classes A and B be

```
class A {
    int g;

    A (int g) {
        this.g = g;
    }

    void add (int x) {
        g += x;
    }
}

class B {
    int g;
    A a;

    B (int g) {
        this.g = g;
        this.a = new A(g+2);
    }

    void add () {
        g += 1;
    }
}
```

Further we have a program with the following statements/declarations

```
A a = new A(1);
B b1 = new B(2);
B b2 = new B(3);
PrintStream out = new PrintStream(System.out);

void println (A x, B y) {
    out.printf("%d %d\n", x.g, y.g);
}

void doSomething () {
    println(a, b2);
    println(b2.a, b1);
    b1.add();
    b2.a.add(2);
    println(b1.a, b2);
    a = b2.a;
    b2.a.add(2);
    println(a, b2);
    b1.a = b2.a;
    b1.a.add(2);
    println(a, b2);
    a.add(2);
    println(b1.a, b1);
    println(b2.a, b2);
}
```

What will be printed when the method doSomething() is called?

b) The following heading of a method calculate() is given

```
double calculate (double x, int n)
```

This method calculates the sum of the first n terms of the series

```
1/x^2 – (2!+1)/x^3 + (4!+2)/x^4 – (6!+3)/x^5 + (8!+4)/x^6 - ....
```

x^n is the notation for "x to the power n" and n! is the notation for "the factorial of n". The number of terms that should be calculated is given by the parameter n. Implement this method. Do this without using any methods from the class Math. Assume: n≥1.

c) Give the declaration of a variable "matrix" with as type a 2-dimensional array of char's with 5 rows and 9 columns. Use constants when necessary.

Program a method numberOfOddOdds() that will be able, for any 2-dimensional array of int's, to count how many oddodds there are in the array. An int x in the array is an oddodd if x is an odd number on an odd row in an odd column. The odd rows/columns in a matrix are the 1st row/column, the 3rd row/column, the 5th row/column an so on.

examples

```
 2 2 7 2  The only oddodd is 7 as it is       2 2 2 2  no oddodds
 2 2 2 2  an odd number on the 1st row        7 7 7 7
 2 2 2 2  in the 3rd column.                  4 4 4 4
```

d)

```
class Problem_1d {
    PrintStream out;
    int a, b;

    Problem_1d() {
        out = new PrintStream(System.out);
        a = 3;
        b = 4;
    }

    void print (int x, int y) {
        out.printf("%d %d\n", x, y);
    }

    int m1 (int a) {
        a *= 2;
        b += 1;
        int c = a + b;
        print(a, b);
        return c;
    }

    int m2 (int c, int a) {
        a += 3;
        b = m1(c);
        print(c, b);
        c = a + b;
        return c;
    }

    void start() {
        print (a, b);
        int c = m1(a);
        print(a, c);
        a = m2(b, c);
        print(a, b);
    }

    public static void main(String[] args) {
        new Problem_1d().start();
    }
}
```

What will be printed when this program is executed?

Problem 2.

For every subproblem of that you encounter in problem 2, program that subproblem in a separate method in the correct class. Use constants when necessary.

a) Given is the following class:

```
class Gemstone {
    String type,            // for instance diamand, ruby and so on
           classification;  // precious or semi-precious
    int weight,             // in carats
        hardness;           // an int in range 1..10 (Mohs scale)
}
```

The constructor and the methods are omitted, as they are not necessary for this problem.

Make a class Necklace. This class should be able to contain a maximum of 75 gemstones. Further the class should have a default constructor and a method add(). The default constructor should initialize the Necklace-object to a necklace without any gemstones. The method add() should make it possible to add 1 gemstone to the necklace.

b) Program in the class Necklace a method

```
Necklace valuableGemstones ()
```

which, in a new Necklace-object, returns all the gemstones that are valuable. For this problem it is defined that a gemstone is valuable if its classification is "precious" and its weight is not less than 100 carats.

c) Program in the class necklace a method

```
void removeSoftGemstones ()
```

which removes all soft gemstones from the necklace. For this problem it is defined that a gemstone is soft if its hardness is less than 4.

d) Given is that the class Necklace contains a method

```
int gemstones (String name)
```

that returns the total number of gemstones of type 'name' in the necklace. You can use this method without having to program it.

Now add to the class Necklace a method

```
int valuableGemstones (String name)
```

which returns the total number of valuable gemstones of type 'name' in the necklace that are not soft gemstones. Program this method without using a for, while or do-while statement.

Problem 3.

a) Write a recursive method

```
String convert (char[] a, int i, int j)
```

that converts the elements on the index positions from i until j (inclusive), of an array of characters, recursively into a string.

```
examples: assume the array a contains {'a', 'b', 'c', 'd', 'e', 'f'}
            convert(a, 0, 0) gives "a"
            convert(a, 0, 1) gives "ab"
            convert(a, 0, 5) gives "abcdef"
            convert(a, 1, 0) gives ""
```

b) Given is that the class String contains a method

```
public String substring (int start, int end)
```

that returns the substring from start till end-1 (inclusive).

```
examples:   "abcdef".substring(3,6) gives "def"
            "abcdef".substring(1,4) gives "bcd"
            "abcdef".substring(0,3) gives "abc"
            "abcdef".substring(2,2) gives ""
```

Write, without using an iteration (do-while, while or for statement) a recursive method

```
boolean occurrences (String s, String sub, int n) // n >= 0
```

that, given a string sub and a string s, returns whether the number of occurrences of sub in s is >= n. Overlapping strings should be counted too.

```
examples:
        occurrences("","aba", 2)              →    false
        occurrences("ab","aba", 2)            →    false
        occurrences("aba","aba", 2)           →    false
        occurrences("abaxxaba","aba", 2)      →    true
        occurrences("abaxxabaxxaba","aba", 2) →    true
        occurrences("ababa","aba", 2)         →    true
        occurrences("xxxxxx","xx", 5)         →    true
```

grade:

```
        Problem a       b       c       d        total

        1.      5       5       5       5         20
        2.      2       6       6       3         17
        3.      4       4                          8
                                                  -- +
                                                  45
```

The grade E follows from the points P with the formula: E = P / 5 + 1