Faculty of Exact Sciences examination Programming
Vrije Universiteit December 16th, 2013 time: 2:45 hours

A N S W E R S

Problem 1.

a)
```
5,  5
53,  53
531,  531
7,  531
7,  5315
7,  53158
7123,  7123
71230,  71230
```

b)
```
double cosine (double x, int numberOfTerms) {
    double result = 0.0,
           numerator = 1.0;
    int sign = 1,
        denominator = 1;

    for (int i = 1; i <= numberOfTerms; i++) {
        double term = sign * numerator / denominator;
        result += term;

        sign = -sign;
        numerator *= x * x;
        denominator *= (2*i-1) * (2*i);
    }

    return result;
}
```

c)
```
static final int NUMBER_OF_ROWS = 5,
                 NUMBER_OF_COLUMS = 3;

char[][] matrix = new char[NUMBER_OF_ROWS][NUMBER_OF_COLUMS];

boolean normal (int[][] m) {
    for (int i = 0; i < m.length; i++) {
        int numberOfZeros = 0,
            numberOfPositive = 0;

        for (int j = 0; j < m[0].length; j++) {
            if (m[i][j] < 0) {
                return false;
            } else
            if (m[i][j] == 0) {
                numberOfZeros += 1;
            } else { // m[i][j] > 0
                numberOfPositive += 1;
            }
        }

        if (numberOfZeros > numberOfPositive) {
            return false;
        }
    }

    return true;
}
```

d)
```
-4, 9
-3, 4
2, 3
-2, -4
-3, 2
```

Opgave 2.

a)
```
class BotanicGarden {
    static final int MAX_NUMBER_OF_PLANTS = 10000;

    Plant[] plantArray;
    int numberOfPlants;

    BotanicGarden () {
        plantArray = new Plant[MAX_NUMBER_OF_PLANTS];
        numberOfPlants = 0;
    }

    void add (Plant plant) {
        plantArray[numberOfPlants] = plant;
        numberOfPlants += 1;
    }
}
```

b) add to the class Plant

```
boolean isOfFamily (String f) {
    return family.equals(f);
}
```

add to the class BotanicGarden

```
BotanicGarden selectFamily (String f ){
    BotanicGarden result = new BotanicGarden();

    for (int i = 0; i < numberOfPlants; i++) {
        if (plantArray[i].isOfFamily(f)) {
            result.add(plantArray[i]);
        }
    }

    return result;
}
```

c)
```
BotanicGarden selectUsingWind (String f) {
    return usingWind().selectFamily(f);
}
```

d) add to the class Plant

```
boolean isWeed () {
    return weed;
}
```

add to the class BotanicGarden

```
void weed () {
    int i = 0;
    while (i < numberOfPlants) {
        if (plantArray[i].isWeed()) {
            plantArray[i] = plantArray[numberOfPlants-1];
            numberOfPlants -= 1;
        } else {
            i += 1;
        }
    }
}
```

Opgave 3.

a)
```
void minimum (int[] row, int startIndex) {
    if (startIndex >= row.length) {
        return Integer.MAX_VALUE;
    }

    return Math.min(row[startIndex], minimum(row, startIndex+1));
}
```

b)
```
boolean palindrome (String s) {
    if (s.length <= 1) {
        return true;
    }

    int indexLastChar = s.length() - 1;
    char firstChar = s.charAt(0);
         lastChar = s.charAt(indexLastChar);
    String middle = s.substring(1, indexLastChar);

    return firstChar == lastChar && palindrome(middle);
}
```