This mock exam is to inform you about the layout and type of questions in the real exams, not about their content.

In the real exams, no materials are permitted except for one handwritten page which is to be handed in with the exam. Questions 1-4 are roughly equivalent and worth 30 points each. You are supposed to choose 3, i.e**. disselect one of these questions**. **Question 5 is worth 10 points and is mandatory**. Feel free to ask your teacher if you have understanding problems. Please use numbered blank pages for your answers, putting your name on each one. Hand in all your work, including scrap: You can get points for partly correct work, even on scrap pages, but not so in case of communication with any neighbours!

Question 1: Graphs

The edge weight list below describes an undirected weighted graph, i.e. all connectios have to be considered bi-directional. Create the correxponding distance matrix by hand and draw the graph. An edge weight list differs ffrom an edge list in that it contains triplets node, node, weight instead of node pairs.

A well-known algorithm for computing the shortest distance between all nodes works as follows: For every node in the graph, try to use it as intermediate node between all pairs of nodes. If the distance via the intermediate node is shorter than the distance recorded in the matrix, replace the matrix entry by the shorter distance. Write a static method which transforms a distance matrix given as parameter into a path matrix recording shortest paths using this algorithm.

What is the complexity of the algorithm?

Edge weight list: [5, 8; 1, 2, 2; 1, 5, 4; 2, 3, 3; 3, 4, 1; 4 ,5, 6; 2, 5, 1; 2, 4, 5; 3, 5, 5]

Question 2: Backtacking

Here is "OpFlips", a simpified "MatchMaths" problem: A formula consists of n integer operands combined by + or – operators, and an integer result, such as "17-3-4+2 = 14". The task is to find all correct fomulas that can be formed by flipping two operators from + to – or vice versa, e.g. "17+3-4-2=14".

Given the following class Formula, write a recursive backtracking method
`static void printSolutions(Formula form, int position, int flipsDone).`
which prints out all correct tranformations of the formula **and comment on its complexity**.The starting call would be `printSolutions(form, 0, 0);`

```
class Formula {
       public  int numops;      // number of operators in the formula
       private boolean [] ops; // length numops, array of operators, true is +
       private int[] numbers;   // length numops+2, all operands and the result

       public Formula(String input, int numops) {
       // creates numbers and ops and stores the input formula there
       }
       public void flipOp(int pos) { ops[pos] = !ops[pos]; // flip it
       }
       // effect can be undone by calling flipOp(pos) again;

       public void evaluate() {
       // prints formula, if mathematically correct
       }
}
```

Hint: Undoing the flips saves copying the formula for every recursive call.

Extra question – 5 extra points: Implement evaluate().

Question 3: Linked Structures

1. ***Briefly*** describe the advantages of using a double linked list over a single linked list. What is the function of an anchor node?

2. Assume a folowing class SortedList :

```
class SortedList {
   private class Node {
       String info;
       Node succ;
       Node pred;
       Node(String info)
       {      this.info = info;   }
   }
   protected static final String UNDEFINED = "###-Undefined-###";
   protected Node anchor = new Node(UNDEFINED);

   public void add(String info)
   { /* adds a Node containing info */ }
   }
   public boolean isEmpty() { /* true if the tree has no nodes */ }

}
```

Implement the two methods in bold type, add, and isEmpty. **What is the complexity of the add method?**

Hint: String implements `public int compareTo(String otherString)`. *It yields –1 if the current String is lexicographically smaller, +1 if it is larger, and 0 if the Strings are equal.*

Question 4: Clever Data Structures

The class below is an IndexedTable, i.e. it contains of an unsorted list of (bulky) entries, and an index table of the same size which contains a sorted order for the bulky list entries. The idea is to do sorting by moving lists of indeces rather than lists of bulky entries. Write the add method which adds an entry, keeping the index table sorted. It is ok to traverse the index table linearly.

```
class IndexedTable {

    private String[] content = new  String[size];
    private int [] index = new int[size];

    public void add(String entry)
    { /* add entry to content
         keeping the index table sorted by insertion
      */
     }
}
```

What is the complexity of finding an entry?

Extra question – 5 extra points: Use binary search to find the insertion point in the index table.

Question 5: Theory

Answer **5 of these 6** quesitons by marking true or false and giving a **brief explanation**. No points are given if the explanation is missing.

True ()
False()

1. The advantage of true random numbers is that they are reproducable

Explanation _____

True ()
False()

2. Dynamic programming reduces the computation time by distributing the computation over several. processors.

Explanation _____

True ()
False()

3. An RPN formula consists of a list of operands followed by a list of operators.

Explanation _____

True ()
False()

4. Tree balancing is only relevant on sorted trees.

Explanation _____

True ()
False()

5. Logarithmic complexity is usually achieved by using the divide and conquer pattern.

Explanation _____

True ()
False()

6. JUnit testing is a good alternative to debugging.

Explanation _____

_____

Good luck ☺