
TDDE45 Exam

Adrian Pop

Oct 27, 2025

These are the final exam instructions for TDDE45 Software Design and Construction. Note that there is a PDF and HTML version of these instructions - the PDF version is mainly for archival purposes and the HTML version is easier to read through.

The scheduled time for the exam is:

2025-10-30 08:00-12:00

1 Allowed material

- Engelsk ordbok / Dictionary: English-native language.
- Design Patterns: Elements of Reusable Object-Oriented Software by Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
- Head First Design Patterns by Freeman & Freeman

Note

Physical books only.

2 Provided materials

- Source code for the assignments.
- An excerpt from Wikipedia ([wiki.pdf](#)) with articles relevant to the assignments.
- A copy of [cppreference](#).
- A copy of the [Java 11](#) documentation.
- A copy of the [Java 17](#) documentation.
- A drawio AppImage (to draw UML diagrams - double click on it to start the application)

3 General Instructions

- Read the instructions and examination procedures for computer exams at IDA.
- Read all assignments carefully and completely before you begin.
- Be precise in your statements - imprecise formulations may lead to reduction of grade.
- Clearly motivate all statements and reasoning.
- Explain calculations and solution procedures.
- The assignments are *not* ordered according to difficulty.
- The exam is designed for 240 minutes with a discrete grading system. No points for half-finished solutions or imprecise formulations.

- Grading: U, 3, 4, 5.

4 Handing in Files

The desktop is writeable – you can store files here (but don't copy the entire source code here since it might be handed in to the grading teacher at the end).

For each assignment, add a file called Qn.ext (where n is the number of the assignment and the extension is one of odt, pdf, or txt). For example a file called Q1.txt for the first assignment. In this file, write the names of the other files used to answer the question.

Tip

The exam is intended to be anonymous – you do not need to put your LiU-ID in the answers.

5 Scoring

Each assignment is scored (0,3,4,5) and preliminarily the following table is used to set the final grades (for example, 2 assignments with a score of 4 and 2 with a score of 5 yields a final grade of 5).

Table 1: Preliminary scoring threshold

	Number of 3's	Number of 4's	Number of 5's
Final grade: 5		2	2
Final grade: 5			3
Final grade: 4			2
Final grade: 4		3	
Final grade: 4	2	2	
Final grade: 3			1
Final grade: 3	1	1	
Final grade: 3	3		

6 Netbeans

Read the source code for the *netbeans* CI environment (included on the filesystem with the instructions). Your task is to detect three different design patterns used in the source code.

- Explain what parts of the code make up the pattern and motivate why the code implements the pattern. Use diagrams if needed (preferably UML, but text or an image from dia or graphviz is also allowed).

6.1 Scoring

3. Find 1 design pattern.
4. Find 2 different design patterns. Fully motivated, clear which code uses the patterns and why.
5. Find 3 different design patterns (maximum 1 with the class named the same as the pattern). Fully motivated, very clear which code uses the patterns and why.

7 Performance Woes

Imagine you have a very large codebase and need to find out why the code suddenly runs very slow after pushing your new code to production. In testing, the size of the database was 5000 and ran fast; in production it is 100 times bigger, but runs more than 100 times slower than in testing.

For simplicity, the program (Q2.cpp) you will try to improve is much smaller, but you should imagine that the code is bigger and that you cannot manually review the code. Instead, you need to use tools to assist you in finding the problem.

For finding the problem, you may want to reduce the problem size. Increase it back again after solving the problem.

```
#include <vector>
#include <random>
#include <iostream>

using namespace std;

#define ARRSIZE 500'000

int isSorted(vector<int>& data) {
    int n = data.size();
    for (int i=0; i<n; i++) {
        for (int j=0; j<i; j++) {
            if (data[j] > data[i]) {
                cerr << "Data is NOT sorted" << endl;
                return 0;
            }
        }
        for (int j=i; j<n; j++) {
            if (data[j] < data[i]) {
                cerr << "Data is NOT sorted" << endl;
                return 0;
            }
        }
    }
    return 1;
}

/* Generated by co-pilot.
 * Guaranteed to be fast (TM).
 * No need to check this code.
 */

void fastSort(vector<int>& data, int start, int middle, int end){
    int left[middle-start+1];
    for (int l_cnt=0; l_cnt<middle-start; l_cnt++){
        left[l_cnt] = data[start+l_cnt];
    }
    left[middle-start] = numeric_limits<int>::max();
    int right[end-middle+2];
    for (int r_cnt=0; r_cnt<end-middle+1; r_cnt++){
        right[r_cnt] = data[middle+r_cnt];
    }
    right[end-middle+1] = numeric_limits<int>::max();
    int i = 0;
    int j = 0;
    for (int k=start; k<=end; k++){
        if (left[i] < right[j]){
            data[k] = left[i];
            i++;
        }
        else{
            data[k] = right[j];

```

(continues on next page)

```

        j++;
    }
}

void fastSort(vector<int>& data, int start, int end) {
    if (start < end){
        int middle = int((end + start) / 2);
        fastSort(data, start, middle);
        fastSort(data, middle+1, end);
        fastSort(data, start, middle+1, end);
    }
}

void fastSort(vector<int>& data) {
    while (!isSorted(data)) {
        fastSort(data, 0, data.size());
    }
}

int main() {
    random_device dev;
    mt19937 rng(dev());
    uniform_int_distribution<mt19937::result_type> dist10000(1,10000);
    vector<int> numbers;

    for (int i=0; i<ARRSIZE; i++) {
        int n = dist10000(rng);
        uniform_int_distribution<mt19937::result_type> pos(0, numbers.size());
        numbers.insert(numbers.begin()+pos(rng), n);
    }

    if (numbers.size() != ARRSIZE) {
        cerr << "Didn't produce the expected number of entries to sort" << endl;
    }

    fastSort(numbers);

    return 0;
}

```

```

# Some possible ways to compile the code
clang++ -O0 Q2.cpp
clang++ -O3 Q2.cpp
g++ -g Q2.cpp
# Running the executable
./a.out
man xxx # Opens man page for xxx
ls /usr/share/doc # Lists available documentation for tools installed on the system

```

7.1 Scoring

3. A reasonable explanation for the problem.
4. Found the problem using a tool (motivate with logs, etc).
5. Found the problem using a tool that could pinpoint the most problematic code location from the start (moti-

vate with logs, etc). Also fixed all problems.

8 Unicode

8.1 Scoring

3. Explain the relationship between latin1 and utf8 encoding in detail.
4. Also write a program that converts latin1 text to utf8 without using system libraries. The program should use properties of the encodings rather than a mapping of all code points.

9 Design for a multi-function device family

The problem you are supposed to design a solution for:

Your company wants to create a new family of devices. These will all need to share the same code base, but will have different functionality. The devices are essentially remote controls, but may have additional functionality:

- Remote control
- Multi-remote (being able to switch between different remotes)
- Detachable microphone (because everything needs a voice assistant)
- Detachable speakers
- Design a solution to the problem using a UML diagram (or similar).
- Create a program that solves the problem.
- Add test cases for this.
- Note that the detachable parts means that the class may need to change at runtime. Your tests should consider this, but it is enough to have code to go from 1 configuration to another (no events such as seeing a microphone connected need to be listened for.)
- Use a programming language available on the computer. You can stub out most of the program itself except when necessary for testing.

9.1 Scoring

3. A reasonable design pattern and UML diagram.
4. A good design pattern for the problem and a (mostly working) implementation.
5. The solution works and is even tested properly.