Homework 2

Introduction to Structured Programming

Presentation:	October 20, 2023
Submission (TU Graz TC):	November 8, 2023, 23:59

Have a look at our course website coding.tugraz.at regarding **plagiarism** and **homework grading**.

0 Introduction

In this homework, you will mainly produce written text. But it is not any text, it is a structured text to solve technical problems – so-called **problem plans**.

As in homework 1, you should write a report and upload the PDF to TU Graz *TeachCenter*.

> ⚠ **Please read all the instructions in this document in detail before working on the homework!**

What is a problem plan and how can I create one?

A plan helps you articulate the high-level structure of your solution to a problem. The plan may include information about different steps that need to be taken, as well as any ordering between different steps. Plans help ensure that your solution adheres to all of your instructions and constraints before you invest a lot of work in a single step. Then as you start working through our solution, you can look back at your plan to make sure you're on the right track.

Plans are also useful for communicating with others: architects and construction teams share plans to make sure everyone agrees on what needs to be built ahead of time. In programming teams, plans are also useful for making sure everyone agrees on what needs to be done. These plans are not code!

In the last lectures and practicals, we gave you an introduction to problem-solving. We provided you with a set of skills to think about problems. Skills such as decomposition, abstraction, pattern recognition, and algorithmic thinking can help you to find solutions. In this homework, you will be presented with three real-world problems that you should solve in a technical context.

Solving a problem means that you write structured instructions in natural language (so-called **problem plans**). Basically, you plan how you solve the given problems. In the next weeks, you will learn to write programs in a programming language, but for now on, we expect you to use the English language for your plans. You can write your plans in any text-based form you are comfortable with – just play around and try things (instructions in plain text, a pseudo-code notation, a list/enumeration of instructions, etc.).

Example

You want to do your weekly grocery shopping. You write yourself a list with all the products that you want to buy yourself.

Subsequently, you will see two variants of problem plans related to buying groceries. Keep in mind that there are many possible ways to go, you may find easier or more complex plans, which does not mean that one approach might be wrong.

Variant 1:

1. Start with the first element on the list which is not crossed out.
2. Locate the element in the grocery store.
3. Put the desired amount of the element in your shopping bag.
4. Cross the item off the list.
5. Repeat steps 1 to 4 until you've finished your list.
6. Go to the cash desk.

Variant 2:

1. Group items on the list according to where they are in the store.
2. Locate the first category in the grocery store.
3. Put each item from the category in your shopping bag.
4. Cross those items off the list.
5. Move to the next category.
6. Repeat Steps 2 to 5 for all categories.
7. Go to the cash desk.

1 Problem planning

We prepared a collection of three different problems for you. All problems have the same structure: you will be presented with an introductory text and a figure that should visualize the problem. After that, a specific task related to the problem is explained. Each problem consists of three tasks, so three problem plans have to be written per problem.

1.1 Problem planning tool

You will use the web-based problem-planning tool *Atlas*. This is a tool that was developed in collaboration with Brown University. The aim of this tool is to help you to understand how to design good plans/strategies for problems. Some tasks within *Atlas* are connected to an advanced code synthesis engine that translates your text to a programming language and tests the solution. The cool thing about it: You are producing programming code without writing program code.

http://atlas.cs.brown.edu/

Atlas is a completely new tool and was just finished. For this reason, you may encounter some unexpected behaviour. Don't get frustrated by that, see this as an opportunity to actively contribute to a promising project and report your problems via email (alexander.steinmaurer@tugraz.at).

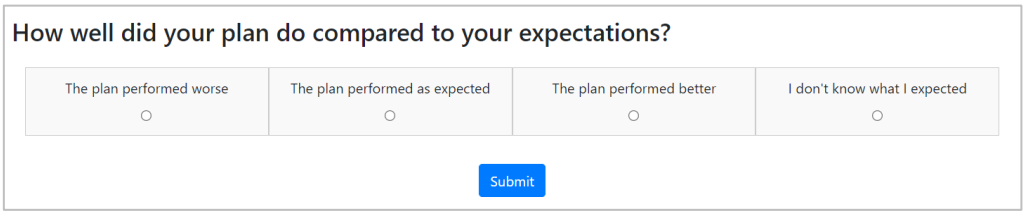
⚠ We expect you to use *Atlas* in a way that we (as the developers of the system) intended the tool. Of course, we appreciate creative solutions for the problems, but we expect **serious** attempts. Even though the system will do some automatic evaluation of your solution, we will review your solutions by hand as well.

1.2 Using the tool to write problem plans

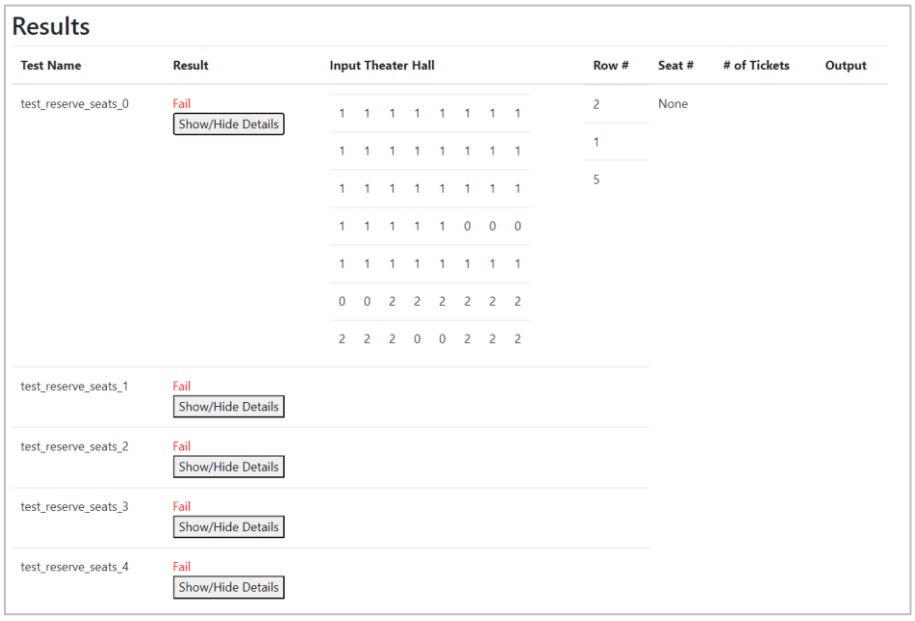
After you open the URL with a modern Browser of your choice, you can select a problem and a task. **Please go through all tasks in the given order** (starting with problem 1, task 1, followed by problem 1, task 2, and so on). When you click on a task you will see the description followed by a text field and a button. Write your problem plan into this text field. When you are happy with your solution, press the button.

As mentioned earlier, some of the tasks within Atlas are connected to an advanced code synthesis engine. To be more precise, problem 1, task 2 and problem 2, task 3 are connected to this system. When submitting these tasks, you will get additional feedback, that you won't receive for all other tasks. Receiving feedback might take some seconds, the (external) service has to make some calculations. After your solution is tested successfully, you will see a reflection question and the results. For these tasks, you can rework your problem plans again to address ideally all testcases. However, there is a restriction: you cannot make more than 5 improvements per task.

The purpose of the reflection questions is to figure out if your intended solution matches with the testcases that you were given.



The *Results* section of Atlas will give you information if your provided problem plan passed a testcase or not. When it says *Fail*, there was some problem with your solution, when it says *Pass*, you are fine. When clicking on *Show/Hide Details* you will get some details on a particular testcase (input and parameters of the testcase).



The final goal should be to pass as many testcases as possible. However, we will rate your dedication to solving a problem and not only if you passed all testcases. That means, even with a good problem plan and – let's say half of the testcases – you will receive the point for this problem as long as we see that you made some serious attempts to solve a problem (a serious attempt means that you had at least some iterations of revising a problem plan that did not work properly). As explained earlier, this homework is more about helping you get into problem-solving and continuously improve your solution to perform well. For those tasks where you will receive feedback, you have a maximum of **five attempts** to receive feedback from the system – so use them wisely.

2 Reflection question

The problem planning tool is about continuous improvement on your solutions and reflecting on your plans. For this reason, there is another final question that has to be answered when you are done with all tasks:

"*In what ways did your planning process change when writing the plans with test case feedback? How much did this feedback help/hinder your planning efforts?*"

Write the answer to this question into the report (see Section 3).

3 Write the report and submit it

Even though you upload all your problem plans to Atlas, there is also a report that has to be submitted in TeachCenter. Copy and paste all your final problem plans from the tool into a document (and add the problem and task number). In addition, answer the before-mentioned reflection question (Section 2) at the end of your document.

Save the document as a PDF and upload it by latest November, 8, 2023, 23:59 CEST in TeachCenter.

Information: The tool Atlas is developed in cooperation with Brown University to support students in learning to program. For this purpose, data within Atlas might be analyzed for further research on program planning in the context of computer science education. All data will be anonymized and aggregated before analyzing. If you still don't want the anonymized data from Atlas to be analyzed, feel free to email me at alexander.steinmaurer@tugraz.at without giving any reason.