Compiler and Language-processing Tools: Handout for the Portfolio Exam

What is a portfolio and why do we have this type of exam?

Portfolios are collections of artifacts. The notion is used in different contexts, ranging from financial asset management to application documents in art schools. In schools and universities, portfolios are used to collect and organize artifacts that characterize a learner's learning biography or make the learner's development visible or document his/her work on a project or topic.

Portfolios can also be used as an examination format that on the one hand enables the presentation of "high lights" of students' work results, but on the other hand also ensures the process of improvement and reflection of results over a longer period of time. Hence, the learner should reflect in the portfolio itself to what extent the teaching-learning process has led to personal development. In this way, the learning portfolio helps both the learner and the teacher to assess the learning process and learning outcomes. In order to use learning portfolios as a meaningful form of examination, they need to be pre-structured to allow for fair and transparent evaluation. Also, since self-reflection is new and unfamiliar for most of us, we want to provide you with a clear structure of the portfolio and outline the grading criteria we will use for our portfolio exam.

The portfolio for the course "Compiler and Language-processing Tool" in Winter Semester 2020 at TUK will be structured into three artifacts:

- a code repository where you will provide the implementation of a compiler part;
- a technical documentation where you will describe and motivate the architecture, design decisions, specific data structures, etc. that you have chosen in your implementation; and
- a text where you will reflect on your learning process and outcomes.

In the following, we describe the grading criteria for these artifacts in more detail.

Grading criteria

Implementation (total: 55%)

The portfolio will be centered around the implementation of a compilation task. The evaluation criteria for the implementation will be based on different aspects that reflect the skills that you practiced in the first part of the semester with the exercises.

Functionality (45%)

For the portfolio, you will implement aspects from the analysis and the translation phase of the compiler. The aspects will be ordered such that you don't need to tackle all of them in one go, but can start with the simpler aspects and gradually increase the difficulty. The portfolio description will further assign weight to the individual aspects.

As with the exercises, we will provide test cases as part of some code template. You can either use this template as starting point or the compiler that you have been writing in the exercises so far.

The provided test cases for the portfolio will *not* cover all corner cases (see below). Further, keep in mind that a passing test case does *not* guarantee or "prove" that an aspect has been successfully implemented. However, it should give you some early feedback and reassurance. In addition to the automated test cases, we will do some manual review of your code to compliance with the non-functional requirements listed below and check the conformity of implementation and documentation.

Additional Test cases (5%)

You should add up additional unit test cases that complement the template test cases. For each test case, you need to provide a short description to explain what aspect the test case covers. Please ensure that the test cases are unit tests, i.e. they should be minimal and each should concentrate on one aspect.

Hint: You can submit the tests, even if you do not implement the functionality required to pass them.

Non-Functional Requirements (5%)

Code style is checked automatically with checkstyle and follows mostly

https://google.github.io/styleguide/javaguide.html

Please use JavaDoc and comments to document essential aspects of your code. If in doubt, please contact us. We will further take aspects like readability and conciseness into account for the grading.

Evaluation Rubric

	1 (inadequate)	2 (poor)	3 (good)	4 (excellent)
Solution	Code does not compile and/or run; solution is incomplete	Runs, but has logical errors	Solution indicates the correct idea, but does not meet all the specifications and/or works for all OO test data	Complete solution runs without errors; it meets all the specifications and works for all OO test data.
Structure / Design	Only few of the selected abstractions, data structures and algorithms are appropriate; high amount of code duplication	Program uses some abstractions; several data structures and algorithms are appropriate	Programm uses mostly appropriate algorithms, and data structures, algorithms	Program uses efficient abstractions, data structures and elements
Style	Code ignores established guidelines;	Program ist minimally document; style	Some required documentation is missing; the	All required documentation is present; the style

	inefficient comments and documentation	guidelines are frequently violated	chosen style guide is mostly followed	guide is strictly followed

Documentation (total: 20%)

The documentation should comprise a summary of your portfolio work and a description of the algorithms and data structures that you used in your implementation. In the text, you should explain and justify your design decisions. Please use graphics and sketches as you see fit.

You should check Google's Design Document Procedure regarding style and structure of your documentation. We suggest having a look at the following sections:

- General principles
- Computer interfaces

The submission format is pdf (approx. 5-10 pages).

Evaluation rubric

	1 (inadequate)	2 (poor)	3 (good)	4 (outstanding)
Content	Description deviates from implementation; description does not support understanding of solution and tries to mask errors in the implementation and design	Description misses relevant aspects; it does not link to the code	Mostly comprehensive and complete description; focus on the relevant aspects; appropriate level of abstraction	Comprehensive and complete description; focus on the relevant aspects; very good level of abstraction
Structure	Not structured; hard to follow; irrelevant aspects dominate	Structure recognisable, but not followed coherently; some irrelevant information	Information presented in logical sequence; structure is plausible; in places, irrelevant aspects	Reasonable, comprehensible structure, appropriate consideration of all relevant aspects
Grammar, orthography	Consistently incorrect, error profile indicates massive orthographic and grammatical uncertainty; no	Numerous errors, uncertainty in formulating thoughts and ideas	Some errors, with a few repeating errors	Nearly error-free

	use of spell checker			
Style	Writing style for scientific work consistently inappropriate (e.g. only lists and no coherent text); strong accumulation of imprecise formulations	Writing style shows clear uncertainty regarding scientific writing; many conceptual oral phrases, often imprecise formulations	Writing style contains inappropriate formulations for a scientific work; sometimes imprecise formulations	Appropriate writing style for a scientific paper; precise wording; appropriate choice of words throughout; no overblown language
Citations	Incomplete, contradictory or intransparent information	Mostly complete, clear deficiencies in the presentation (e.g. inconsistent information; incongruence with information in the text)	Mostly complete, minor flaws in the presentation	Consistently correct and complete, uniform presentation following established citation styles
Technical language	Avoidance or incorrect use of technical terms, which indicates ignorance of the corresponding concepts	Frequent avoidance or misuse of technical terms, indicating a poor understanding of the relevant concepts	Tendency to paraphrase complex technical terms, which indicates a basic understanding of the corresponding concepts	Consistently understandable use and, if necessary, explanation of technical terms, which indicates a deep understanding of the corresponding concepts

Reflection (total: 25%)

The purpose of the reflection is to show that you are able to reflect and assess your own work and learning process critically. It should answer the following questions:

- What was the most interesting thing that you learned while working on the portfolio? What aspects did you find interesting or surprising?
- What part of the portfolio are you (most) proud of? Why?
- What adjustments to your design and implementation were necessary during the implementation phase? What would you change or do different if you had to do the portfolio task a second time?

- From the full lecture, what topic did excite you most? Why? What would you like to learn more about and why?

You may add further aspects as you see fit; if in doubt, please contact us for advise. The submission format is pdf (approx. 3-5 pages).

	1 (inadequate)	2 (poor)	3 (good)	4 (outstanding)
Content	Text does not answer any of the questions	Text misses relevant aspects; it does not to the portfolio task	Text comprises all aspects named in the questions; some answers seem unplausible	Text elaborates on the questions and has a strong connection to the presented work; it links to topics in the course beyond the portfolio task
Structure	Not structured; hard to follow; irrelevant aspects dominate	Structure recognisable, but not followed coherently; some irrelevant information	Information presented in logical sequence; structure is plausible; in places, irrelevant aspects	Reasonable, comprehensible structure, appropriate consideration of all relevant aspects
Grammar, orthography	Consistently incorrect, error profile indicates massive orthographic and grammatical uncertainty; no use of spell checker	Numerous errors, uncertainty in formulating thoughts and ideas	Some errors, with a few repeating errors	Nearly error-free
Style	Writing style for scientific work consistently inappropriate (e.g. only lists and no coherent text); strong accumulation of imprecise formulations	Writing style shows clear uncertainty regarding scientific writing; many conceptual oral phrases, often imprecise formulations	Writing style contains inappropriate formulations for a scientific work; sometimes imprecise formulations	Appropriate writing style for a scientific paper; precise wording; appropriate choice of words throughout; no overblown language

Citations	Incomplete, contradictory or intransparent information	Mostly complete, clear deficiencies in the presentation (e.g. inconsistent information; incongruence with information in the text)	Mostly complete, minor flaws in the presentation	Consistently correct and complete, uniform presentation following established citation styles
Technical language	Avoidance or incorrect use of technical terms, which indicates ignorance of the corresponding concepts	Frequent avoidance or misuse of technical terms, indicating a poor understanding of the relevant concepts	Tendency to paraphrase complex technical terms, which indicates a basic understanding of the corresponding concepts	Consistently understandable use and, if necessary, explanation of technical terms, which indicates a deep understanding of the corresponding concepts

Grading scale

We will use the following grading scale:

%	Grade
90 - 100%	1.0
85 - 90%	1.3
80 - 85%	1.7
75 - 80%	2.0
70 - 75%	2.3
65 - 70%	2.7
60 - 65%	3.0
55 - 60%	3.3
50 - 55%	3.7
45 - 50%	4.0
< 45%	5.0

On collaboration with team members, other students and beyond

Programming is a creative process. Individuals must reach their own understanding of problems and discover paths to their solutions. During this time, discussions with friends

and colleagues are encouraged, and they must be acknowledged when you submit your written work. When the time comes to write code, however, such discussions are no longer appropriate. Each program must be entirely your own work!

Do not, under any circumstances, permit any person to work on your code base. In particular, you may not test or debug another student's code, nor may you have someone test or debug your code. If you can't get code to work, consult the teaching assistant! You may look into different resources such as books, internet articles, papers etc. for ideas on how to solve the portfolio task. You may also discuss problems with your classmates on Mattermost. In your submission, all sources must be acknowledged and indicated. The standard penalty for violating these rules on one part of one assignment is to receive a zero for the entire assignment.

(The above policies were adapted from policies used by Norman Ramsey at Purdue University in Spring 1996.)

Registration, submission and other administrative aspects

The portfolio exam will be done in the period from Jan 11 - Feb 22 2020, 12:00. It comprises tasks equivalent two weeks of full time work (as required by the examination regulations). In this time, we will continue with the lectures, but will not have additional exercises sheets.

- Please register for this portfolio exam in QIS.
  - If you are a Bachelor student, registration must be before Dec 16, 2020.
  - Master students must register before Dec 28, 2020.
- You can step back from the exam without consequences 1 week before the exam starts (i.e. Jan 04).
- You will get access to a private git repository where you can host your portfolio, both code and documents. Grading will be done on the last version you pushed before Feb 22 2020, 12:00. We will not take the history of the repository into account when grading.
- We will provide the task description in English only. However, you can submit your documentation and reflection in English or German.

In case of illness or other emergencies with a duration of *more than 2 weeks*, please contact the examination office to discuss extensions.