LINKÖPING UNIVERSITY
Department of Computer and Information Science
Software and Systems
Christoffer Holm

2020-05-09

Computer examination in **TDDD38** Advanced Programming in C++

Date 2020-05-09

Time 08-14

Department IDA

Course code TDDD38

Exam code DAT1

Staff

Teacher on call: Christoffer Holm (christoffer.holm@liu.se)
Will answer questions through Microsoft Teams or E-mail.
Examiner: Klas Arvidsson, 013-28 21 46 (klas.arvidsson@liu.se)
Administrator: Anna Grabska Eklund, 013-28 23 62

Grading

The exam is split into two parts. The exam will be graded either *Passed* or *Failed.* To get a passing grade you have to *clearly* show your understanding of the course content in both part I and part II.

Communication

- You can ask questions to Christoffer Holm (christoffer.holm@liu.se) or Klas Arvidsson (klas.arvidsson@liu.se) through the chat in Microsoft Teams or by E-mail.
- General information will be published when necessary in Microsoft Teams through the team called `Team_TDDD38_Exam_2020_05_09`. Be sure to check there from time to time. A suggestion would be to turn on notifications in Microsoft Teams so you don't miss any important information.
- All communication with staff during the exam can be done in both English and Swedish.
- All E-mails must be sent from your official LiU E-mail address.
- In case of emergency call the teacher on call.

Rules

- You must sit in a calm environment without any other people in the same room.
- All types of communication is forbidden, the exception being questions to the course staff.
- All forms of copying are forbidden.
- You must report any and all sources of inspiration that you use. You may use cppreference.com without citing it as a source.
- Each of your source code files must have the following comment as their first line: **"I have read and understood the rules of the examination, and I swear to follow those rules."**
- When using standard library components, such as algorithms and containers, try to chose ”best fit“ regarding the problem to solve. Avoid unrelated/unnecessary computations and unnecessary data structures.
- C style coding is to be avoided.

- All concepts discussed during the course are OK to use.
- Your code must compile. Commented out regions of non-compiling code are acceptable if they clearly demonstrate the idea. Write a comment describing why that piece of code is commented out.
- You must be ready to demonstrate your answers to the staff after the exam if asked to.
- Failure to follow these rules will result in a *Failed* grade.

Submission

Submission will be done through Lisam on this page:
`https://studentsubmissions.app.cloud.it.liu.se/Courses/TDDD38_2020VT_OX/submissions`
You can also find this page by going to `https://lisam.liu.se`, navigating to the TDDD38 course page and clicking on "Submissions" in the left-hand side menu. There your should see the following submissions:

- 2020-05-09: Partial submission (09:00)
- 2020-05-09: Partial submission (10:00)
- 2020-05-09: Partial submission (11:00)
- 2020-05-09: Partial submission (12:00)
- 2020-05-09: Final submission part I
- 2020-05-09: Final submission part II

Partial submission: On the marked times you must send in the current state of all your solutions (all files). *Failure to do so within 5 minutes of the marked time will result in a failing grade.* We do not expect complete or even compiling solutions at this point.
Suggestion: Set an alarm so you don't forget.

Final submission: When you are done with the exam, you must send in your solutions through "Final submission part I" and "Final submission part II".

- Your solution(s) to part I should be source code files (.cc, .cpp, .h, .hh, .hpp).
- Your solution to part II should be a PDF document.
- Send *all* your files to `christoffer.holm@liu.se` and `klas.arvidsson@liu.se` by E-mail. This includes any .doc, .docx, .odt and .txt files. The subject line must be `COURSE: Exam 2020-05-09` where `COURSE` is replaced with either `TDDD38` or `726G82`.
- Remember that the first line in your source code files must be the following comment: **"I have read and understood the rules of the examination, and I swear to follow those rules."**
- The final submission must be submitted no later than 14:00.

Submitting through Lisam: Attach the files to your submission and press the submit button (it doesn't matter which one if there are multiple). You can select multiple files by holding Ctrl and clicking the files you want to attach.

You will be prompted "Do you really want to submit?". Double check that you have everything, and then press "Submit" on the popup.

You will then see a popup: "You will be redirected automatically when everything is finished" Once that has finished you will redirected to the submission page. You should also get a confirmation E-mail.

Part I

Introduction

This part of the exam deals with practical programming skills. You will discuss your solution to this part in part II of the exam.

Note that your code should compile on Ubuntu 18 with g++ version 7 or later with the flags: `-std=c++17 -Wall -Wextra -Wpedantic`. You can test your code on ThinLinc if you don't have access to Ubuntu 18 or g++ version 7 on your local machine.

The problem

In `html.cc` there is a given program. This program implements a simple framework for working with a data representation of (simplified) HTML documents. One of the main features of the program is to print the HTML elements as their textual representations. The program does not deal with parsing HTML documents, nor is it expected to. The intention of this framework is to generate structured HTML documents from code.

However, the person who wrote this code needs your help. You see, the code doesn't really use modern or even good C++ practices. Your job is to show how to make the code better by introducing classes, polymorphism, the STL and templates.

The focus for this assignment is for you to demonstrate that you can apply the language features discussed during the course to make the code better in any way you see fit. Some examples of concepts you could focus on when making the code better are: readability, maintainability, usability, code safety and efficiency. Note that this list is for inspiration, it is not a requirement. You don't have to use these concepts if you find other ways to improve the code. Just make sure to clearly discuss your intentions in part II.

The assignment

You must identify **suitable** parts of the given code that can be improved, and then demonstrate how to make those improvements. Your improvement must involve:

- STL Algorithms and containers
- Classes and Polymorphism
- Templates

For each concept you must demonstrate at least one place in the code that can be improved by introducing/using that concept.

Note: It is not required that you rewrite *everything*. It is enough that you rewrite parts of the code to demonstrate your ideas and understanding.

It is up to you to show that you understand these concepts. Remember that more advanced features does not necessarily imply better code.

Note: If you have trouble showing all of these concepts in one solution, you are allowed to create different solutions based on the given code. If you do this, place each solution in its own separate file and write a comment that describe which concepts you are covering in that file.

Suggestions and hints

Suggestion: Try to quickly analyze which parts will be easier and which will be harder to rewrite and plan your time accordingly.

Hint: There are a lot of comments in the code. Some of these comments contains a wishlist. These are improvements that the author would like the code to contain. You are free to use these wishlists as inspiration (you don't have to cite them as inspirations).

Hint: Some parts might be improved by completely rewriting them. Your solution doesn't have to use code from the given file, as long as your solution performs the same work as the given program but in a better way.

There are more hints and suggestions in the given file.

Part II

Rules

The answer to this part must be written as a text. You need to use a program where you can insert headers, text and code examples. You can for example use Microsoft Word or OpenOffice. It is also OK to use a pure text format (for example markdown). The important part is that the formatting clearly separates headers, text and code examples (and that you can export it as a pdf). The entire text should be possible to read and understand without reading your solution to part I. This means that you have to insert relevant pieces of code from your solution into the document. You document must be at least 1000 words and at most 2000 words (excluding code examples).

The assignment

You must answer each the following questions about your solution to part I. Remember to demonstrate **suitable** usage of these concepts in each question. More advanced features does not necessarily imply better code. It is recommended that you write one header per question.

1. Describe the class hierarchy of your solution. You should do **one** of these:
   - describe the classes and their relationships textually
   - draw a UML diagram (photos of hand drawn diagrams or digitally drawn diagrams are both OK)
2. Discuss how and why your usage of polymorphism is better than the given code. Describe the reasoning behind each virtual function, each class and the encapsulation. Discuss how these things improve the design of the program.
3. Describe a piece of your solution where you use templates and explain why you made those changes. If you have multiple places in the code to choose from, it is up to you to describe the one you think demonstrates the usage of templates best.
4. Describe a piece of your solution where you use STL algorithms and explain why you made those changes. If you have multiple places in the code to choose from, it is up to you to describe the one you think demonstrates the STL algorithms best.
5. Are there any things in your code or the given code that would be improved by using an appropriate container? Discuss why the code is improved by this choice or why none of the other containers would make an improvement. Mention one way the code improved **and** mention either a drawback or another improvement.
6. Does your solution follow the rule of 5?
   - If yes, motivate why you used the rule of 5.
   - If no, would it be a good addition? Why? Why not?