Rules

Rules - General

- Failure to follow the rules will result in a failing grade.
- Each solution is submitted according to the instructions of that assignment.
- Questions are asked through Zoom. Click on "Ask for Help" and we will help you as soon as possible.
- The final submission of your code should follow good C++ practices.
- You are to sit in an undisturbed environment without any other people in the same room. You should be visible and connected to Zoom at all times.
- You will be identified during the exam. Have your photo ID ready.
- Be ready to demonstrate your answers after the exam.
- If you need to take a break, send in the current state of your exam to the submission "2020-08-25: Break" in Lisam.
- General information during the exam will be published here: `https://www.ida.liu.se/~TDDE18/exam/2020-08-25/index.en.shtml`

Rules - Time plan

- All files must be submitted to Lisam no later than 17:45.
- If you want to attempt higher grades we suggest that you spend around two hours on part I and part II and the remaining hour on part III.
- If you only want to attempt a passing grade we suggest you spend half of the time on part I and the other half on part II.

Rules - Aids

- All forms of communication is forbidden, except with the course personnel.
- All forms of copying is forbidden.
- Every source you use for inspiration should be cited (except cppreference).
- You may use `cppreference.com` freely.
- You may use any C++ book if you cite it.
- You may use one page (A4) of your own notes if you submit them as an appendix to part II.

Rules - Grading

The exam consists of three parts. Complete solutions/answers to part I and part II are required for a passing grade. It is also required that you have submitted to the "Examination rules" submission in Lisam, which confirms that you swear to follow the rules. You have 3 hours to complete the exam. The third part is designated for higher grades. Plan your time accordingly. Use the last 15 minutes to check your solutions.

Part III consists of two assignments.

- To get grade 4 you need to solve *one* assignment.
- To get grade 5 you need to solve *both* assignments.

Agree to the examination rules

Before starting to work on Part I you must submit the message **"I have read and understood the rules of the examination, and I swear to follow those rules"** to the submission called "2020-08-25: Examination rules (14:00 - 17:45)" in Lisam (see below).

Do this before starting the exam!

Part I

Assessment of part I

For a passing grade on this part you must:

- follow all instructions and requirements presented in the assignment
- make sure that your code follows good programming practices
- write classes that have a clear responsibility and functions that have well defined purpose
- have good encapsulation and resource management

Instructions for submitting part I

You submit your solution through Lisam. You can find the the submissions page here: `https://studentsubmissions.app.cloud.it.liu.se/Courses/TDDE18_2019HT_7K/`. You can also find it by going to the course page of TDDE18 (even if you are taking 726G77) on `http://lisam.liu.se` and clicking "Submissions" in the menu. There you should see the following submissions (note that each part will become visible once it starts):

- 2020-08-25: Examination rules (14:00 - 17:45)
- 2020-08-25: Part I (14:30 - 17:45)
- 2020-08-25: Part II (14:30 - 17:45)
- 2020-08-25: Part III (14:30 - 17:45)

Attach the files you want to submit. Hold `Ctrl` to select multiple files. Confirm the submission. You should get a confirmation E-mail.

Your final submission must be well tested and should compile with `g++` version 7 with the following flags: `-std=c++11 -Wall -Wextra -Wpedantic -Weffc++` on Ubuntu 18. You can test this by using ThinLinc. You can use your local compiler and tools working out your solution, but be aware that we will assess your solution using g++ version 7 compiler and said flags.

Part I - Assignment (14:30 - 17:45)

If you haven't already done so, read the examination rules and then accept them by making a submission to *"2020-08-25 Examination rules"*. If you don't do this your exam will automatically fail.

In this assignment you must create a program consisting of at least two classes. Both classes should be closely related to the theme TEMA. Your program must demonstrate your knowledge of *constructors*, *data members*, *member functions* and *encapsulation*. You must also create a main program that demonstrates how your classes are used by testing each part of each class.

- One of the classes is *allowed* to be a simple aggregate.
- The other class must demonstrate *at least* two examples within **each** of the specified subjects (*constructors*, *data members*, *member functions* and *encapsulation*). At least one of the member functions must take parameters and perform a operation relevant to the class. Only returning a data member is too trivial.

You **must** include testcases for when an object of one of your classes is declared as `const`. Note that at least one member function must be called on the const object.

Remember to always use the specified theme TEMA. [1] In order to help you with your creativity, here are some suggested words/phrases related to TEMA that might help you figure out what your classes should be and do:

- PHRASE

Remember that it is up to you to show how much knowledge you have. This assignment give you reasonable freedom to actually show that knowledge.

[1] Observe that you don't have to be an expert within the given theme. If you feel like you don't have enough knowledge within this theme, then it is fine to make up your own facts about your theme.

Part II

Assessment part II

For a passing grade on this part you must:

- follow all instructions and requirements presented in the assignment
- correctly describe how your classes work
- describe how you arrived at your solution

Instructions for submitting part II

This assignment should be answered with text. You need to use a program where you can write headers, text and code examples. You could use Microsoft Word, OpenOffice or LibreOffice. It is also OK to use a purely textual format (for example markdown). The important part is that there is a clear distinction between headers, text and code. You must also be able to export your answers as a PDF. **This part must be possible to read without first reading your solution to part I**.

You must write somewhere between 500 and 2000 words. For reference, this single page is 435 words in LaTeX source code. There are plenty of ways to calculate the word count of a document. Check how you can do it in your program, or use an online application. With reasonable font settings (compare to this page) you can simply estimate 500 words per page.

You must submit your document as a PDF (one file) to the submission called "2020-08-25: Part II (14:30 - 17:45)" in Lisam (which will submit it to Urkund).

Part II - Assignment (14:30-17:45)

In this part you will explain the code you wrote for part I. Below there is a list of everything that must be included in your answer. You could for example create one header for each item. You must include ***ALL*** of these in your answer. We are not looking for "right" or "wrong" answers here. Instead we want to understand your thought process and how you think.

1. Describe how you came up with your solution. Here we are looking for how you reasoned.
2. Describe how your code demonstrates your understanding of *data member* and explain how they are used in your code.
3. Describe how your code demonstrates your understanding of *constructors* and explain how they are used in your code.
4. Describe how your code demonstrates your understanding of *member functions* and explain how they are used in your code.
5. Describe how your code demonstrates your understanding of *encapsulation* and explain how it is used in your code.
6. Describe how your code demonstrates your understanding of *const* and discuss what limitations and what advantages are induced when adding `const` to an object.

Remember to demonstrate all code and why you added it if you want that piece of code to be included in the assessment.

Part III

Assessment part III

This part consists of two assignments.

- To get grade 4 you need to solve *one* assignment.
- To get grade 5 you need to solve *both* assignments.

Note: An incomplete solution can still give a higher grade. We will make an assessment based on your demonstration of what you know. This means that even if you don't have the time to fix everything in the assignment you can still get a higher grade if your answers to the questions are reasonable and you have solved *enough* of the assignment.

Instructions for submitting part III

You submit your solution through the submission "2020-08-25: Part III (14:30 - 17:45)" in Lisam. You must submit your code with the filenames `assignment1.cc` and/or `assignment2.cc`. You submit the answers to the questions as a PDF.

Part III - Assignment I

Treasure hunters often form parties together to ensure success. There are two types of people in these parties, the normal treasure hunter and the archaeologist. The archaeologist has an academic education and therefor demands a different formula for calculating salary.
You need to create a polymorphic class hierarchy to represent the following classes so that the given program can execute without warnings and/or errors.
You need to create the following classes:

TreasureHunter This is the basic kind of treasure hunter.

A Treasure hunter is created with two parameters. Theses parameters are of the types `std::string` and `int`. The string represents the hunters name and the integer represents how skilled the hunter is, both are data members in the class.

A Treasure hunter also has two member functions that can be called, `salary()` and `get_info()`. `salary()` calculates and returns the monthly salary of the treasure hunter by multiplying its skill by `250`. The return value has to be of the type `double`. `get_info()` returns a string with the hunters name and its salary separated by `" - "`.

Archaeologist is an extension of `TreasureHunter`.

An Archaeologist is created with an extra parameter compared to the Treasure hunter. This extra parameter is the prestige of the archaeologists education and is of the type `double`. It is saved as a data member in the class.

The salary for archaeologists is calculated the same way as for Treasure hunters but is also multiplied by the prestige. You cannot repeat code, so you have to call the base class version.

There are partial testcases implemented in `assignment1.cc`.

Part III - Assignment II

In `assignment2.cc` there is a working program given. Your assignment is to rewrite this program so that it only uses STL algorithms. You are not allowed to use any loops at all. You are **only** allowed to use the following STL algorithms:

- `std::transform`
- `std::copy`
- `std::iota`
- `std::sort`
- `std::min`
- `std::min_element`
- `std::minmax`

Note: You don't have to use all of these algorithms, but any algorithm you use must come from this list.