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 "2021-01-15: Break" in Lisam.
- General information during the exam will be published here:
  `https://www.ida.liu.se/~TDDE18/exam/2021-01-15/index.en.shtml`

Rules - Time plan

- The exam starts at 09:00 and you will be let into Zoom at 08:30. You should be in Zoom no later than 08:30.
- Part I and part II must be submitted to Lisam no later than 12:15.
- Part III is submitted no later than 12:15 + Bonus time.
- If you want to attempt higher grades we suggest that you spend around two hours on part I and part II and the remaining time 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.
- Remember that the bonus gained from the labs only gives you more time on part III.

Rules - Bonus time

- Bonus time is valid only during the first regular exam after the course (January).
- Each met deadline in the course give 5 minutes bonus time for part III (higher grade) to a total of at most 30 minutes.
- The deadline for part I and part II is not affected by bonus time.

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 "2021-01-15: Examination rules (08:00 - 12:15)" 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_2020HT_LF/`. 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):

- 2021-01-15: Examination rules (08:00 - 12:45)
- 2021-01-15: Part I (09:00 - 12:15)
- 2021-01-15: Part II (09:00 - 12:15)
- 2021-01-15: Part III (09:00 - 12:45)

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

If you haven't already done so, read the examination rules and then accept them by making a submission to *"2021-01-15 Examination rules"*. If you don't do this your exam will automatically fail.

Think about Cities.[1] Invent two reasonable classes related to Cities. Here we call those classes `B` and `C`, however you should name them something appropriate relating to Cities. `B` and `C` should interact in some way. It may be as simple as "`B` consist of several `C`". Requirements checklist:

- `B` must be a class that clearly uses encapsulation (hiding things in the class)
- `B` must use one STL container as data member
- `B` must have at least two data members
- `B` must have at least one constructor with parameters initializing all members by a data member initializer list (do not confuse with `std::initializer_list`)
- `B` must have at least one member function using parameters and modifying the instance in some way (perhaps add an instance of `C` to the container?)
- `B` must have at least one member function that return the result of some calculation without modifying the instance
- `C` must be a class clearly using encapsulation
- `C` must have at least two data members

Both `B` and `C` may have additional properties as long as it does not interfere with any of the listed requirements.

In order to help you with your creativity, here are some suggested words/phrases related to Cities that might help you figure out what your classes should be and do:

- Street
- add street
- increase population

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 "2021-01-15: Part II (09:00 - 12:15)" in Lisam (which will submit it to Urkund).

Part II - Assignment

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 may 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 *STL containers* and explain how it is used in your code.

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 "2021-01-15: Part III (09:00 - 12:45)" 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

Now that Christmas is over some people will have to change their wishlists since they got gifts during Christmas. Your job in this assignment is to write a program that removes all the gifts that the user got from their wishlist.
Do this by writing a program that does the following:

1. Create a `std::vector<std::string>` called `wishlist` that contains the wishlist (if you don't know what to put, we recommend: Book, Bicycle, Laptop, Socks, Decorations and Candy).
2. Prompt the user for what gifts they got during Christmas by asking them "What did you get?". You should then read each word that the user entered into a new vector called `gifts`. The user is done when they press |ctrl+D|.
3. You should then remove all the elements in `wishlist` that also occurs in `gifts`. Note that `gifts` might contain things that aren't in `wishlist`. Those elements can be ignored.
4. Print the updated wishlist to `std::cout` where each entry in the wishlist is on its own line in the output. The order you print the wishlist entries in doesn't matter.

You are not allowed to use any loops at all. You are **only** allowed to use the following STL algorithms:

```
std::copy    std::remove_if        std::find
std::sort    std::set_difference
```

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

There is a sample output given in `assignment1.cc`.

Part III - Assignment II

Santa Clause had troubles during Christmas because of newly recruited elves wrapping gifts incorrectly. Unfortunately he doesn't know where the problem arose because it is not logged anywhere who did what in the wrapping process. So to solve this problem he has hired you write a program that keep track of the wrapping process.

In this assignment you will create a polymorphic class hierarchy for the elves responsible for the gift wrapping. You need to create the following classes such that the given program can execute without errors:

Elf A base class representing an elf.

An `Elf` consists of a `std::string` which represents the elf's name, and two member functions:

`get_name` which simply returns the name of the elf.

`handle` which is a function that's supposed to simulate the work of the elf. This function takes a string called `gift` and returns a string. The behaviour of this function depends on which type of elf this is. For the `Elf` class this function should be *pure-virtual*.

Wrapper Is a derived class of `Elf`. This represents an elf that wraps a gift in wrapping paper. This class contains a string called `color` which represents the color of the wrapping paper.

`handle` takes the passed in string `gift` and returns a string on the following format: `[gift] wrapped in [color] paper` where `[gift]` is the value of `gift` and `[color]` is the value of `color`.

Boxer Is a derived class of `Elf`. This is an elf that packs gifts into boxes. It doesn't have any additional data members.

`handle` returns a string on the format: `box containing a [gift]` where `[gift]` is the value of the passed in string `present`.

Each class must have appropriate constructors and correct `const` usage. In `Boxer` you must import the constructor from `Elf`. I.e. you cannot declare a new constructor in `Boxer`.
The program given in `assignment2.cc` is not fully complete. You must add your classes and fill in the appropriate types in the comments.

The expected output of a correctly implemented program is:

```
box containing a box containing a Book wrapped in blue paper wrapped in red paper

This present was packaged by:
Eminelf
Elf Mayweather
Mike Elfson
Elfpac
```