TDDE18/726G77 – Examination

Example

Rules

- All code sent for assessment must compile and be well tested.
- Electronic devices are not allowed. Phones must be switched off and placed in a coat or bag.
- Outdoor clothes and bags must be placed in the designated area.
- Students may leave no earlier than one hour after the exam start.
- Fill in invigilators designated list if you need to leave the room.
- All contact between students are strictly prohibited during the exam.
- Books and notes may be reviewed by invigilators during the exam.
- Questions regarding specific assignments or regarding the exam in general are submitted via the communication client.
- System questions can be answered by an assistant if you raise your hand.
- Assignments sent in after the end of the exam will be disregarded.
- You can correct flaws and ask for new assessment until an assignment has grade “Pass” or “Fail”. An assignment can be graded as “Fail” if no significant improvement took place since last attempt.

Aiding material	One C++-book One A4-page with any notes on both sides

Examination

The exam consists of two parts, Part I and Part II. Both are assessed live which means that handed in assignments will be assessed during the exam. Any flaws that the assessment revealed can be fixed and then the assignment can be handed in again for re-assessment.

All assignments are graded *Passed*, *Try again* or *Failed*. The grade *Passed* will be set if the assignment is fully completed according to specification and without any incorrect practices. The grade *Try again* will be set if there are some issues that need to be fixed. *Failed* is set if no sufficient improvement was made on fixing the given feedback.

Grading guidelines

For a passing grade you need to pass all assignments in Part I.

For a higher grade you need to pass all assignments in Part I and pass some assignments in Part II based on the table(s) below.

Grade	**Part I**	**Part II**
3	All	
4	All	One assignment
5	All	Two assignments

Table 1: Grading TDDE18

Grade	**Part I**	**Part II**
G	All	
VG	All	One assignment

Table 2: Grading 726G77

Computer environment

Log on

When instructed, log in as normal using you LiU-ID.

Desktop environment

Upon successful log in you will enter the desktop environment. The communication client should start automatically. Note that the network is inaccessible. Networked application features may thus malfunction.
It is important that you leave the communication client running during the entire exam. We may send out public corrections and hints. Notify assistant if it does not start automatically within 5 minutes after log in or after selecting the fish on the desktop.

Terminal commands

`e++17` is used to compile with “all” warnings *as errors*.
`w++17` is used to compile with “all” warnings. **Recommended**.
`g++17` is used to compile `without` warnings.
`valgrind --tool=memcheck` is used to check for memory leaks.

C++ reference

During the exam you will have *partial* access to `http://www.cppreference.com/`, but only through the desktop icon “Web access”. Do note that not everything on cppreference will be available (in particular the pages under the “Language” section will be blocked). If you are unable to access a page that should be available (it might have been blocked by mistake) then you can send a message through the exam client. *Note:* The search functionality should work, but only if you do it through cppreference. You *cannot* search on DuckDuckGo.

Given files

Any given files reside in the folder `given_files` on your desktop. This folder is write protected, thus you don’t have to worry about accidentally changing the given files. To modify a given file, you must first copy it to your work folder, use your desktop as a work folder. You are expected to know how to do this, it is part of the course.

Log off

When your assignment and exam grade is satisfactory (and correct) in your communication client it is safe to leave. If you run out of time you have to leave without knowing the result of your last attempt, contact the examiner by email after the exam to know the result. Terminate all open programs and log out.

Part I

In this part you are presented with three assignments. Your solutions to these assignments must fulfill all specified requirements, follow good programming practice and consist of valid C++ code.

Assignment #1 – STL

Your assignment is to build a system for validating and printing information about Linux-packages. First you will implement a class which represents a package. Secondly you will use that class to perform a series of operations on a list of packages.

The class `Package` represents a package and must at least contain a name, a publication year, and size and a checksum. Besides this, the class must fulfill the following requirements:

- The class must be correctly encapsulated and the user should only have access the things that are relevant for them.
- Comparison of two packages must be implemented as an operator which returns `true` if the left package is older than the right package.
- A function `is_valid()` which checks whether a package is valid according to the following formula: `name[0] + year + size == checksum`

You might have to implement more functions than these to make the steps below work properly.

In the main program these three steps must be implemented. Use the given list of packages in `assignment1.cc`.

1. Remove all packages that are not valid (using `is_valid()`). Either you do this by removing the elements from given list, or you create a new list with only the valid packages.
2. Get and print the name of the oldest package.
3. Calculate and print the total size of all valid packages (i.e. sum the size of all valid packages).

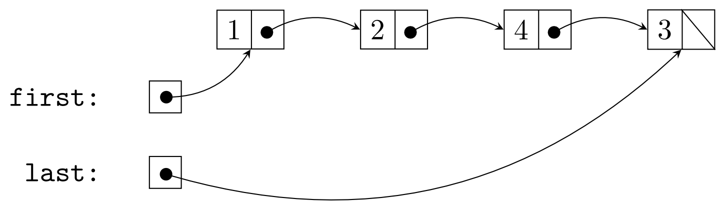
Non-functional requirements:

- Your solution must use *at least* one appropriate STL algorithm to solve *at least* one non-trivial part of the steps above.

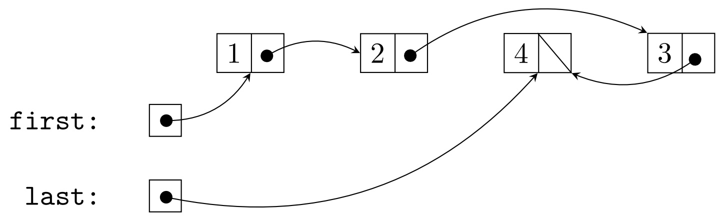
Functional requirements: Execution of the program should produce the following output in the terminal:

```
-== Package Stats ==-
Oldest package: neofetch
Total size of all packages: 4736
```

Assignment #2 – Memory



Write code which creates the linked structure above. Then write code which modifies the structure to match the structure given below. You do not have to write a general solution, it is enough to handle exactly the given structure.

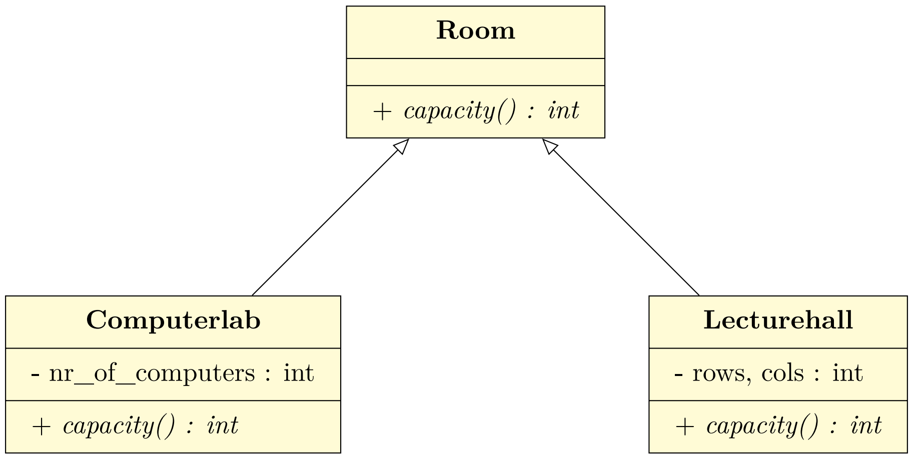


Non-functional requirements:

- All elements must be dynamically allocated. No need to deallocate elements in this assignment.
- You are not allowed to declare any variables besides those declared in the given `main()`.
- You must use the given `Node` from `assignment2.cc`

Assignment #3 – Polymorphism

Implement the following class hierarchy:



- `Room::capacity()` has no defined behaviour.
- `Lecturehall::capacity()` returns `(rows * cols) + 1`.
- `Computerlab::capacity()` returns `nr_of_computers * 2`.

Non-functional requirements:

- All your classes must be declared and defined in *one* file.
- The given file `assignment3.cc` has a main program and a function that should work without any modifications.

Functional requirements: Execution of the program should produce the following output in the terminal:

```
Capacity is 18
Capacity is 73
Capacity is 169
Capacity is 20
Capacity is 48
```

Part II

This part consists of assignments that provides opportunity for higher grades.

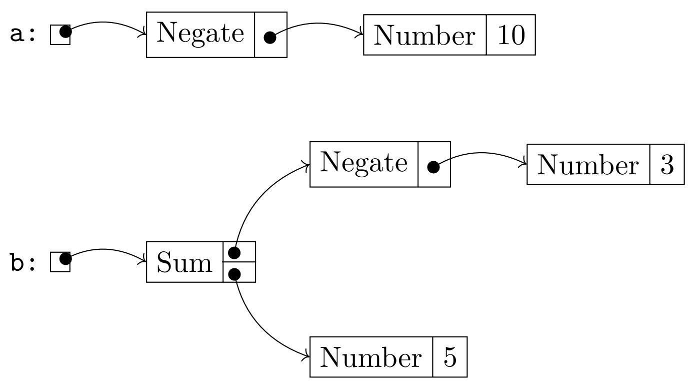
Assignment #4 – Memory

In this assignment you will implement three classes as part of a polymorphic inheritance hierarchy to represent mathematical expressions.

- `Number` represents an integer.
- `Negate` multiplies the number that comes directly after with −1. In example `a` below, `Negate` will receive the number 10 and output −10.
- `Sum` calculate the sum of the two numbers that comes after.

It is enough that you implement the functions that are required to make the main program in `assignment4.cc` work correctly. Correct encapsulation is however required.

The given main program construct three expression you can use for testing, but you need to modify it so that it works correctly. Below you can see two of the expressions:



Non-functional requirements:

- Each class must be responsible to correctly destroy the nodes they use.

Functional requirements: Execution of the program should produce the following output in the terminal:

```
-10
2
3
```

Assignment #5 – STL

A list of randomized points is given. These points need to be grouped into different square regions. Each square region has side length `r`. To find which region a point belongs to we use the following line:

```
Region key(std::floor(point.x / r), std::floor(point.y / r))
```

where `key` represent the region which the point `point` is part of.

After this we are interested in finding the average point for each region that contains at least one point.

1. Create an appropriate container which can store the regions that has currently been found. For region we must keep track of which points is placed within it.
2. Iterate all points and place them in the region they belong to, based on the key given above.
3. For each region, sum the x-coordinates of all points and divide by the number of points. Do the same for all y-coordinates. Print the resulting x- and y-values to the terminal.

Hints

- You will need to use more than one type of container.
- You need to add at least one operator to the type `Region` to ensure that you chosen container will work properly. Note that this operator must take both `x` and `y` into account.

Non-functional requirements:

- You must use appropriate STL containers to store the regions.

Functional requirements: Execution of the program should produce the following output in the terminal:

```
(2.5, 2.5)
(8, 3)
(11.7, 12.84)
(10000, -5000)
```