Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Informatik for CSE
Autumn Semester 2018

Exercise Organization and Submission Environment

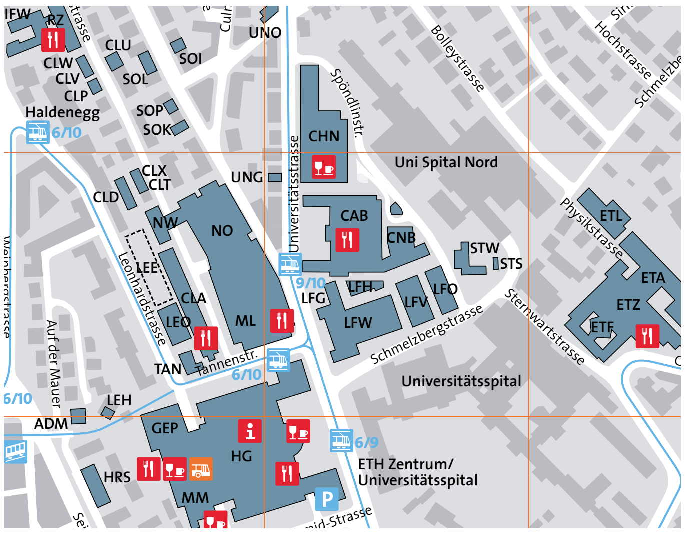
Exercise Subscription

Once you have registered for the course in mystudies, you can enroll to an exercise group. To do that, please go to `https://expert.ethz.ch/enroll/AS18/infcse`, log in with your ETH account and then choose one of the listed exercise groups. Enrolling takes place on a first come, first served basis.

Should you face any technical problems, please contact the course's head assistant, Francois Serre, `serref@inf.ethz.ch`.

Exercise Times and Places, Site Map

Wed 10:15 - 12:00		
Manuel Kaufmann	CAB G 56	de
Robin Worreby	LFW B 3	en
Sebastian Balzer	LFW E 13	de
Mon 13:15 - 15:00		
Roger Barton	LFW E 13	de



Exercise Organisation

- Exercises will be made available online via online submission system [code]expert (`https://expert.ethz.ch`).
- You can work on an exercise for about one week after you have looked at it in the exercise groups (the *submission period*).
- You submit all your solutions via [code]expert.
- Your programming solutions are checked and automatically graded by the submission system. You will get an immediate feedback indicating the correctness of your solution. Your assistant has access to your submissions, will check your solution and give feedback. Solutions to theory exercises will manually be graded by your assistant within the system.
- The assistant will look at the final submission only. A submission is final if it was the last submission within the submission period of an exercise.
- The assistant has the right to override the points provided by the auto-grader, for example if the auto-grader failed to detect that you did not follow the exercise instructions.
- Within the submission period you can always resubmit.
- In order to submit a program, it has to compile without errors.

- Programs should only contain language constructs that have been discussed in the course or exercise sessions. This may sometimes be harder if you already have some programming experience – it is nevertheless mandatory (and beneficial for you) to respect this rule.
- Program texts are written by and for humans, even if they can be understood by machines. Therefore you must pay attention to the visual appearance of your programs. In particular, this requires commenting as well as indentation and use of spaces, e.g. before and after operators and keywords. If your coding style is unacceptable in this sense, we reserve the right to deduct points.
- The assistants are asked to correct your submitted exercises within a week.

Exam and Bonus Points

During the semester you can collect bonus points for the exam by submitting solutions to specially marked bonus exercises. You can achieve points for each of these exercise, and the points collected during the semester will be summed up. In order to be able to solve the bonus exercises, you might need to unlock them by collecting experience points (XP) solving the other weekly tasks, which do not count towards the final grade. The achieved grade bonus is proportional to the achieved points over all bonus exercises, with a maximum of 0.25.

Bonus Points Rules

1. **No cheating allowed.** While we encourage *discussing* exercises in groups, each submission must represent the work of a single student. If we discover, that two or more students submitted the same solution, or that you submitted a solution from a previous year, you will lose the complete bonus grade: your assistant will set the achieved points to zero for all exercises of the complete course.
2. **Follow the rules**. Certain exercises have specific rules, e.g., exercise 1.2 does not allow to use multiplications, loops, or recursion. Submitted solutions that do not follow these rules do not receive any points (set to zero manually by your assistant). This includes the rule that your solution can only contain language constructs that have been discussed during the course or the exercise sessions.
3. **No hard coded results**. Solutions that are not solving the given task but only the given test cases count as failed. Your assistant will reduce the score for such hard-coded solutions.

The Programming and Submission Environment

We use a web-based integrated development environment (IDE) for the exercises. You can work at any computer providing internet access. Note that ETH features a lot of computing rooms with public computers that you can use to execute the exercises.

Of course, you can also use your own computer, if available. If you intend to buy a (new) notebook you may want to check Neptun, a platform for ETH members providing good conditions (`http://www.neptun.ethz.ch/`).

If you intend to work with your own laptop, then bring it along to the first exercise hour. Goal of the first exercise session is that everyone has access to and knows how to use the web-based IDE and the submission system.

For this lecture, we use [code]expert, an online submission system and web-based integrated development environment (IDE).

You log into the submission system using the following URL: `https://expert.ethz.ch`. To login, use your regular ETH credentials (nethz username and password).

Note: The submission system is still young. The IDE is even brand-new. If you encounter issues with the system, please report them. We are in direct contact with the developers. Depending on the issue we may be able to fix it even before the course ends, so you can still benefit. Please report issues using the email contact address on the lecture homepage.

We have a daily maintenance window from 08:00 - 09:00 (morning). This window allows us a timely installation of bugfixes. During this time the system may not be always available, but these down times are usually short. If the system is not available during the maintenance window, simply try again after a few minutes.

Performing a programming task

For the first login, use the link `http://expert.ethz.ch/enroll/AS18/infcse`. This link allows you to enroll in the exercise group.

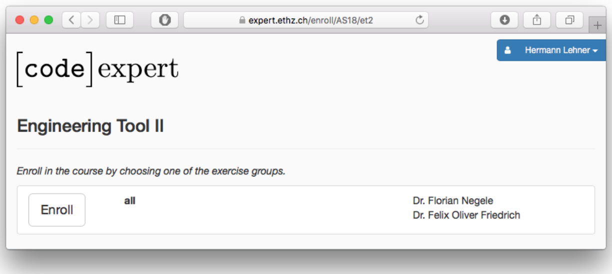
Performing a programming task is generally done in the following steps:

1. Open the programming environment (via exercise link or overview)
2. Write the program as requested in the exercise.
3. Test the program manually using your own inputs (compile & run).
4. Use the automated tests.
5. Submit. Remember, you can submit multiple times.

Of course, you can go back to a previous step if you are not satisfied with the result. The following sections explain the above steps in more detail.

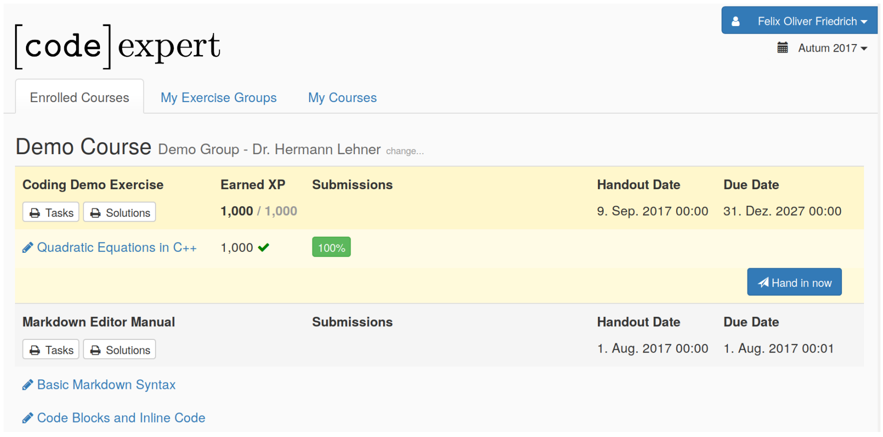
Some of the tasks will not require any coding. In this case, the IDE turns from a coding environment into a relatively simple text editor to write your answer. Within this editor you can also upload attachments. For example you can upload screenshot images if asked in the task description.

Enrolling in the Online Submission


expert.ethz.ch/enroll/AS18/et2
Hermann Lehner
[code]expert
Engineering Tool II
Enroll in the course by choosing one of the exercise groups.
Enroll
all
Dr. Florian Negele
Dr. Felix Oliver Friedrich

The first time you log into the system, and select a programming task of this course you are automatically enrolled in the online exercise submission of this course. Before you can start working on the task, you have to find and select your exercise group.

The Overview Page


[code] expert

Felix Oliver Friedrich
Autum 2017

Enrolled Courses | My Exercise Groups | My Courses

Demo Course Demo Group - Dr. Hermann Lehner change...

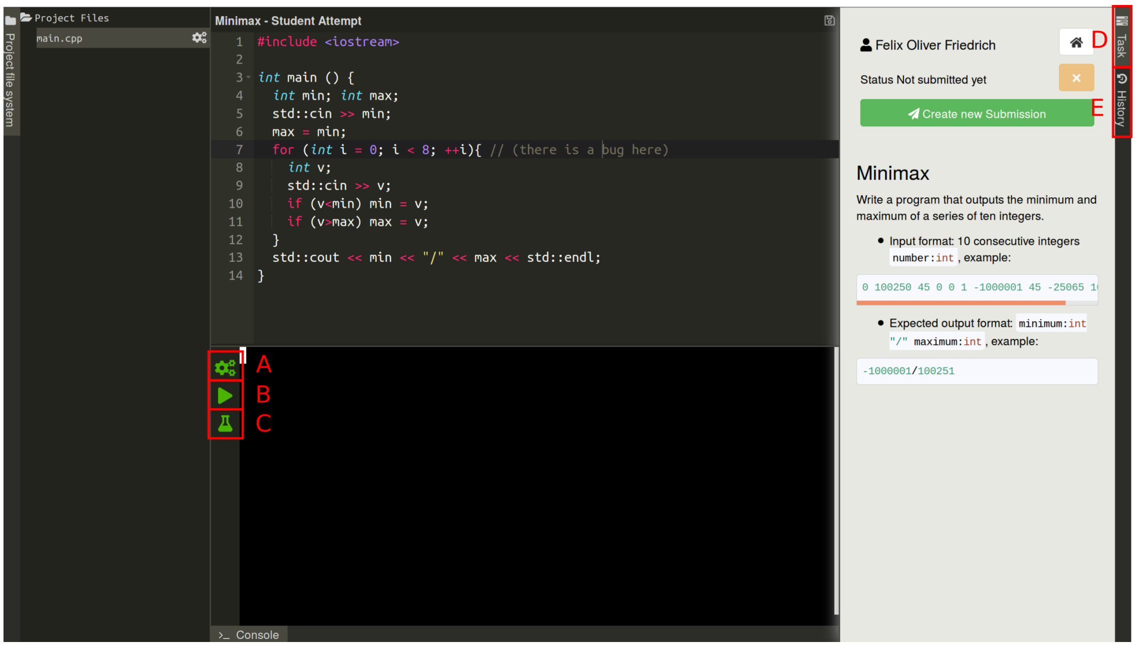
Coding Demo Exercise	Earned XP	Submissions	Handout Date	Due Date
Tasks Solutions	1,000 / 1,000		9. Sep. 2017 00:00	31. Dez. 2027 00:00
Quadratic Equations in C++	1,000 ✓	100%		
				Hand in now

Markdown Editor Manual	Submissions	Handout Date	Due Date
Tasks Solutions		1. Aug. 2017 00:00	1. Aug. 2017 00:01
Basic Markdown Syntax			
Code Blocks and Inline Code			

The overview page shows the available exercises and your progress on completion of the programming tasks. You reach the overview page by selecting overview in the top right menu or if you login to the system via `https://expert.ethz.ch`. In the overview, you can see the course that you are enrolled in and your group assignment. For each exercise you can see the handout date and the due date. The due date is not strictly enforced by the system to allow exceptions (e.g., late hand-ins due to sickness). In the exercise section you see the links to the programming tasks. Clicking on a programming task link opens the programming environment. For each programming task, the overview page shows its submission status. Section *Submission* contains an overview of the different submission statuses.

Programming

By clicking on a programming task link in the overview page you will open the specific task in the programming environment. The screen looks like this:



```
#include <iostream>

int main () {
  int min; int max;
  std::cin >> min;
  max = min;
  for (int i = 0; i < 8; ++i){ // (there is a bug here)
    int v;
    std::cin >> v;
    if (v<min) min = v;
    if (v>max) max = v;
  }
  std::cout << min << "/" << max << std::endl;
}
```

Felix Oliver Friedrich

Status Not submitted yet

Create new Submission

Minimax

Write a program that outputs the minimum and maximum of a series of ten integers.

- Input format: 10 consecutive integers `number:int`, example:

```
0 100250 45 0 0 1 -1000001 45 -25065 1
```

- Expected output format: `minimum:int "/" maximum:int`, example:

```
-1000001/100251
```

The file overview to the left lets you select a file to edit. The file is then opened in the editor. The name of the file that is important for the programming task is usually mentioned in the exercise task description. In the editor, you will write the actual code. If you have multiple files open, you can switch between them by using the tabs above the editor. After you wrote your program, you have to compile it (A). Compile time errors are listed in the output area. If your program compiles without errors, you can execute it by clicking on the run button (B). To test your solution, you can run it against a set of predefined test inputs. This is done by pressing the test button (C).

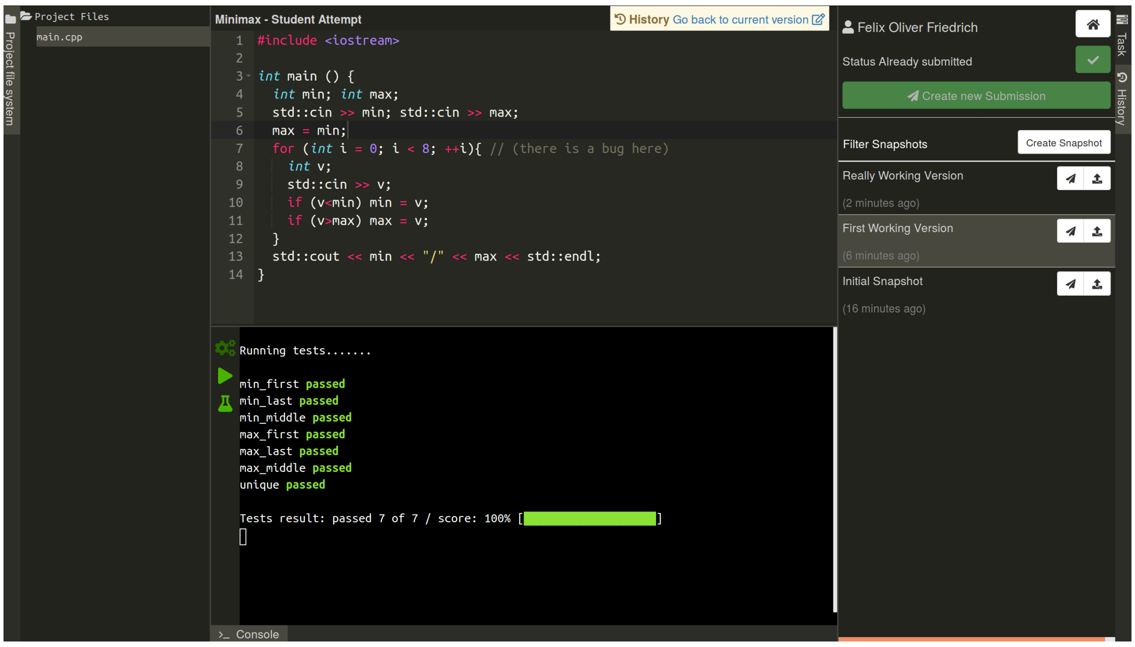
During program and test execution, the terminal shows the output of your program.

Use return or the send button to send the input to your program.

The task area shows the programming task you are currently working on, the result of your previous submissions, and a short description of the task. With the task rider (D) to the right you can switch on / off the task description. With the history rider (E) to the right you can switch to the history.

Where is the save button?

- The file system is transaction based and is saved permanently (autosave). When opening a project it is found in the most recent observed state.
- The current state can be saved as (named) *snapshot*. It is always possible to return to saved snapshot.
- The current state can be submitted (as snapshot). Additionally, each saved named snapshot can be submitted.



```cpp
#include <iostream>

int main () {
  int min; int max;
  std::cin >> min; std::cin >> max;
  max = min;
  for (int i = 0; i < 8; ++i){ // (there is a bug here)
    int v;
    std::cin >> v;
    if (v<min) min = v;
    if (v>max) max = v;
  }
  std::cout << min << "/" << max << std::endl;
}
```

```
Running tests.......

min_first passed
min_last passed
min_middle passed
max_first passed
max_last passed
max_middle passed
unique passed

Tests result: passed 7 of 7 / score: 100% [                    ]
```

Useful keyboard shortcuts

The following table lists some useful keyboard shortcuts for the text editor:

Combination	Action
Ctrl-/	Toggle comment
Tab	Indent
Shift-Tab	Outdent
Ctrl-Z	Undo
Ctrl-Shift-Z, Ctrl-Y	Redo
Ctrl-F	Find
Ctrl-H	Replace
Ctrl-K	Find next
Ctrl-Shift-K	Find previous
Ctrl-L	Go to line

Tests

The system tests your program using multiple test cases. Each test case consists of an input and an expected output. Your program passes a test if it produces an actual output for an input that matches the expected output of the test case. The comparison is quite tolerant. It is performed line by line and ignores redundant lines such as "Please enter a number:". Further, the comparison ignores white space such as spaces and newlines and the casing of words.

The result of a test case can be:

- **Correct.** Your program returns the correct answer, i.e., the actual output matches the expected output.
- **Wrong Answer:** Your program does not return the correct answer, i.e., the actual output does not match the expected output.
- **Program Error:** The program either crashed (e.g. division by zero), or returned an exit code unequal to zero.
- **Timeout:** The duration of the test case exceeded the maximal allowed duration. The timeout settings are usually generous and are primarily there to catch endless loops.

Submitting

To submit your programming task, use the submit button in the programming environment. The system automatically enables the tests and uses them to rate your solution according to the score assigned to each test case. Note that your assistant is only required to look at the last submitted version of each programming task, so make sure that your last submitted version is the one you want to be corrected.

A programming task can have the following submission statuses:

- **Not submitted.** This task is not submitted: Not submitted yet.
- **Submitted.** This task is submitted, but not yet reviewed. This is indicated with a percentage showing your score in this task. Example: 40%.
- **Reviewed with comment**. This task is submitted and reviewed. Your teaching assistant left a comment. This is indicated by a small mail icon on the right hand side of the score. Example: 80% ✉.
- **Reviewed without comment**. This task is submitted and reviewed. Your teaching assistant did not leave a comment. This is indicated by a small eye icon on the right hand side of the score. Example: 100% 👁.