Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

**Prof. Dr. Friedrich, Dr. Lenzner, Boockmeyer, Neumann, Stangl**
**Sommersemester 2017**

# Woche 05 – (Adv.) Competitive Programming

**Abgabe 22.05.2017 17:00 Uhr, über das Judge-Interface**

**Aufgabe 1** (assignments)**.** (100 Points – 2 second timelimit)

Professor Friedrich wants to design the perfect assignments for his students. To do that, he takes feedback about the past assignments into account when designing a new one. Unfortunately for him, the students are not very detailed with their feedback: They either tell him that the assignment ist hard, or, that it is too hard.

After a few semesters, Prof. Friedrich has worked out a few things about the way students give their feedback:

- No matter how good the students are, he can always come up with an assignment that they consider too hard.
- If they consider an assignment too hard, then any harder assignment will get the same feedback. The same goes the other way: If an assignment is only considered hard, any easier assignment will also only be considered hard.
- Their opinion does not change over the course of the semester. If you give them an assignment with the same difficulty as one given out before, you will get the same feedback.

He does not want to scare his students away, but still wants to challenge them. With that in mind, he considers the ideal difficulty for an assignment to be one that the students consider too hard, but not by much. Any easier difficulty should only be considered hard.

For this semester, this method will now be used for all courses of the chair. Your task will be to write a program that selects the difficulty for assignments. To make your job easier, Prof. Friedrich has developed a method to represent the difficulty of assignments as natural numbers. He is *very* precise in his rating, his difficulty scala goes up to $2^{64}$ (exclusive). However, he does not want to share with you how the rating works. Your success will be based on how many assignments you need to hand out to find the ideal difficulty for each course.

**Input/Output** The first line contains $c$ ($0 \leq c \leq 100$), the number of courses your program will be used for. After that, you should handle the courses one by one.

Now you can hand out an assignment with a specific difficulty by writing a line containing the letter `a` followed by the chosen difficulty to the output[1]. After doing that, you receive a line of input containing the students response. It contains either the letter `h` for hard, or the letter `t` for too hard.

When you are sure that you have found the ideal difficulty for this course, you can write the letter `i` followed by the difficulty to the output. After that, you start selecting assignments for the next course (until all courses have been processed).

**Points** There are two groups of test sets:

- *easy:* The first group worth 30 points only contains courses for first semesters, so all difficulties should be smaller than 256.
- *hard:* For the second group of courses worth 20 Points, there are no additional assumptions.
- *actually-hard:* For the third group of courses worth 50 Points, there are also no additional assumptions.

**Judge** With this kind of task the judge results have somewhat different meanings:

- **No-Output:** Your program failed to find the ideal difficulty for the first course, probably due to invalid output.
- **Timelimit:** This task still has a timelimit to avoid invalid program blocking the judge. The limit is generous enougth that this should only happen to invalid programs or ones that take way to many tries to find the ideal difficulty. If you think your program is correct but still hit the timelimit, contact us on the forums.
- **Wrong-Answer:** Your program either reported a wrong ideal difficulty, or took too many tries for at least one course.
- **Run-Error:** Your program produced invalid output (not according to the format). This could also mean that the the oracle that writes responses to your program crashed. If you suspect that this happened, contact us on the forums. We are working on making that case a different result.

[1] To make sure that the difficulty has been written, the output has to be flushed. The easiest way to do this with standard C++ IO is with `std::endl`, for example: `std::cout << difficulty << std::endl`. Replace `std::endl` with `std::flush` if you want to flush without writing a newline character.

### Sample interaction[2]

```
--> 1
<-- a 0
--> h
<-- a 1
--> h
<-- a 2
--> t
<-- i 2
```

[2] `-->` signals that a line of input was send to the difficulty selection program. `<--` signals a line written by the difficulty selection program. In this example, the program took 3 assignments to find the ideal difficulty.