Programming Expertise
University of Potsdam SS 2023
Detlef Groth, Christian Kappel
Test-Exam July 13th, 2023

You have 90 minutes time for the implementation of the programming tasks. Make after the steps 1-3 intermediate versions: GooboParser1.cpp, GooboParser2.cpp and GooboParser3.cpp . Submit at the end of the exam those single versions to the exams folder on Moodle or if this does not work by E-Mail preferentially as zip archive (surname.zip) to me (E-Mail: dgroth@uni-potsdam.de) - or USB stick. 75% of the tasks will be on correctness and 25% of the tasks will be weighted by usefulness and clearness of the implementation.
The theory questions in 4-6 are to be answered first without any aids and the sheet with the answers is to be handed in after about 15 minutes. For the computer tasks 1-3 all aids are allowed during the exam. This does not include personal support from fellow students or other persons.

Clarification

With this I state, that I will not take and give any not allowed support during the exam.

Name, Matrikel-Number.: Signature:

1. 2 points (layout console application)
2. 4 points (implementation of console application)
3. 3+1 points (advanced application tasks)
4. 2 points (theory – C)
5. 2 points (theory – C)
6. 2 points (theory - C++)

Sum: 15 points

Good luck !!

1. Layout application, command line arguments (2 points)

The Gene Ontology is a curated vocabulary to annotate genes and their products. It undergoes constant evaluation and fixing problematic terms. These entries get the line *is_obsolete: true* and in many cases alternative GO ids are suggested using a *consider* line. Create the basic outline of a console application with main function, help function and checking of command line arguments. Save the possible command line arguments in variables or use a command line processor like argparse. If the right number of arguments was not given call the help function and exit the application. The three arguments and the optional forth argument should be: an option name like '--consider-table' or '--obsolete-stats' a goobo input filename (for the GO-file) and an optional third argument for a namespace like 'molecular_function' Check if the given filename points to a valid filename, check if the given command name is a valid one and that the optional third argument is a valid namespaces (molecular_function, cellular_component or biological_process). The help message should explain the application use clearly.

C++ filename first task: ________________________________

2. Opening Obofile and creating a consider table (4 points)

Our program should work with any goobo file. Please don't hardcode the filename in your application. Implement a *considerTable* function which returns a tabular output for all obsolete GO terms and their offered alternative GO id(s) which are given on consider lines in the GO obofile. If there are no alternative GO's give NA's. The function should as well allow to display only a subset for a given namespace. If the user gives the for example the three arguments: --consider-table *filename.obo molecular_function* on the command line, then the go-obo file is parsed and only information for go-ids belonging to this namespace should be displayed to the terminal. If the namespace is not given all ids should be displayed which are obsolete regardless of the namespace. The function should itself not print the data to the terminal but it should return the results for instance as a nested vector. Hints: If your code is very slow, use limit your search to the first 1000 entries first to save your programming time during development. You should un-comment this in your final program. Here a possible example invocation and the output:

```
$ app-name  --consider-table go-2020-01.obo
GO:0000005      GO:0042254
GO:0000005      GO:0044183
GO:0000005      GO:0051082
…
GO:0000020      GO:0045950
GO:0000039      GO:0005324
GO:0000039      GO:0005886
GO:0000044      NA
…
```

Left are the obsolete ID's, right are the new alternative, GO:0000044 has no alternative, consider ID yet.
C++ filename(s) second task: ____

3. Advanced terminal application (3+1 points)

3.1.(1 point): Extend your application as a C++ class (optional, extra point)
3.2 (2 points): Implement the *obsoleteStats* functionality which should show the number of obsolete GO entries and how many of them have alternative consider id(s) and how many not (NA) entries. Hint you might just use the returned vector from the task2 to calculate this. Here a possible output:

```
$ app-name  --obsolete-stats go-2020-01.obo
consider    1217
no-consider 2263
```

3.3. (1 point) Add an optional outfile argument so that the output is not written to the terminal but to the given filename. To distinguish between a namespace argument and a filename you can assume that the user has to give the output filename using a tab extension.

```
$ app-name  --consider-table go-2020-01.obo consider-ids.tab
```

Should write directly the table from task 2 into the file consider-ids.tab