CT30A3370

Project 2: Unix utilities

Documentation

Joonas Liedes

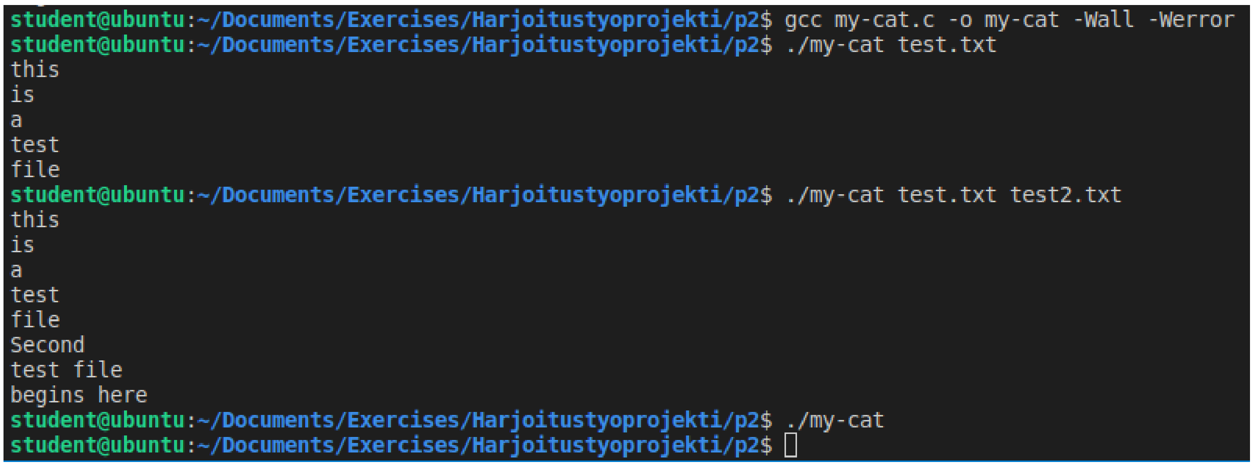
My-cat

The objective of this sub-project was to develop a program resembling the familiar Unix utility cat. The program should read a file (or multiple) and print out the contents. The file names to be read are passed as command line arguments. See more detailed requirements in the project description.

The program was implemented by processing the command line arguments simply in the main method and the file reading was done in a separate function readFile(). Reading the file was done with fgets using a 100 character long buffer, while the printing was implemented with the standard formatted printf() function. Errors regarding file handling were addressed and valgrind was used to check for errors and memory leaks (see summary below).

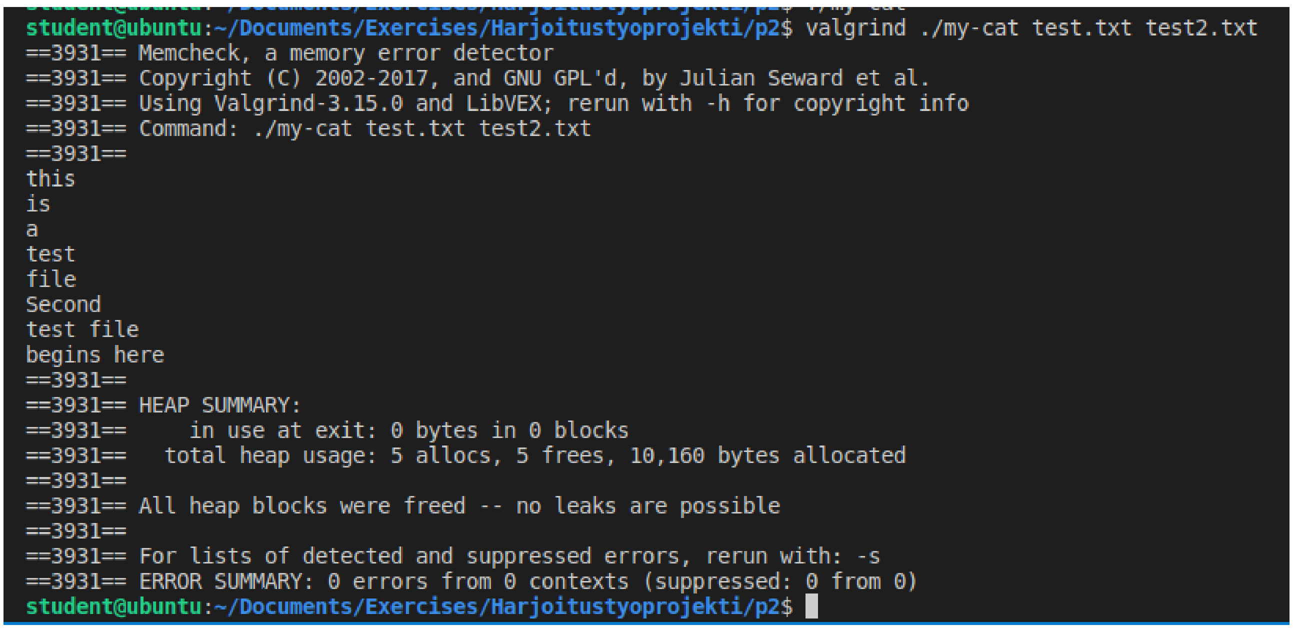
As the project description did not specify anything about line length, a static buffer of 100 characters was used with fgets() for simplicity. For varying line lengths or very long lines there are more sophisticated solutions like getline() (see my-grep).

Screenshot of the programs basic functionalities tested:



```
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ gcc my-cat.c -o my-cat -Wall -Werror
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ ./my-cat test.txt
this
is
a
test
file
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ ./my-cat test.txt test2.txt
this
is
a
test
file
Second
test file
begins here
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ ./my-cat
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$
```

Screenshot of valgrind summary:



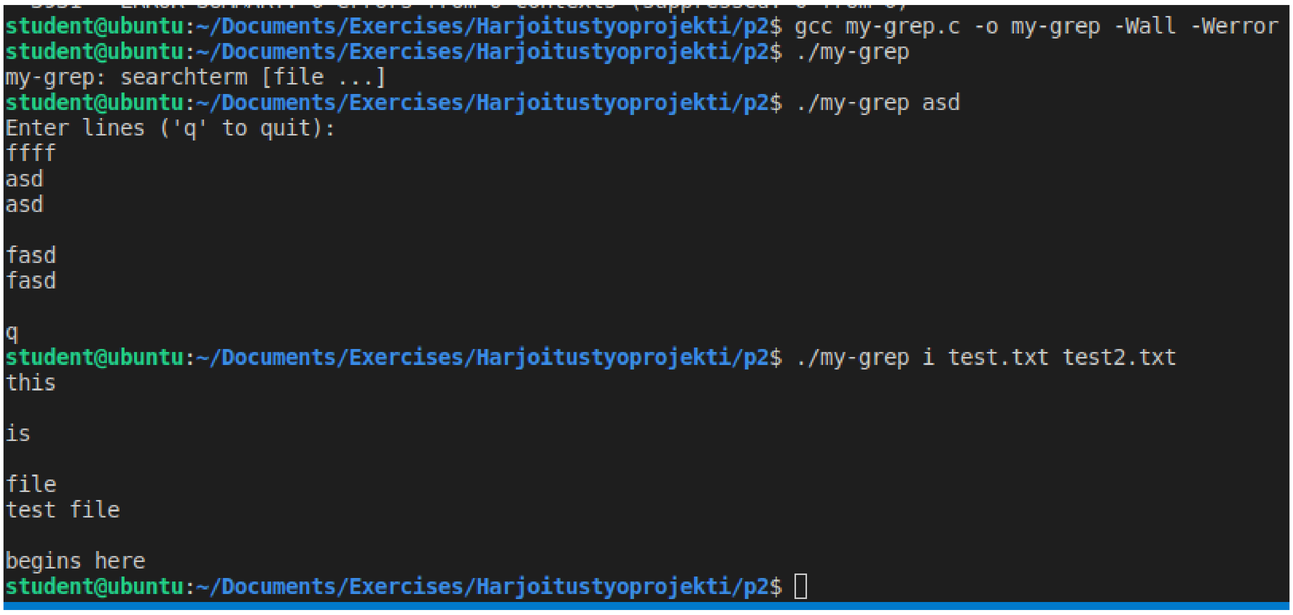
```
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ valgrind ./my-cat test.txt test2.txt
==3931== Memcheck, a memory error detector
==3931== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==3931== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==3931== Command: ./my-cat test.txt test2.txt
==3931==
this
is
a
test
file
Second
test file
begins here
==3931==
==3931== HEAP SUMMARY:
==3931==     in use at exit: 0 bytes in 0 blocks
==3931==   total heap usage: 5 allocs, 5 frees, 10,160 bytes allocated
==3931==
==3931== All heap blocks were freed -- no leaks are possible
==3931==
==3931== For lists of detected and suppressed errors, rerun with: -s
==3931== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$
```

My-grep

The objective of this sub-project was to develop a program imitating the Unix utility grep. The program searches line by line for a user specified search term in a file (or multiple files) and prints the matching contents. The file names to be read are passed as command line arguments. See more detailed requirements in the project description.

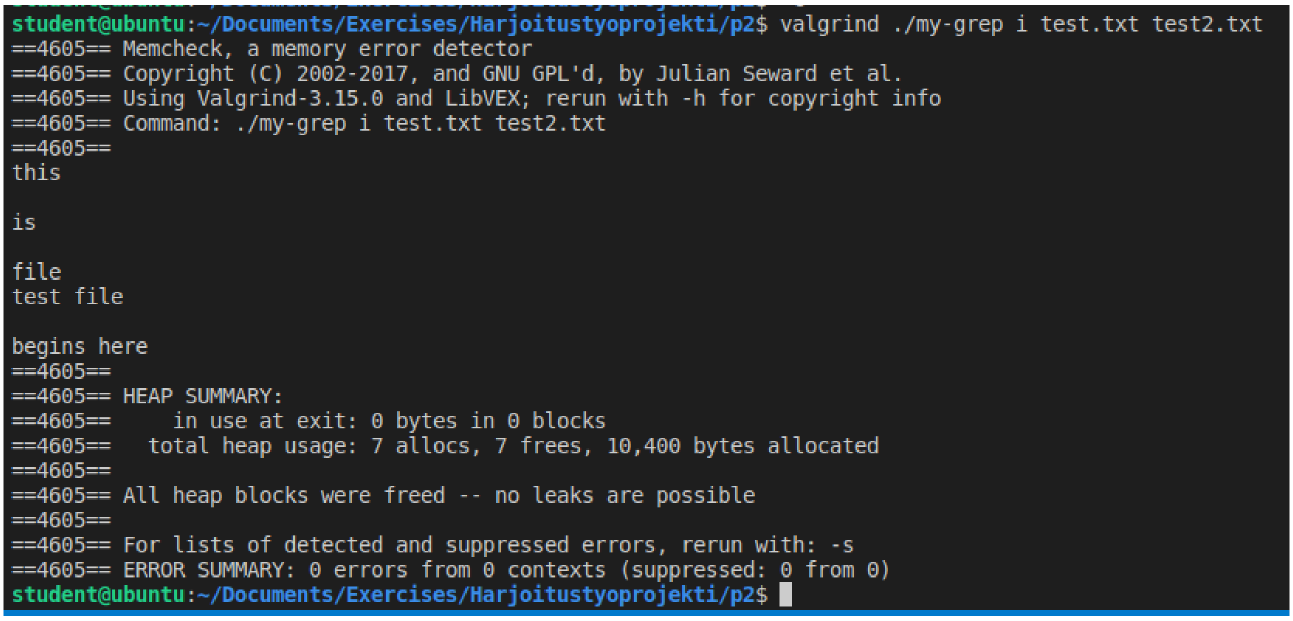
Command line arguments were handled via main method and the file reading functionality in addition with the search was done in a separate function readLine(). For reading arbitrarily long lines with varying lengths getline() was utilized for its dynamic memory allocation features. Strstr() was used for matching the search term with line contents. Errors regarding file handling were addressed and valgrind was used to check for errors and memory leaks (see summary below).

Screenshot of the programs basic functionalities:



```
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ gcc my-grep.c -o my-grep -Wall -Werror
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ ./my-grep
my-grep: searchterm [file ...]
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ ./my-grep asd
Enter lines ('q' to quit):
ffff
asd
asd

fasd
fasd

q
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ ./my-grep i test.txt test2.txt
this

is

file
test file

begins here
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ 
```

Screenshot of valgrind summary:



```
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ valgrind ./my-grep i test.txt test2.txt
==4605== Memcheck, a memory error detector
==4605== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==4605== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==4605== Command: ./my-grep i test.txt test2.txt
==4605==
this

is

file
test file

begins here
==4605==
==4605== HEAP SUMMARY:
==4605==     in use at exit: 0 bytes in 0 blocks
==4605==   total heap usage: 7 allocs, 7 frees, 10,400 bytes allocated
==4605==
==4605== All heap blocks were freed -- no leaks are possible
==4605==
==4605== For lists of detected and suppressed errors, rerun with: -s
==4605== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ 
```

My-zip & My-unzip

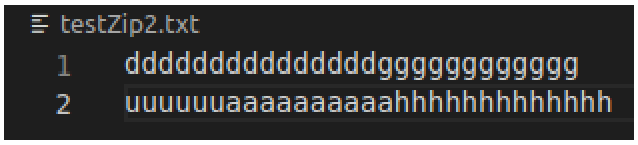
The goal of this sub-project was to develop a compression utility resembling Unix's zip. The compression should be implemented with run length encoding (RLE). In practice this means turning all consecutive characters of the same type into tuples containing the number of consecutive characters and then the character itself. Format for compression was determined as 4 byte integer plus the single character in ASCII. File names to be compressed are provided as command line arguments.

Command line arguments were handled in the main method. File was read using getline() in the readLine method. After each line, the encode method was called to compress the line into the forementioned format using fwrite(). Encode() was implemented simply with a nested loop counting the occurrences of consecutive characters of the same type. The output of the my-zip tool is then redirected to a file of the users choosing via the Unix shells redirection operator '>'. Errors regarding file handling were addressed and valgrind was used to check for errors and memory leaks (see summary below).

The other part of this sub-project was to develop the counter part to the compression called my-unzip. The goal of this program is to decompress the file compressed by my-zip to the standard output. As in my-zip, one or multiple files to be decompressed were handled as command line arguments by the main method. Decompression was implemented in the decode() function using fread(). Errors regarding file handling were addressed and valgrind was used to check for errors and memory leaks (see summary below).

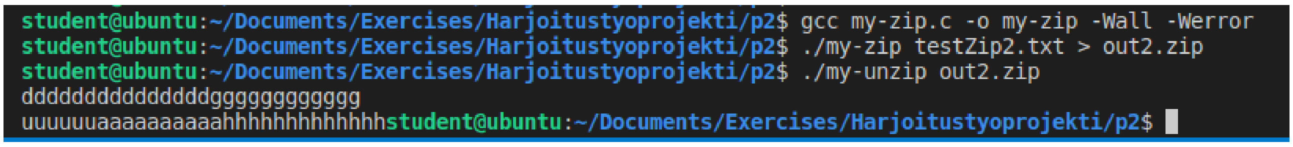
It should be noted that RLE is a very simple compression algorithm and only works well for situations where there are multiple instances of the same character consecutively (like depicted below). The worst case happens when every consecutive character is unique, then RLE results in doubling of the size since every character is followed by the corresponding amount.

Screenshot of the used test file



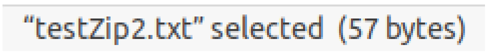
```
testZip2.txt
dddddddddddddddgggggggggggg
uuuuuuaaaaaaaaaahhhhhhhhhhhhh
```

Screenshot of my-zip and my-unzip functioning using above test file:

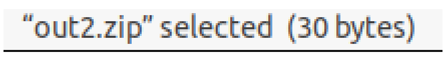


```
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ gcc my-zip.c -o my-zip -Wall -Werror
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ ./my-zip testZip2.txt > out2.zip
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ ./my-unzip out2.zip
dddddddddddddddgggggggggggg
uuuuuuaaaaaaaaaahhhhhhhhhhhhhstudent@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ 
```

File sizes before and after compression:

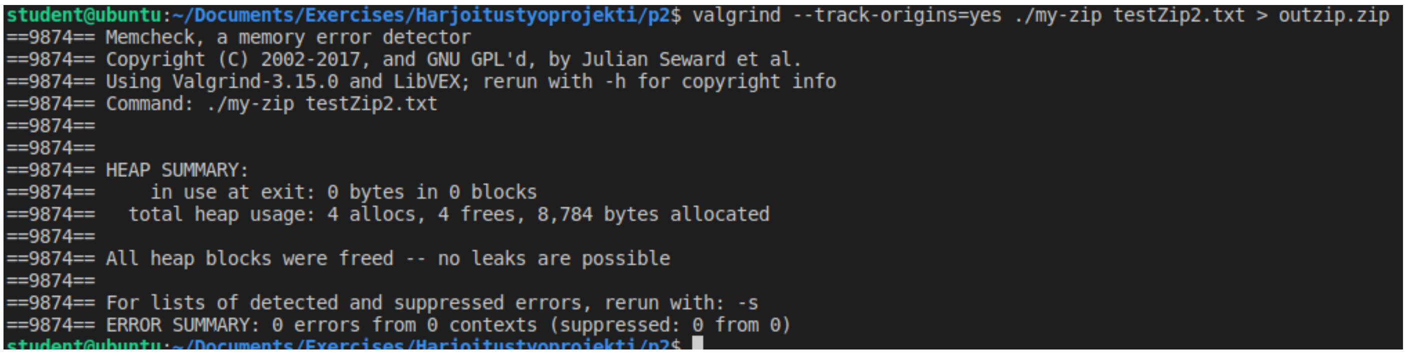


"testZip2.txt" selected (57 bytes)

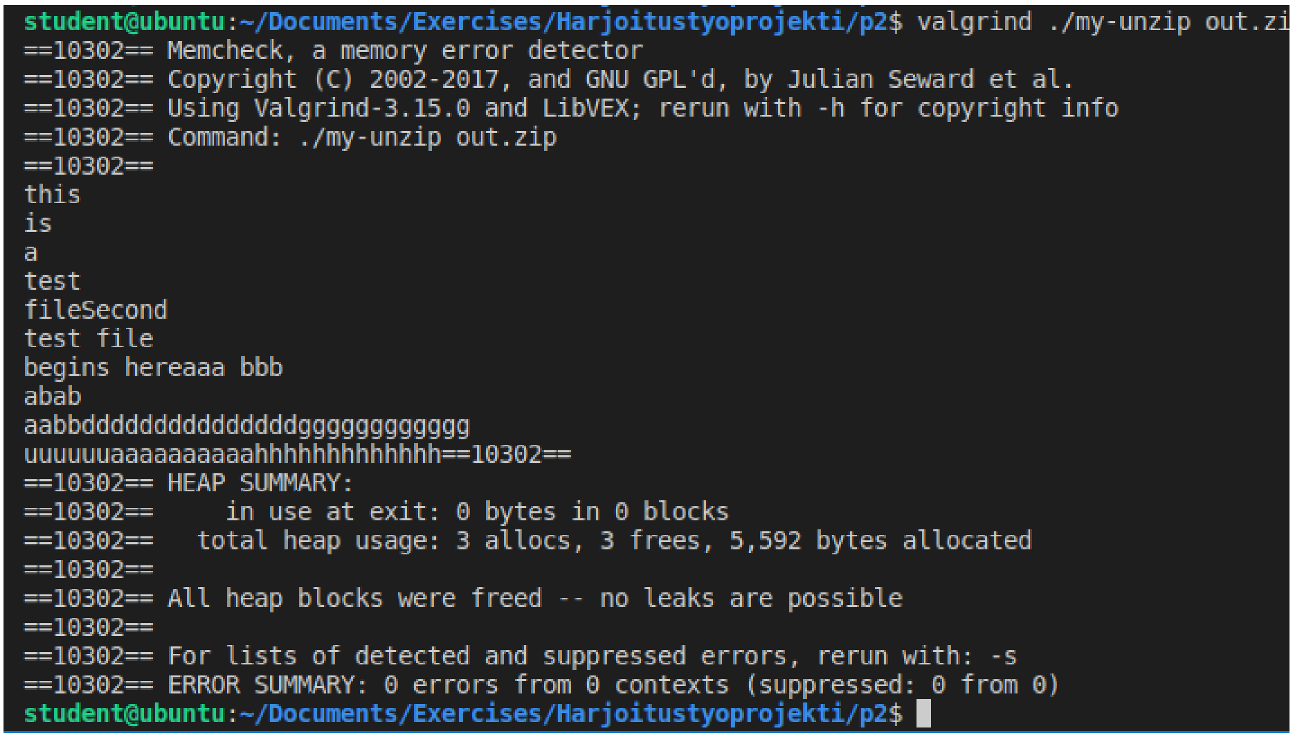


"out2.zip" selected (30 bytes)

Screenshot of my-zips and my-unzips valgrind summaries:



```
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ valgrind --track-origins=yes ./my-zip testZip2.txt > outzip.zip
==9874== Memcheck, a memory error detector
==9874== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==9874== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==9874== Command: ./my-zip testZip2.txt
==9874==
==9874==
==9874== HEAP SUMMARY:
==9874==     in use at exit: 0 bytes in 0 blocks
==9874==   total heap usage: 4 allocs, 4 frees, 8,784 bytes allocated
==9874==
==9874== All heap blocks were freed -- no leaks are possible
==9874==
==9874== For lists of detected and suppressed errors, rerun with: -s
==9874== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$
```



```
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$ valgrind ./my-unzip out.zi
==10302== Memcheck, a memory error detector
==10302== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==10302== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==10302== Command: ./my-unzip out.zip
==10302==
this
is
a
test
fileSecond
test file
begins hereaaa bbb
abab
aabbddddddddddddddddgggggggggggg
uuuuuuaaaaaaaaaaahhhhhhhhhhhhh==10302==
==10302== HEAP SUMMARY:
==10302==     in use at exit: 0 bytes in 0 blocks
==10302==   total heap usage: 3 allocs, 3 frees, 5,592 bytes allocated
==10302==
==10302== All heap blocks were freed -- no leaks are possible
==10302==
==10302== For lists of detected and suppressed errors, rerun with: -s
==10302== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
student@ubuntu:~/Documents/Exercises/Harjoitustyoprojekti/p2$
```