CSCI 3614, Systems Programming
Assignment 1 (27.02.2019)

This assignment contains 5 tasks, giving you 100 points in total. In your GitLab repository create a project named **Assignment 1**. For each task create a corresponding folder (see the name of the folder after the points mark), with a *Makefile* inside. For each task it should be possible to build it using `make` command. Your deliverables are going to be uploaded as a single zip file, including only source files (not the binary files). Upload procedure will be announced shortly.
Also make sure that your code is neat, clean and thoroughly commented. Try to use meaningful variable names.
Deadline: 08.03.2019

1 Concatenation (20 points - concat)

Create a function `char *concat(char *format, ...  /* arguments */)` that accepts variable number of arguments (only integers and strings), and concatenates them into a single string and returns it as the result. Prototype of the function should be placed into **concat.h** header file and the implementation into corresponding **concat.c** file. Use **main.c** only to test your function, i.e. it should not contain anything else.

2 Counter (30 points - counter)

Your task is to implement a C program that is going to read a plain text file and print following information to standard output:

- How many lines does a file have?
- What are the lengths of the longest and the shortest lines?

The name of a file or absolute path to a file will be given as a single command line argument. In case, if the file cannot be found, your program should print an error message to stderr and exit.

Note: For I/O, you are only allowed to use `read` and `write` functions. Make also sure that your program does not read the file character by character and uses a buffer instead.

Also, each launch of your program should add an entry to the end of a log file **counter.log**. Each entry in the log file has a following format:

[date/time] result

where **result** is

- **-1** if file was not found
- **number of lines, length of the longest line, length of the shortest line** otherwise.

3 fwdstdout (30 points - fwdstdout)

Your task is to implement a function `void fwdstdout(char *path)`, which takes a filename as input and switches output destination of the `write(FILENO_STDOUT, ..., ...)` (or `printf`) methods from standard output to the given file. Make sure that you create the file if it does not exist, or clear it completely if it does.

4 Generate a file (10 points - generate)

Implement a program that accepts two command line arguments, a name of a file and its size. As the name of the task states, your job is to generate an empty file of the given size. After compilation, executing the program using the command

```
$ ./generate result.txt 1000
```

should generate an empty file **result.txt** of size **1000 bytes**. Hint: use lseek to achieve desired result.

5 File Descriptor Information (10 points - filedesc)

Your task is to implement a function `int fdinfo(int fd)`, that given a file descriptor as an integer argument, returns file status flags mode and prints information about all the file status flags present to standard output, or returns `-1` and writes a message to standard error, if file descriptor is not active. Prototype of the function should be placed into **filedesc.h** header file and the implementation into corresponding **filedesc.c** file. Use **main.c** only to test your function, i.e. it should not contain anything else.