Objectives:
This assignment is intended to give experience in developing multithreaded programs that require thread synchronization and deadlock prevention. My implementation uses mutexes. So, they were the main focus for this assignment.

How to use:
Makefile commands (assignment spec):
- Executing ‘make’ produces the a4tasks executable file.
- Executing ‘make clean’ removes any unneeded files produced in compilation and the `submit.tar` archive
- Executing ‘make tar’ produces the ‘submit.tar’ archive.

Running the program:
After compiling the a4tasks binary it can be invoked using the command line:

```
./a4tasks inputFile monitorTime NITER
```

Where the arguments are defined as:
inputFile: file describing the tasks to be executed.
monitorTime: integer (in milliseconds) that specifies how often a monitor thread runs.
NITER: integer noting the amount of iterations each task executes before the simulator finishes.

Design Overview:
In general, c++ code is used with functional programing. Code is also separated into separate files to decrease recompile time and create logical code separation.

1. main
   - entry point to the program

2. task.h
   - contains a struct for a task

3. taskManager
   - runs the simulation
   - contains the program logic

4. util
   - handles mutex initialization, lock, and unlock
   - handles pthread create and join

5. parsers
   - parses CLI arguments and files into vars
   - parses vars into strings

Project Status:
The program properly uses threads that take global resources and prints results as

specified in the example. So, the program is complete and adheres to specifications

however due to the nature of the project an exact match was never made to the example output.

If letters are used as the monitor time or as nIter they will be taken as zero.

Testing and Results:
A continuous integration checker was used to ensure that the program always compiled and exited with exit status EXIT_SUCCESS during development.
The example input was run and loosely compared with example outputs. They roughly matched. The program was run with test files of my creation too.

Acknowledgments:

Linux man pages
https://linux.die.net/

mutex initialization, lock, and unlock & delay
http://webdocs.cs.ualberta.ca/~c379/F18/379only/raceC.c