CS4760 **Operating Systems**
Assignment # 1 Due Date: Sept 17, 2019

Important: Please do all assignments on `hoare`

Unix System Calls and Library Functions

The goal of this homework is to become familiar with the environment in hoare while practicing system calls. We will be using `getopt` and `perror` as well as fork().

Your project should consist of one program, which will `fork` off versions of itself to do some file processing. To do this, it will start by taking some command line arguments. You must implement at least the following command line arguments using `getopt`:

```
-h
-i inputfilename
-o outputfilename
```

The -h option should display all legal command line options and how it is expected to run, as well as the default behavior. If input and output filenames are not specified, the defaults should be input.dat and output.dat

Once you have parsed the command line arguments and validated them, then you should attempt to open the inputfile. It will start with a number on a line by itself, with that number indicating the amount of subtasks your program will have to solve using copies of your process created with `fork`. Every other line of the file will contain a subtask, which will consist of a list of integers. An example of this input file is below:

```
3
3 6 107 -8 1 3 7
1 3 -10 5 -35 50 4 25 -20
5 3 -5 7 8 9 1
```

The original process should read the first line of the file. Once it has read that line, it should then go into a loop based on that number, with each iteration of the loop forking off a copy that will then process the next line. Once that child has finished its work (defined below), it will write some data to the outputfile and then terminate. At that point, when the parent detects its child has terminated, it should do another iteration of the loop until done. After all children have terminated. the parent should write the pids of all of its children that it launched, as well as its own pid to the output file.

When a child process starts, it should read the next line of the file. We see in our example file that the first forked child would read the line with 7 numbers.

The task the child must complete with these numbers is the subset sum problem. In particular, your process must find if any subset of numbers in that list sum to 0. If it does, it should output the set of numbers that sums to zero to the output file, starting with its PID. Code to do the subset sum problem can be found at

`https://www.geeksforgeeks.org/subset-sum-problem-dp-25/`

For example, given the first subtask in the above code, possible output might be:

```
13278: -8 1 7
```

Note that the set of numbers might not contain a set of numbers that sums to zero. If the numbers are small, this might be quickly discovered. In that case, output as follows:

```
13278: No subset of numbers summed to zero.
```

However, given a sufficiently large list, this might take too long for our needs. I want your individual child processes to ”give up” after 1 second. In other words, after 1 second of elapsed time since they have started, they should output to the file the following:

```
13278: No valid subset found after 1 second.
```

In addition to the above time limit, your overall project should have a maximum duration of 2 seconds. This should be handled through a timed interrupt signal sent to your master process. If your master process receives this signal, it should terminate all children, close all files and exit. You can find an example on how to do this in our unix textbook, section 9.3, Program 9.7, periodicasterik.c

Please make any error messages meaningful. The format for error messages should be:

```
logParse: Error: Detailed error message
```

where `logParse` is actually the name of the executable (`argv[0]`) that you are trying to execute. These error messages should be sent to stderr using `perror`.

It is required for this project that you use version control, a Makefile and a README. Your README file should consist at a minimum of a description of how I should compile and run your project, any outstanding problems that it still has, and any problems you encountered. Your Makefile should have an option to clean up object files.

What to handin

Handin an electronic copy of all the sources, `README`, Makefile(s), and a log of your revisions using your version control software. Create your programs in a directory called *username*.1 where *username* is your login name on hoare. Once you are done with everything, *remove the executables and object files*, and issue the following commands:

```
chmod 700 username.2

cp -p -r username.2 /home/hauschild/cs4760/assignment2
```

I want to see the log of when your program files were changed. You should check in the files at least once a day while you are working on them. Omission of a `Makefile` will result in a loss of another 10 points, while an insufficient `README` will cost you 5 points. I do not like to see any extensions on `Makefile` and `README` files. Before the final submission, perform a `make clean` and keep the latest source checked out in your directory.

You do not have to hand in a hard copy of the project. Assignment is due by 11:59pm on the due date.