COMP 2004 - Fall 2021
Instructor: Dr Vinicius Prado da Fonseca
Assignment 2

1. (50%) (Textbook 3.21) The Collatz conjecture concerns what happens when we take any positive integer n and apply the following algorithm:

   $$n = \begin{cases} n/2, & \text{if n is even} \\ 3 \times n + 1 & \text{if n is odd} \end{cases}$$

   The conjecture states that when this algorithm is continually applied, all positive integers will eventually reach 1. For example, if n = 35, the sequence is

   35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1

   Write a C program using the fork() system call that generates this sequence in the child process. The starting number will be provided from the command line. For example:

   ```
   ./a.out  8
   ```

   the number 8 is passed as a parameter on the command line, the child process will output:

   ```
   8, 4, 2, 1.
   ```

   Because the parent and child processes have their own copies of the data, it will be necessary for the child to output the sequence. Have the parent invoke the wait() call to wait for the child process to complete before exiting the program. Perform necessary error checking to ensure that a positive integer is passed on the command line.

2. (50%) (Programming problem 3.20 of the textbook with modifications) An operating system's pid manager is responsible for managing process identifiers. When a process is first created, it is assigned a unique pid by the pid manager. The pid is returned to the pid manager when the process completes execution, and the manager may later reassign this pid. Process identifiers are discussed more fully in Section 3.3.1.

   What is most important here is to recognize that process identifiers must be unique; no two active processes can have the same pid.

   Use the following constants to identify the range of possible pid values (for testing these values may be changed, your code must be able to deal with the changes):

```
#define MIN_PID 300
#define MAX_PID 5000
```

You must use a linked list. Read the linked lists pdf and review the code in the course material.

Implement the following API for obtaining and releasing a pid:

`int allocate_map(void)` - Creates and initializes a data structure for representing pids; returns −1 if unsuccessful, 1 if successful. (Hint: review what is meant by an empty linked list. The solution here is trivial, don't overthink)

`int allocate_pid(void)` - Allocates and returns a pid; returns −1 if unable to allocate a pid (all pids are in use)

`void release_pid(int pid)` - Releases a pid

This programming problem will be modified in assignment 3 (Chapter 4) and in Chapter 6.