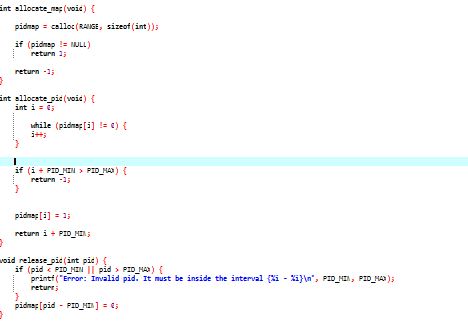
**Description:**

The operating system assigns every process an identification number when it’s started, as a way to tell processes apart without relying on process names. If a program has a high numbered PID, it doesn’t necessarily mean there are a lot of programs running.

We needed a program that assigns a PID to every new running process that enters the operating system and assigns it a PID from the pool of given PIDs, with a maximum number of 1000 PIDs. Now if a process terminates its process and ends, the PID manager will wait for some time and then assign the released PID to some other process that comes up. All the PIDs released will be returned to the same map from where they were allocated.

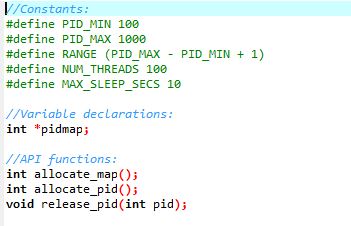
The main task is to make sure that no two active processes have same PIDs.

**Algorithm:**



Here are the three functions we will be using in our program to allocate and release PID to processes. Allocate\_pid will allocate PIDs to the new processes, while the allocate\_map and release\_pid will handle the case of releasing Ids by processes.

**Code snippet:**



Here, I needed to define another header file to use in our main program, where I have defined all the constraints that were asked in the problem.

**Boundary Conditions:**

MIN PID 100

MAX PID 1000

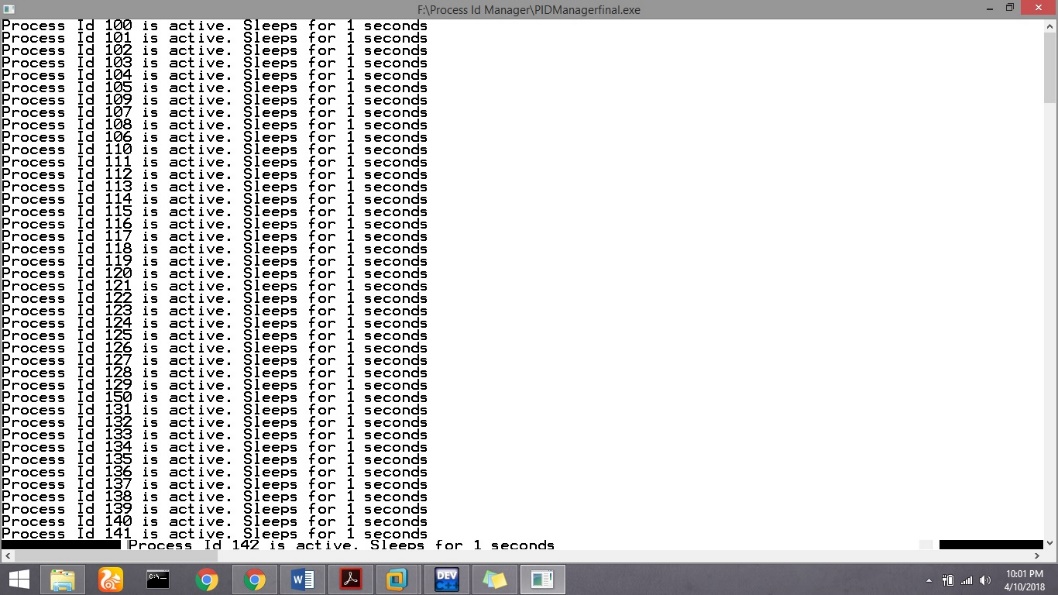
int allocate map(void)—Creates and initializes a data structure for representing pids;

returns—1 if unsuccessful, 1 if successful

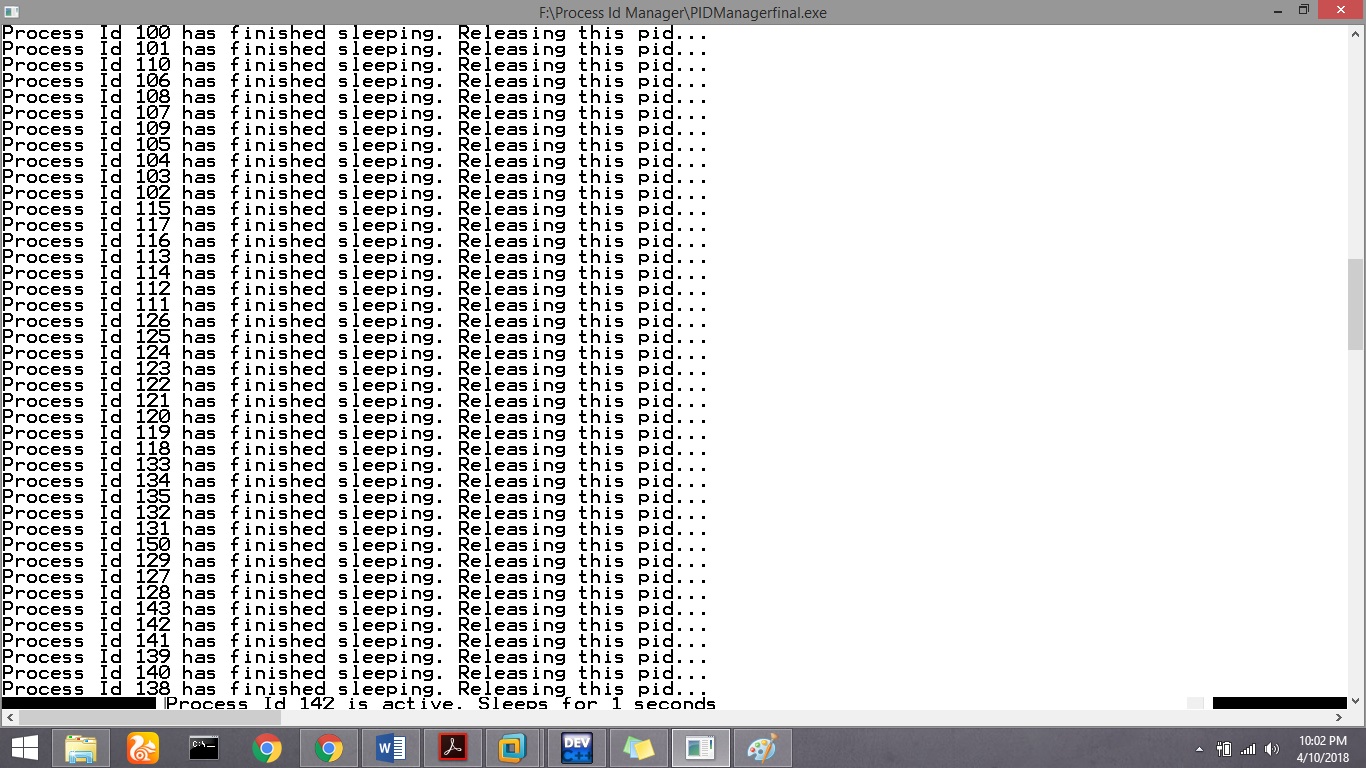
int allocate pid(void)—Allocates and returns a pid; returns— 1 if unable to allocate a pid

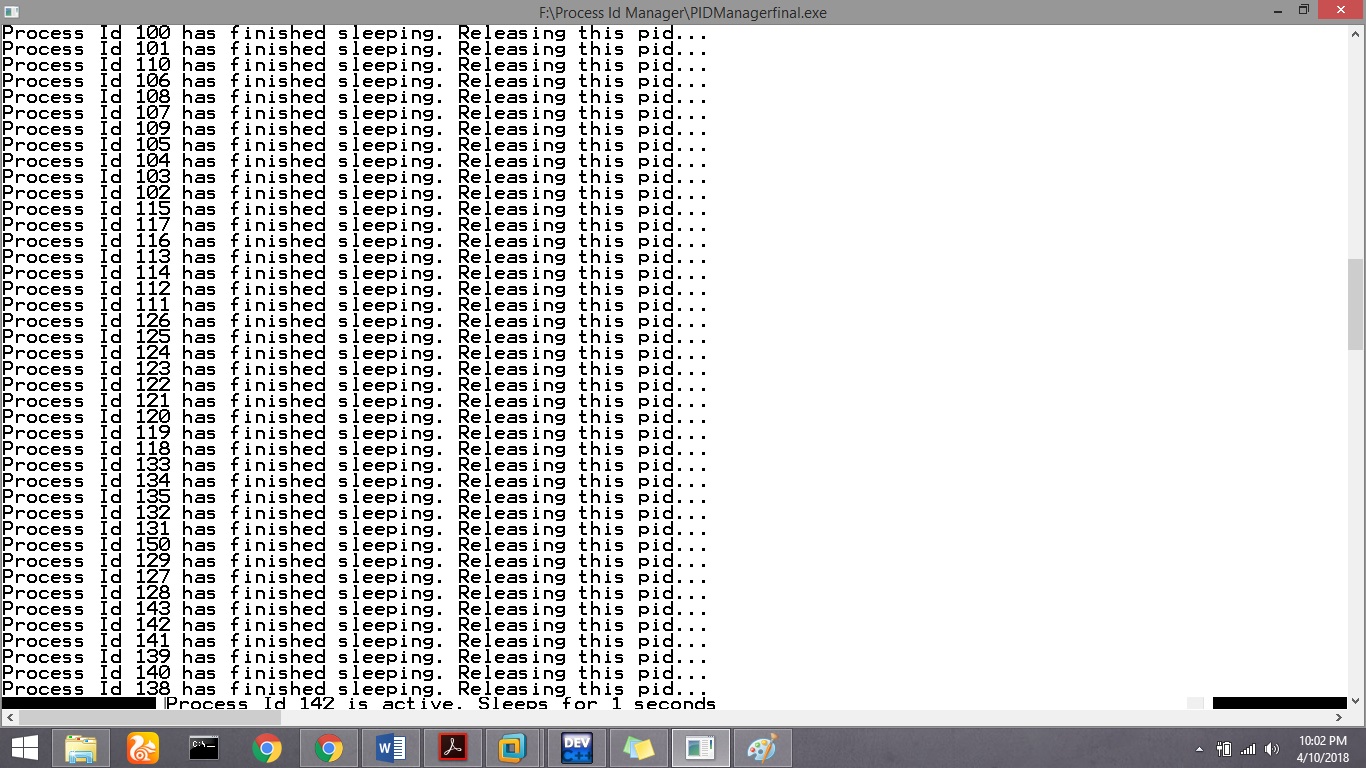
(all pids are in use)

void release pid(int pid)—Releases a pid

**Test cases:**

Here we initiate the program with assigning PID values to 100 processes.

2.

3.

**Remark:**

**No, I have not commited on github for 5 weeks before the submission date.**