Hisham Juneidi

203-979-8580 - hishamj6@vt.edu - github.com/HishamJuneidi
linkedin.com/in/hisham-juneidi-590828170

OBJECTIVE

Obtain a summer position or internship in 2019 to add to my skill outside of my classroom academia

EDUCATION

B. Sc. Computer Science, Minor in Mathematics **Grad: May 2020**
Virginia Tech

TECHNICAL SKILLS

Java (Very Proficient), **Git/Github** (Very Proficient), **C** (Proficient), **Swift** (Proficient), **Angular 6** (Proficient), **JavaScript** (familiar), **Systems Programming** (Threading, Processes, Mutexes, semephors, Signals, linking)

WORK EXPERIENCE

Cellink: Software Engineering Co-op (Fall of 2018)

- Create Google test using Google's C++ framework to test all the functions Report any unexpected
- behavior for the Cellink's Bio X printer

MedicaSoft: Software Engineering Internship (June 2018 – August 2018)

- Designed and implemented a front-end to properly transmit user information to MedicaSoft's AWS servers. Handled obtainment validity of usernames, passwords, email addresses, zip codes, phone numbers, and location information.
- Used AWS Lambda and DynamoDB to store login credentials and personal identification data, as well as signal administrative privileges.
- **Technologies Used**: **Angular 6** for designing and implementing our front-end, **Postman** for unit testing.

PROJECTS

Shell Project, Virginia Tech, February 2019

- Implemented a shell in C by forking a child process that handles built in commands like fg, bg, kill, and stop that a user provide in the terminal. Using functions like fork(), exec(), waitpid()
- Handling signals like SIGINT and SIGSTP that the child process make and keep updating the child process status using the process id of the child
- Create pipes that redirect the I/O by opening and closing stdout/stdin file descriptors. Using dup2() to make the pipe connection

Thread pool, Virginia Tech, March 2019

- Creating semaphores and locks to ensure thread synchronizations and avoid race conditions and dead locks. Using pthread_mutex_init(), pthread_mutex_lock(), sem_init(), sem_post(), sem_wait()
- Applying work sharing principle which is adding tasks on a global queue that all the threads remove and execute
- Applying work stealing principle which each worker thread maintaining its own local queue. And when a thread does not have a task, it can steal from the neighboring thread

ACTIVITIES/ EXPERIENCE

Math Tutor and General Chemistry at Norwalk Community College
Member of Programming Team Club
Member of IEEE at Virginia Tech
Volunteer for 100 hours at Stamford Hospital