**Question 1**

After log-in, the name of the computer given to me is Crow.

The process running my command shell is TCSH

**Ps xf 32478: PPID = 32390**

**< 32390 , TCSH > :PPID = 32389**

**< 32389 , SSHD > : PPID = 32379**

**< 32379 , SSHD > : PPID =1066**

**< 1066 , SSHD > : PPID = 1**

**< 1 , Systemd > PPID = 0**

In comparing the info obtained above with the one from the class notes, we can see that the shell is being run through the Open SSH sshd, a daemon program for ssh, connecting to computers over a network hence the remote log in and is just like the diagram including the user commands just below the shell process.

There is no entry for System 0 because it does not have any parent process hence it is the root process. This process starts other processes. Process 0 is implemented with the help of the hardware.

**Question 2**

It doesn’t make sense to make an exec code in a multithreaded code because when the exec call finishes correctly, it doesn’t return to the main code. It just executes and since in a multithreaded code, the address space is still shared including the code and static data and heap. And when the exec call finished, it erases all the data for all the thread because it cleans up the address space.

**Question 3**

The strtok\_r version is necessary for multithreaded code because strtok() function uses a static buffer while parsing the string that is message cannot be reused and is discarded i.e. static. Since both functions return a pointer to the next token and access to the previous string is lost in the next token call, there needs to be a way of keeping track of the individual tokens hence the need for a save pointer. The strtok\_r saves pointer to the individual strings and sub tokens so since a multithreaded code is one shared memory access, different parts of the code can call the function with reference to each save pointer and still be correct.