# CSC 369 Worksheet 6 Solution

August 20, 2020

1. Done. See link here
2. First, I need to find out how much memory is in my system, and how much is free. Running the command `free -m`, we have

```
ubuntu@primary:~$ free -m
              total        used        free      shared  buff/cache   available
Mem:            981         157         131           0         691         668
Swap:             0           0           0
```

   Using this information, I can write that the computer has

   - 981 MB of total memory
   - 131 MB of free memory

   Second, I need to answer if these numbers match my intuition.

   It does match my inution that free memory should be less than total memory.

   **Notes**

   - I should ask professor the type of intution the author was expecting
   - Installing Ubuntu Virtual Machine link: here
   - Start virtual machine by typing command: `multipass start ubuntu-lts`

3. I need to write a little program `question_3.c` (I will create this instead of `memory-user.c` for recording keeping purposes) that uses a certain amount of memory.

   It's critera are:

- The program should take one command-line argument: the number of megabytes of memory it will use.
- The program should allocate an array.
- The program should constantly stream through the array, touching each entry.
- The program should do this indefinitely, or for a certain amount of time also specified in command line.

For it's solution, please referr to `question_3.c`

**Notes**

- Learned that a large array ($> 5000$) can be generated using heap memory [1]
- **Command Line Arguements in C**

```
#include <stdio.h>

int main (int argc, char *argv[])
{

  return 0;
}
```

  - `argc`
    - Is the number of arguments passed to the program
  - `argv`
    - Is an array of strings
    - Each string represents one of the arguements passed to the program

  **Example**

For example, the command line

```
gcc -o myprog myprog.c
```

would result in the following values internal to GCC:

```
argc
     4
argv[0]
     gcc
argv[1]
     -o
argv[2]
     myprog
argv[3]
     myprog.c
```

- **String to integer**
  - **Syntax:** `int atoi(const char *string)`

    **Example**

    ```
    int output = atoi("20") /* Stores 20 in output*/
    ```

**References**

1) Stackoverflow, Allocating A Large (5000+) Array, link