SYST 44288 – OPERATING SYSTEMS & SYSTEMS PROGRAMMING
ASSIGNMENT 1: C PROGRAMMING REVIEW – 6.5%

DUE DATE: SEE SLATE. WORK IN TEAMS OF TWO.

Before starting: Use GitHub for ALL assignments & project

- Commit your code regularly
- When committing code, write descriptive comments in GitHub about what you coded for that particular commit.
- You must include your gitHub link in the comments of the dropbox on SLATE
- **Otherwise you will receive a zero grade**

Submission Requirements

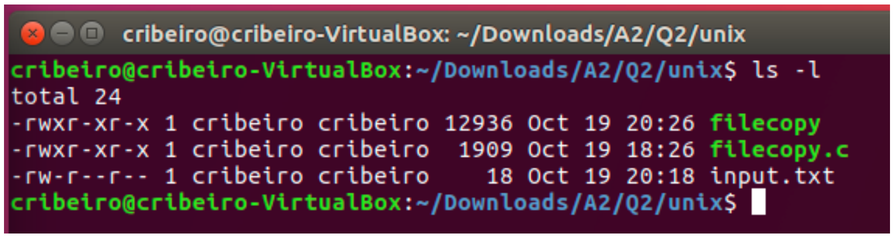
If you do not include the screenshots you may get a zero grade. These requirements are necessary as students have different OSs and your code may not run on my OS. These submission requirements allow me to mark your assignment after the due date if it doesn't run on my OS and avoids students getting a zero grade.

1. **Screenshots for EACH Question**

   Save all screenshots in a single Screenshots.doc file

   A. **Must show the command "ls –l" entered on the command line and output of this command.**

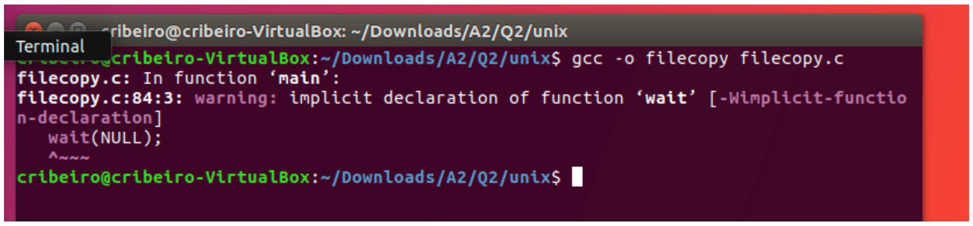
   SAMPLE:



```
cribeiro@cribeiro-VirtualBox: ~/Downloads/A2/Q2/unix
cribeiro@cribeiro-VirtualBox:~/Downloads/A2/Q2/unix$ ls -l
total 24
-rwxr-xr-x 1 cribeiro cribeiro 12936 Oct 19 20:26 filecopy
-rwxr-xr-x 1 cribeiro cribeiro  1909 Oct 19 18:26 filecopy.c
-rw-r--r-- 1 cribeiro cribeiro    18 Oct 19 20:18 input.txt
cribeiro@cribeiro-VirtualBox:~/Downloads/A2/Q2/unix$
```

   B. **Must show compiled without errors, warnings are acceptable.**

   SAMPLE:



```
cribeiro@cribeiro-VirtualBox: ~/Downloads/A2/Q2/unix
cribeiro@cribeiro-VirtualBox:~/Downloads/A2/Q2/unix$ gcc -o filecopy filecopy.c
filecopy.c: In function ‘main’:
filecopy.c:84:3: warning: implicit declaration of function ‘wait’ [-Wimplicit-functio
n-declaration]
   wait(NULL);
   ^~~~
cribeiro@cribeiro-VirtualBox:~/Downloads/A2/Q2/unix$
```

   C. **Must show execution and correct output**

2. **Upload to Dropbox:**
    - Do **NOT** zip the file not any other format of archiving files
    - Do **NOT** include any folders
    - Screenshots.doc
    - 1.1.c
    - 1.2.c
    - 1.3.c
    - 1.4.c
    - You must include your gitHub link in the comments of the dropbox on SLATE, **otherwise you will receive a zero grade**

Rubric & Important Submission Checklist

- **Please see SLATE**
- Add GitHub link in the comment box of the dropbox **(if you do not provide this you will receive a zero grade). You cannot email it later if you forget.**
- Submit screenshots as outlined below **(May receive a zero if not submitted)**
- Submit the source code not just the compiled code
- Submit correct file names as outlined
- Submit to correct dropbox folder on SLATE

TOTAL: 14 marks

C Programming Review: **Taking in command line arguments**

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

argv is an array of pointers of type char (i.e. array of strings). This array holds all of the arguments passed when executing the program. If your program is named 1.1.c, after it has successfully compiled you will execute the program by calling its name and then passing all necessary command line arguments. Ex. ./1.1 3 0 600000

argc is the length of this array.
argv[0] = 1.1
argv[1] = 3
argv[2] = 0
argv[3] = 600000

Complete the following programs in C:

1. (2 points) Armstrong Numbers
Number n is called Armstrong's number of order k if it is equal to the sum of k-th powers of its digits.

For example: $371 = 3^3 + 7^3 + 1^3$ Armstrong's number of order 3. Refer to the Armstrong Numbers research paper provided for details on how Armstrong numbers work.

Write C program which prints all the Armstrong's numbers of a given order k, within the given interval between p and q. You should use functions like pow() provided by the math library.

Input: The user will type the integers k, p, and q on the command line. Taking in one integer at a time, **in this order**.

Output should consist of a single number per line with no extra or blank lines. See next page.

Test Input (k, p, q)	Test Output
3 0 600000	153 370 371 407
4 0 600000	1634 8208 9474
5 0 600000	54748 92727 93084
6 0 600000	548834

Save file as 1.1.c & do not create subfolders for each program

2. (2 points) Let n be a positive integer. Euler's phi-function φ(n) is defined to be the number of positive integers not exceeding n that are *relatively prime* to n.

Write a program that can be used to calculate and display the value of Euler's phi-function for a series of positive integer, separated by spaces, entered on the command line.

Save file as 1.2.c & do not create subfolders for each program

3. (2 points) Using pointers and *avoiding unnecessary local variables*, write a function, *rmchr* that takes a string and a character as command line arguments. The *rmchr* function should remove **all** occurrences of the character from the string. The *rmchr* function should not leave spaces characters in the string where characters have been removed. The resulting string should be returned using an appropriate data type and printed to the command line.

Save file as 1.3.c & do not create subfolders for each program

4. (2 points) Using pointers and *avoiding unnecessary local variables*, write a function, *rmstr* that takes two strings as command line arguments, removing all occurrences of any character from the second string in the first string. The *rmstr* function should not leave spaces characters in the string where characters have been removed. The resulting string should be returned using an appropriate data type and printed to the command line.