ECE 2035 Homework 1

This is a warm-up project whose objective is for you to set up the infrastructure you need for subsequent assignments. Help is available through **piazza**, during **TA office hours** (posted on Canvas) and by sending **email** to ece2035-help@ece.gatech.edu. There are also helpful links to tutorials and primers here.

HW1-1: The goal of this part is to use a Linux/Unix environment and become familiar with its facilities. Toward this end, you must either acquire/access and run an appropriate Linux/Unix distribution that fits your computing environment or connect to a machine in the Klaus 1448 Linux cluster. The following are some the options available:

- Unix underlying Mac OS X
- Linux Bash shell on Windows 10 (Anniversary Update): installation guide
- VMware on Windows 8 or 10: installation guide
- Virtual Box on Windows 8.1
- Red Hat on Linux machines in Klaus room 1448
- Dual booting (e.g., Ubuntu 12.04 and Windows 7)
- Running native Ubuntu (e.g., 14.04.3 LTS)

Once you do this, then perform the following exercises.

1. Create and edit a text file that contains the following:
   - Tell one interesting fact about yourself.
   - Which Linux/Unix distribution are you using (e.g., Ubuntu 14.04.03)?
   - The method you are using to run Linux/Unix (e.g., one of the options above).
2. Find a suitable application to capture a screenshot which shows your text file open in an editor, running under Linux/Unix.
3. Submit your screenshot in **jpeg format** in a file of **size no greater than 200K**.

In order for your solution to be properly received and graded, there are a few requirements.

1. The file must be named **`HW1-1.jpg`**.
1. The file must be less than 200K bytes.
2. Your solution must be properly uploaded to the Canvas site (under Assignments) before the scheduled due date.

Please note that for all solutions uploaded to Canvas, all filenames must be exactly as specified, including proper capitalization and file extension (e.g., HW1-1.jpg, HW1-2.c, etc.).

HW1-2: The goal of this part of the project is to modify a short C program, compile it using the GNU C Compiler `gcc`, and run it. A program shell `HW1-2-shell.c` is provided. You must copy/rename it to `HW1-2.c` and modify it to compute the union of two integer sets, which are declared and initialized as the global variables `SetA` and `SetB`. Your program should print each element in the union (in any order). It should then print the number of elements in the union. Assume that the sets both hold exactly ten integers and that they are "sets" in that there are no duplicate elements within a set.

Be sure to try multiple test cases, but do not change the declaration of the global variables (you should change only their initial values, such as the elements in the arrays to create new test cases).

You should open a "terminal window" to run `gcc` under Linux/Unix (type `man gcc` for compiler usage or look up GCC online documentation on the internet). If you do not have gcc

installed, you can use "sudo apt-get install gcc" (see this article or this one for quick tutorials on apt-get and installing packages).
Note that in the terminal window, you can enter any of the Linux commands (such as `ls`, `cd`, `cp`; for reference see http://ece2035.ece.gatech.edu/assignments/Linux_Cmd_Cheatsheet.pdf).
Use the Linux command `cd` to change your current working directory to the directory in which you placed the shell program. For example,

```
    > cd ~/Documents/2035/hw1
```

or

```
    > cd /mnt/c/Users/Linda/2035/hw1
```

You can list the files in that directory using

```
    > ls -la
```

You can copy a file using `cp` or rename a file using `mv` (move a file to a new file). For example:

```
    > cp HW1-2-shell.c HW1-2.c
```

You can use any of the available text editors normally found on Linux, including `vi, vim,` and `emacs`. Using the text editor of your choice modify the `HW1-2.c` program to compute the span as described above.

Once you write your program, you can compile and run it using the Linux command line:

```
    > gcc HW1-2.c —g —Wall —o HW1-2
    > ./HW1-2
```

You should become familiar with the compiler options specified by these flags.

In order for your solution to be properly received and graded, there are a few requirements.

1. The file must be named **`HW1-2.c`**.
1. Your name and the date should be included in the header comment.
2. *Do not* `#include` *any additional libraries.*
3. In the starting *shell* program, it is especially important not to remove or modify any print statements since they will be used in the grading process.
4. Your solution must include proper documentation and appropriate indentation.
5. Your solution must be properly uploaded to Canvas before the scheduled due date.

HW1-3: The goal of this part is for you to install MiSaSiM, modify a short assembly program `HW1-3-shell.asm`, simulate, test and debug it in MiSaSiM. The MiSaSiM simulator can be installed according to the instructions at http://lindawills.ece.gatech.edu/misasim/. Copy or rename the shell program to `HW1-3.asm` and modify it to compute the union of the two integer sets `SetA` and `SetB` allocated and initialized in the shell program. These sets contain ten elements each. Store the union set in the contiguous memory space allocated starting at label `SetC`. (There are 20 words allocated, but the union might not use all twenty words.) Again, assume that the two sets are "sets" in that there are no duplicate elements within a set.

In order for your solution to be properly received and graded, there are a few requirements.

1. The file must be named **`HW1-3.asm`**.
1. Your name and the date should be included in the beginning of the file.
2. The starting *shell* program should not be modified except for the replacement of the comment "# write your code here..."

3. Your program must store its result at the memory location labeled `SetC` when it returns. This answer is used to check the correctness of your code.
4. Your program must return to the operating system via the **`jr`** instruction. *Programs that include infinite loops or produce simulator warnings or errors will receive zero credit*.
5. Your solution must include proper documentation.
6. Your solution must be properly uploaded to Canvas before the scheduled due date**.**

Honor Policy: In all programming assignments, you should design, implement, and test your own code. **Any submitted assignment containing non-shell code that is not fully created and debugged by the student constitutes academic misconduct.** The only exception to this is that you may use code provided in tutorial-0.zip or in the examples on the course website as a starting point for your programs (http://ece2035.ece.gatech.edu/examples/index.html).