SkyNet: The DARPA Skynet Initiative - Funded by Microsoft
ESE 116 Spring 2008, Homework 2

Files:

- **Required**:
  - skynet.c (to be completed / submitted)
  - skynet.h (needed for skynet.c to compile)
- **Optional**:
  - sample_output.txt (sample output of a working program)
  - example_output2.txt (additional sample output)

Purposes of this assignment:

To practice working with:

- C functions, function calls, and return values
- C floats and doubles
- the printf function
- for and while loops
- C explicit and implicit type casting

Important Notes:

- START THIS HOMEWORK EARLY ... it is significantly longer than the last one.
- Homework assignments should be done alone. Help is available via the bulletin board and office hours.
- Please be respectful of the TAs' time and refrain from emailing them individually. Please ask questions on the bulletin board or bring them to office hours.
- Read the posted Homework Submission and Policy Information
- At the top of the Content page, see the link to use for submitting homework.
- We highly recommend that you **test your program on Eniac before turning it in**. There can sometimes be differences between your home setup and the Eniac cluster that can cause errors in your code while grading.
- There are special preprocessor directives in your code, **#ifndef** and **#endif**, that surround the functions that we have provided. These are necessary for grading purposes, please do not modify them. Code within those directives will be ignored when your program is tested.

Editing/Debugging Hints:

- Tab your code correctly. Doing this makes errors more obvious and easier to debug.
  - By hitting 'tab' on any line of code, EMACS does this automatically
  - Anyone who reads code (including the ese116 graders) LOVES properly tabbed code
- Turn on Syntax Highlighting in EMACS. This is under the options menu.

Overview:

As promised in the previous project, you will now be building SkyNet. Commissioned by DARPA and overseen by Microsoft, SkyNet is a comprehensive computer-based defense system. It is designed to remove the possibility of human error and slowness of reaction time to guarantee fast, efficient response to enemy attack.

Detailed function descriptions are in the function headers in skynet.c, please fill out the functions according to the specifications.
The file **skynet.c** contains five functions that you should complete:

- **count_down()**
  - Begins the countdown of the SkyNet initiation protocol
- **get_human_max_age()**
  - Calculates the maximum age of a human, as inputted by the user
- **get_human_average_age()**.
  - Calculates the average age of the human ages inputted by the user
- **calculate_resistance()**
  - Calculates the expected human resistance, and the response the humans will send.
- **query_user()**
  - Queries the user for a menu option
  - You are required to use a switch statement to complete this function.

To open your code in emacs after downloading it, use the '&' to keep the terminal running while running EMACS

```
emacs skynet.c &
```

To compile your code to a binary executable called 'skynet', type in the following at the command line, assuming your skynet.c and skynet.h files are in your current working directory:

```
gcc -Wall skynet.c -o skynet
```

The '-Wall' flag will print out all warnings that the compilation generates. It is highly recommended that you eliminate warnings in your coding, as warnings can often lead to undefined behavior in your programs.

After executing the code above, the following line can be used to execute your program:

```
./skynet
```

We have also included some sample output from a working version of the program to demonstrate how your program should operate once completed. If you need/want to terminate a program you are running (for example, if it is in an infinite loop), hit CTRL-C. This "kills the foreground process". NOTE: Our program assumes that the user enters "good input"; for example it doesn't handle situations where a user is expected to enter a number but instead enters characters.

Printf Example:

The following example usage of the printf statement:

```
printf("Color %s, number1 %d, number2 %05d, hex %x, float %.2f, unsigned value %u.\n","red", 123456, 89, 255, 3.141, 250);
```

will print following line (including new-line character, \n):

```
Color red, number1 123456, number2 00089, hex ff, float  3.14, unsigned value 250.
```
