Hack 5.0

Computer Science I

Department of Computer Science & Engineering
University of Nebraska–Lincoln

Introduction

Hack session activities are small weekly programming assignments intended to get you started on full programming assignments. You may complete the hack on your own, but you are *highly encouraged* to work with another student and form a hack pair. Groups larger than 2 are not allowed. However, you may discuss the problems *at a high level* with other students or groups. You may not share code directly.

If you choose to form a Hack Pair, you *must*:

1. Both join a hack pair on Canvas (go to People then Hack Pairs)
2. You must both work on the hack equally; it must be an equal effort by both partners. Do not undermine your partner's learning opportunity and do not undermine your own by allowing one parter to do all the work.
3. Turn in only one copy of the code under the individual whose last name comes first (with respect to Canvas).

You are graded based on style, documentation, design and correctness. For detail, see the general course rubric.

Category	Point Value
Style	2
Documentation	2
Design	5
Correctness	16
Total	**25**

Table 1: Rubric

Correctness:

- 9 points for student test cases (1 each for writing *and* passing)
- 7 points for the official test suite, proportionally

Problem Statement

To get some practice designing and using functions, you will create a small library of utility functions by implementing the following functions with the given prototypes and specified functionality.

1. `double degreesToRadians(double degree);` - Write a function to convert degrees to radians using the formula
$$\frac{d \cdot \pi}{180}$$

2. Write a function to compute the air distance between two locations identified by their latitude/longitude.

```
double getAirDistance(double originLatitude,
                      double originLongitude,
                      double destinationLatitude,
                      double destinationLongitude);
```

   The air distance between two latitude/longitude points can be calculated using the Spherical Law of Cosines:

$$d = \arccos\left(\sin(\varphi_1)\sin(\varphi_2) + \cos(\varphi_1)\cos(\varphi_2)\cos(\Delta)\right) \cdot R$$

   where

   - φ_1 is the latitude of location A, φ_2 is the latitude of location B
   - Δ is the difference between location B's longitude and location A's longitude
   - R is the (average) radius of the earth, 6,371 kilometers

   Note: the formula above assumes that latitude and longitude are measured in radians r, $-\pi \leq r \leq \pi$, but the function will expect the latitude/longitude to be in degrees. Latitude should be in the range $[-90, 90]$ and longitude in the range $[-180, 180]$. Negative values correspond to the southern and western hemispheres.

3. An object traveling at a velocity v experiences time dilation relative to a stationary object which is quantified by the Lorentz equation:
$$T = \frac{t}{\sqrt{(1 - \frac{v^2}{c^2})}}$$

where t is the normal amount of lapsed time (stationary object) and T is the dilated time experienced by the traveling object. For small velocities, the dilation is small, but for velocities approaching a *percentage* (on the scale $[0, 1]$) of the speed of light, c, the dilation becomes significant.

For example, at 25% the speed of light, a year for the object traveling would correspond to 1.032796 years at the stationary object (or nearly 12 extra days). A person traveling at high velocity would experience "slowed" time relative to the stationary *frame*.

Implement a function to compute the dilated time given the normal time t (units may vary) and the *percentage* (i.e. $\frac{v}{c}$) of the speed of light.

```
1  double lorentzTimeDilation(double t, double percentC);
```

Instructions

- You are encouraged to collaborate any number of students before, during, and after your scheduled hack session.
- Design at least 3 test cases for each function *before* you begin designing or implementing your program. Test cases are input-output pairs that are known to be correct using means other than your program.
- Include the name(s) of everyone who worked together on this activity in your source file's header.
- Place your prototypes and documentation in a header file named `utils.h` and your source in a file named `utils.c`.
- In addition, implement all of your test cases in a *test driver* file named `utilsTester.c` which should output the expected output, the actual output and a message on whether or not the test case passed. You must have at least 3 test cases for *each* of your functions. You should *not* prompt for input or use command line arguments. Your test cases should be hardcoded in your test driver's `main` function.
- Turn in all of your files via webhandin, making sure that it runs and executes correctly in the webgrader. Each individual student will need to hand in their own copy and will receive their own individual grade.