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. Collaboration is allowed and, in fact, *highly encouraged*. You may start on the activity before your hack session, but during the hack session you must either be actively working on this activity or *helping others* work on the activity. You are graded using the same rubric as assignments so documentation, style, design and correctness are all important. This activity is due at 23:59:59 on the Monday following the hack session in which it is assigned according to the CSE system clock.

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* 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 1.032 years at the stationary object. 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 of the speed of light.

```
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.