Computer Engineering Department

TED University

CMPE 252 C Programming, Spring 2020

HOMEWORK 1

30.03.2020, Monday

DUE DATE: 13.04.2020, Monday , 23.59

Send your C code file via Moodle.

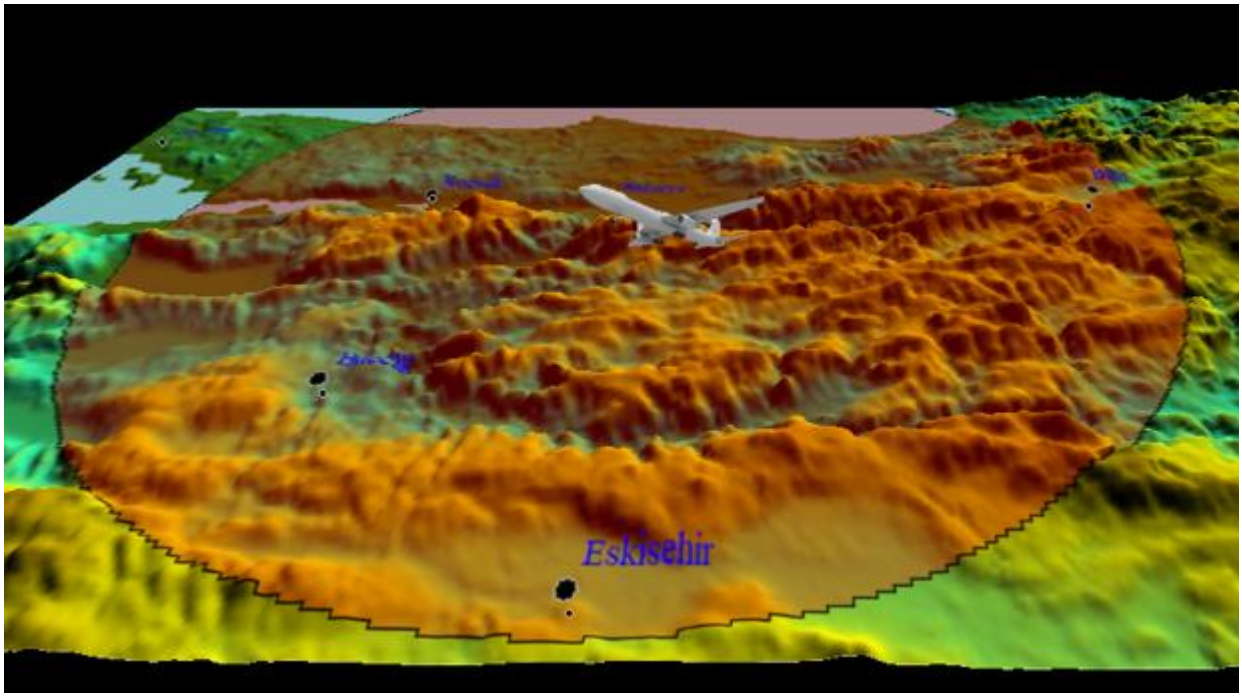
This is an individual work. No team work is allowed.

Similarity check will be applied to submitted codes.

NEAREST CITIES WITHIN A RANGE?

In this homework you will be dealing with geographic position data of cities all around the world.

Often, pilots in an aircraft need to search for the nearest cities within some range of distance when flying their course. This information may be needed for mission / flight planning purposes or sometimes for emergency procedures. In this homework, you are required to develop a program which calculates the nearest cities of the aircraft, given the current aircraft position, the search range using an input file which includes the world cities database. The calculated cities will be sorted from the nearest city to the most distant city within the calculation range and will be written to an output file including data about the cities in the given format.



For example, in the above figure, an aircraft flying from Ankara to İstanbul is depicted at a position near Sakarya and 100 km is provided by the pilot as the search range of the cities. As can be observed, the nearest cities are Sakarya, Kocaeli, Bilecik, Eskisehir and Bolu from nearest to farthest.

REQUIREMENTS

1. **Your program shall be executed from the command line as below using 5 command line arguments:**

>program.exe <*AircraftLongitude(Deg)*> <*AircraftLatitude(Deg)*> <*SearchRange(km)*> WorldCities.txt <*OutputFileName*>

e.g.:

```
>hw1sol.exe 30.51 40.62 100  WorldCities.txt Out2.txt
```

where, each command line argument is described as below:

<AircraftLongitude (Deg)> : Longitude of the Aircraft's current position in decimal degrees. (Refer to Section – Background Information for description of Longitude / Latitude)

<AircraftLatitude (Deg)> : Latitude of the Aircraft's current position in decimal degrees. (Refer to Section – Background Information for description of Longitude / Latitude)

<SearchRange (km)> : The search range provided by the pilot in kilometers.

"WorldCities.txt" : Fourth argument should be the world cities database filename which will be provided along with the homework description.

<OutputFileName> : Fifth argument should be the output file name which will be created and the outputs of your solution will be written to.

2. **The provided input World Cities Database file ("WorldCities.txt") that should be parsed for searching the nearest cities has the following format:**

```
NO LON-DEG LON-MIN LON-SEC LON-EW LAT-DEG LAT-MIN LAT-SEC LAT-NS CITY_NAME CNTRY_NAME POP
1 99° 31' 44.4087" E 16° 28' 22.7891" N Kamphaeng-Phet Thailand 58787
2 12° 11' 23.9867" E 5° 33' 25.1854" S Cabinda Angola 66020
3 100° 20' 56.4051" E 16° 26' 20.4143" N Phichit Thailand 35760
4 15° 46' 59.8813" E 17° 02' 59.9853" S Onjiva Angola 10169
5 24° 49' 48.0051" E 28° 39' 35.9856" S Kimberley South-Africa 142089
6 105° 26' 16.7907" E 10° 22' 55.2145" N Long-Xuyen Vietnam 158153
7 92° 50' 00.0071" E 56° 00' 59.9976" N Krasnoyarsk Russia 927200
8 100° 08' 02.4015" E 15° 42' 03.6142" N Nakhon-Sawan Thailand -999
9 28° 03' 21.5908" E 28° 52' 40.8108" S Hlotse Lesotho 47675
10 13° 24' 21.6155" E 12° 34' 37.1854" S Benguela Angola 151226
```

The first line of the file is the header line which includes the titles of each column of the data file which are:

NO: Unique identifier of the cities in ascending order in range [1,2540], where there are 2540 cities in all over the world in the data file.

LON-DEG, LON-MIN, LON-SEC, LON-EW: These four columns constitute the Longitude of the City in DMS format. (Refer to Section – Background Information for description of Longitude / Latitude)

LAT-DEG, LAT-MIN, LAT-SEC, LAT-NS: These four columns constitute the Latitude of the City in DMS format. (Refer to Section – Background Information for description of Longitude / Latitude)

CITY_NAME : Name of the City

CNTRY_NAME : Name of the Country that the City belongs to.

POP : Population of the city.

Each column of data in one line should be separated by blank character(s).

3. **Your program's output file format should be as depicted below:**

```
1763 30° 24'  0.0083" E 40° 45' 59.9798" N Sakarya Turkey 286787 18.76
1759 29° 55' 22.8082" E 40° 46' 26.3854" N Kocaeli Turkey 196571 52.37
1769 29° 58' 58.7871" E 40° 8' 59.9962" N Bilecik Turkey 40285 68.73
1772 30° 31'  8.4011" E 39° 47'  2.3859" N Eskisehir Turkey 514869 92.96
1922 31° 36' 43.1869" E 40° 44' 20.4141" N Bolu Turkey 96629 93.86
```

where each line should include information of the computed nearest cities. Each column data are separated by a blank character. The columns of each line shall correspond to below information:

NO: identifier of the city.

LON-DEG, LON-MIN, LON-SEC, LON-EW information of the city as four consecutive columns constituting longitude of the city in DMS format. (Refer to Section – Background Information for description of Longitude / Latitude)

LAT-DEG, LAT-MIN, LAT-SEC, LAT-NS information of the city as four consecutive columns constituting latitude of the city in DMS format. (Refer to Section – Background Information for description of Longitude / Latitude)

(Note that output Seconds value in DMS format should be composed of 7 characters, with 4 digits after the decimal point)

CITY_NAME: Name of the City

CNTRY_NAME : Name of the Country that the City belongs to.

POP : Population of the city.

DISTANCE: The computed Great Circle Distance of the City from the current aircraft position. (Refer to Section – Background Information for description of Great Circle Distance computation) The computed distance should be formatted in 2 decimal digits (i.e. two digits after the decimal point).

The consecutive lines should include information of the cities sorted from the nearest city to farthest city to the current aircraft position (as given by the command line argument).

4. **Along with this HW1 description, you will be given 5 test case output files (*Out1.txt, Out2.txt, Out3.txt, Out4.txt, Out5.txt*) generated for your own debugging purposes. Please make sure that your output files are "exactly the same" as these output files corresponding to below sample execution of our ground truth software so that you will be sure that your solution and the output format is correct.**

```
.\hw1sol.exe 32.0 39.0 130  WorldCities.txt Out1.txt
.\hw1sol.exe 30.51 40.62 100  WorldCities.txt Out2.txt
.\hw1sol.exe -59.1 -32.3 300  WorldCities.txt Out3.txt
.\hw1sol.exe -121.88 38.35 1000  WorldCities.txt Out4.txt
.\hw1sol.exe 172.001 -44.0055 250 WorldCities.txt Out5.txt
```

! For grading, your executables will be run by other test cases from almost all continents of the world and correct output will be expected from your software.

5. **You can use any available sorting algorithm in your source code in the literature, but your source code is expected to be comprised by only 1 source file (so all your functions including the sorting function should be in the same file).**

6. **Use exactly below values for conversion constants to be able to get the same results as the ground truth code used for grading:**

   : DEG_2_KM = (111.194926644558550) : degrees to kilometers conversion constant
   : DEG_2_RAD = (0.017453292519943) : degrees to radians conversion constant
   : RAD_2_DEG = (57.295779513082323) : radians to degrees conversion constant.

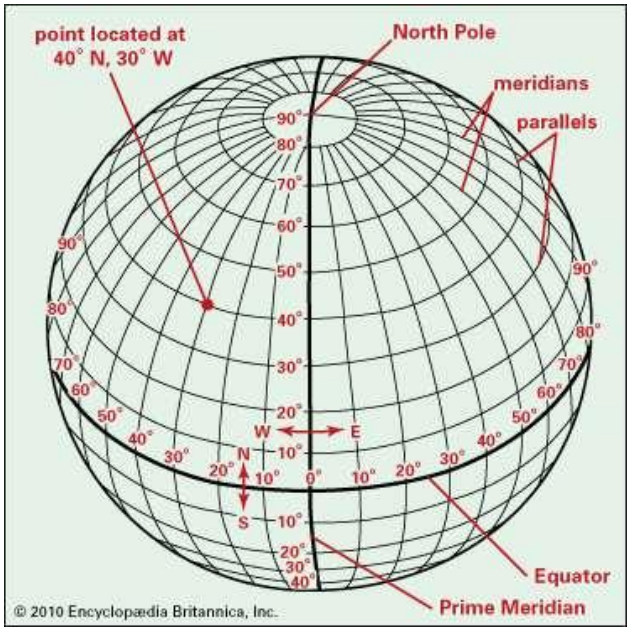
   HINTS

   - ***In this homework, you are not expected to use different aspects of C programming language than we have covered in lectures. For example,***
     - ***proper usage of fprintf, fscanf functions should be enough to parse input files and write output files.***
     - ***Implementation of arrays for storing input database column data are suggested.***
     - ***Usage of strings are mostly limited to storing city textual information as character arrays and usage of fscanf / fprintf %s placeholder for input output operations.***
     - ***Pay attention to units of the input or required output column data, especially the position coordinates and distance data.***
     - ***Pay attention to units of input and output arguments of math.h functions such as sinus ("sin()"), cosine ("cos()") functions. (e.g. refer to*** https://www.gnu.org/software/libc/manual/html_node/Trig-Functions.html#Trig-Functions)

BACKGROUND INFORMATION

LATITUDE / LONGITUDE:

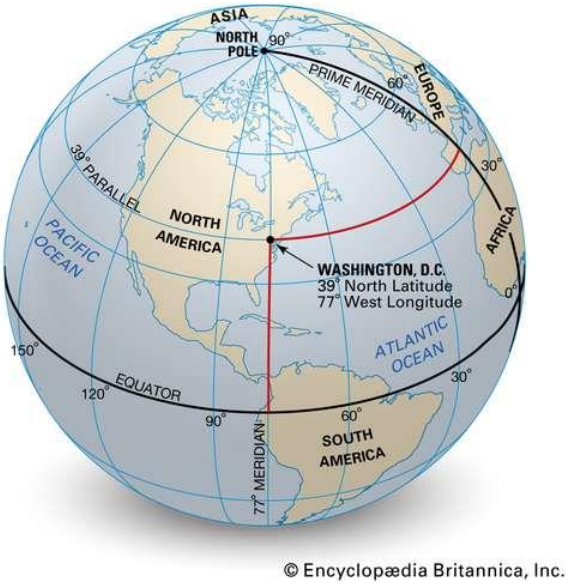
Latitude and Longitude *("Enlem & Boylam" in Turkish*) constitutes a geographic coordinate system by means of which the position or location of any place on Earth's surface can be uniquely determined and described. [1]

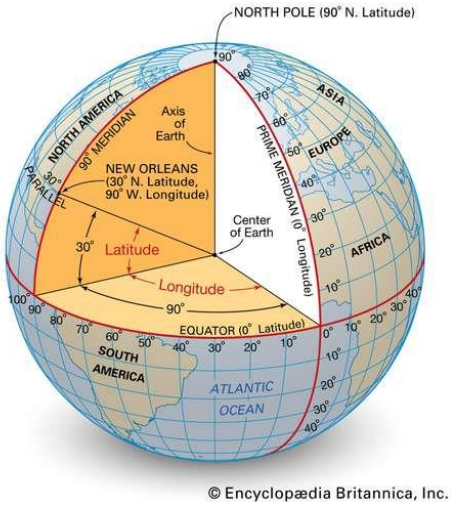


Latitude is a measurement on a globe or map of location north or south of the Equator. [1]

Longitude is a measurement of location east or west of the prime meridian at Greenwich, the specially designated imaginary north-south line that passes through both geographic poles and Greenwich, London. [1]

The combination of meridians of longitude and parallels of latitude establishes a framework or grid by means of which exact positions can be determined in reference to the prime meridian and the Equator: e.g. : a point described as 40° N, 30° W, is located 40° of arc north of the Equator and 30° of arc west of the Greenwich meridian. [1]





[1] https://www.britannica.com/science/latitude

Degrees/Minutes/Seconds (DMS) vs Decimal Degrees (DD)

Latitude / Longitude Coordinate Systems are composed of angular coordinates where;

- latitude lines range between -90 and +90 degrees,
- longitude coordinates are between -180 and +180 degrees.

There are two ways to write / represent geographic coordinates;
1- Decimal Degrees format
2- DMS format

DMS format uses uses degrees-minutes-seconds (DMS) representation of an angle. Minutes and seconds range from 0 to 60. For example, the geographic coordinate expressed in degrees-minutes-seconds for New York City is:

LATITUDE: 40 degrees, 43 minutes, 50.1960 seconds N or 40° 43' 50.1960'' N
LONGITUDE: 73 degrees, 56 minutes, 6.8712 seconds W or 73° 56' 6.8712'' W

where

- ° symbol is used for degrees
- ' character is used for minutes
- '' character is used for seconds
- any of the letters N or S is used for latitudes as the last character to show the hemisphere of the world that point belongs to (N = north hemisphere, S = South hemisphere)
- any of the letters E or W is used for longitudes as the last character to show the which side of the prime meridian that the point belongs to (E = east of the prime meridian, W = west of the prime meridian)

Decimal Degrees directly make use of angular coordinates in degrees (as real numbers with enough precision) [2]

A **DMS** value is converted to **Decimal Degrees** using the formula:[2]

$Ddec = Sign(D + \frac{M}{60} + \frac{S}{3600})$ where,

Sign = -1, If point's hemisphere is "S" for latitudes or "W" – west of the prime meridian for longitudes
= 1 , otherwise (i.e. N or E)

- e.g. coordinates of New York City are [40.730610, -73.935242] in lat/long decimal degrees form. [3]

To calculate the **D, M** and **S** components from **Decimal Degrees,** the following formulas can be used:

$$\mathrm{D} = \mathrm{trunc}(\mathrm{D_{dec}})$$
$$\mathrm{M} = \mathrm{trunc}(60 \times |\mathrm{D_{dec}} - \mathrm{D}|)$$
$$\mathrm{S} = 3600 \times |\mathrm{D_{dec}} - \mathrm{D}| - 60 \times \mathrm{M}$$

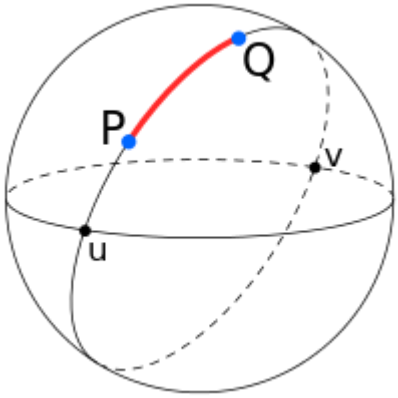
where **|Ddec - D|** is the absolute value of Ddec - D and **trunc** is the truncation function (*i.e **floor** in C*).
Note that with this formula, only D can be negative and only S may have a fractional value.

Sources:
[2] https://en.wikipedia.org/wiki/Decimal_degrees

[3] https://www.latlong.net/place/new-york-city-ny-usa-1848.html

GREAT CIRCLE DISTANCE :

The shortest path between two points on a sphere, also known as an orthodrome, is a segment of a great circle.



(from [7])

To find the great circle (geodesic) distance on the earth sphere between two points located at latitude δ and longitude λ of (δ_1, λ_1) and (δ_2, λ_2) on a sphere of a radius a is: [4]

$$d = a \cos^{-1} [\cos \delta_1 \cos \delta_2 \cos (\lambda_1 - \lambda_2) + \sin \delta_1 \sin \delta_2].$$

Hints:
- use **a = (DEG_2_KM * RAD_2_DEG)** using the macro values provided within the homework description.
- use math standard library (math.h) for trigonometry functions, buy pay attention to the units of the parameters of the function implementations. [5] [6]

[4] https://mathworld.wolfram.com/GreatCircle.html
[5] https://www.gnu.org/software/libc/manual/html_mono/libc.html#Trig-Functions
[6] https://www.gnu.org/software/libc/manual/html_mono/libc.html#Inverse-Trig-Functions
[7] https://en.wikipedia.org/wiki/Great-circle_distance