# ECS40 Winter 2017 2017-03-07

## *Homework 4 due Sun 2017-03-19 at 20:00*

**Use the handin directory hw4 to submit your work**

## *Emergency landing application*

### Description

In this assignment (HW4), you will implement a program called **`mayday`** that prints a list of airports and runways that are closest to a given location. The program will list airports that have runways longer than a given minimum length. It will also print the distance from the given location to the airport. In a fictional scenario, a pilot who encounters an emergency situation would use your program to get a list of the nearest landing options. The pilot would enter his/her current location as latitude and longitude (in degrees decimal) as well as the minimum runway length needed to land (in ft).

This assignment will test your C++ proficiency in opening and reading text files, using strings, using STL containers, using STL algorithms, and implementing function objects.

### Program specification

The program will read data from two text files containing information about Federal Aviation Administration (FAA) facilities and runways. The files **`Facilities.txt`** and **`Runways.txt`** are provided and contain a list of 19700 facilities (such as airports, heliports, seaplane bases) and 23595 runways located mostly in the United States, but also in remote locations. Each line in the **`Facilities.txt`** file contains the description of a facility (airport, heliport or seaplane base). The first part of the line contains the site number, a 10-character string that uniquely identifies the facility. The rest of the line contains other information, including the facility type, name, code, and position (latitude and longitude in various formats). For example, San Francisco International Airport has site number **`02187.*A`** , type **`AIRPORT`** , code **`SFO`**, name **`SAN FRANCISCO INTL`** , latitude **`135427.7000N`** and longitude **`440551.5000W`** (expressed in seconds decimal). The positions of these fields in the line are:

- Site number: characters 1-10
- Type: characters 12-24
- Code: characters 25-28
- Name: characters 131-180
- Latitude (sec decimal): characters 536-547
- Longitude (sec decimal): characters 563-574

Other fields are not relevant to this assignment.

Each line in the **`Runways.txt`** file contains the description of a runway. The first part of the line contains the site number of the facility it belongs to (i.e. the 10-character string described above). The rest of the line contains other information about the runway, including its name and length (in feet). For example, runway 10L/28R of San Francisco International airport has site number **`02187.*A`** , name **`10L/28R`** and a length of **`11870`** ft. The positions of these fields in the line are:

- Site number: characters 1-10
- Name: characters 14-20
- Length: characters 21-25

Other fields are not relevant to this assignment.

The **mayday** program will read the current position (latitude and longitude in degrees decimal) and minimum runway length from the command line arguments.
The program should then open and read the text files, create appropriate **Facility** and **Runway** objects, and store them using STL vectors. The facilities should then be sorted in order of proximity to the current location using the STL **sort** algorithm and a function object **Closer** defined in the file **Closer.h** . The program should then examine each facility in order of proximity and search for runways that belong to this facility (as identified by the site number) and have a length larger than or equal to the minimum runway length required. This search will use the STL **find_if** algorithm and a function object **SiteNumber** defined in the file **SiteNumber.h** . For each facility (up to a maximum of 10 facilities) the program will print the type, code, name, and distance from the current position, followed by a list of runways having a sufficient length.

You are given the files **Facility.h** and **Runway.h** which define classes that represent facilities and runways respectively. You should not modify these files. You must create the files **Facility.cpp** and **Runway.cpp** to implement the member functions of the classes **Facility** and **Runway** . You are also given a file **mayday.cpp** which contains an incomplete implementation of the **main** function. You must add lines to the file **mayday.cpp** to complete the implementation in three places identified with the comment

```
// Insert your code here
```

The comments preceding the above line indicate what must be implemented. You should not remove lines from the file **mayday.cpp** . You must create the files **Closer.h** and **SiteNumber.h** to include the definition of the corresponding function objects. You are also given the files **gcdistance.h** and **gcdistance.cpp** that include the implementation of the calculation of the distance between two locations on Earth specified by their latitudes and longitudes. These two files should not be modified.

### Facility class

The member functions of the **Facility** class are defined as follows:

**Facility(string s)**
The constructor takes a single **string** argument. The argument **s** contains a full line read from the **Facilities.txt** file. The constructor should initialize the data members of **Facility** by selecting the appropriate substrings from the argument. The latitude and longitude fields should be converted to double values using the **convert_latitude** and **convert_longitude** member functions. The sign of the **latitude_** and **longitude_** data members should be determined by checking whether the latitude and longitude fields end with **N** or **S**, and **E** or **W** respectively.

```
string site_number(void) const
```

This function returns the facility's site number.

**`string type(void) const`**
This function returns the facility's type.

**`string code(void) const`**
This function returns the facility's code.

**`string name(void) const`**
This function returns the facility's name.

**`double latitude(void) const`**
This function returns the latitude of the facility in degrees decimal. Latitudes in the southern hemisphere are negative numbers.

**`double longitude(void) const`**
This function returns the longitude of the facility in degrees decimal. Longitudes in the western hemisphere are negative numbers.

**`double distance(double lat, double lon) const`**
This function returns the distance in nautical miles between the facility and the position defined by (lat,lon) in degrees decimal. The implementation of this function uses the **`gcdistance`** function provided in files **`gcdistance.h`** and **`gcdistance.cpp`** .

**`double convert_latitude(string s) const`**
This function converts the string **`s`** representing a latitude in seconds decimal to a **`double`** value in degrees decimal. One degree is 3600 seconds. The sign of the result is positive if the string **`s`** ends with **`N`** and negative if it ends with **`S`** . For example, the latitude represented by the string **`135427.7000N`** should be converted to the value **`37.6188`**

**`double convert_longitude(string s) const`**
This function converts the string **`s`** representing a longitude in seconds decimal to a **`double`** value in degrees decimal. One degree is 3600 seconds. The sign of the result is positive if the string **`s`** ends with **`E`** and negative if it ends with **`W`** . For example, the longitude represented by the string **`440551.5000W`** should be converted to the value **`-122.3754`** .

### Runway class

The member functions of the **`Runway`** class are defined as follows:

**`Runway(string s)`**
The constructor takes a single **`string`** argument. The argument **`s`** contains a full line read from the **`Runway.txt`** file. The constructor should initialize the data members of **`Runway`** by selecting the appropriate substrings from the argument.

**`string site_number(void) const`**
This function returns the site number of the facility that the runway belongs to.

**`string name(void) const`**
This function returns the name of the runway.

**`int length(void) const`**
This function returns the length of the runway in ft.

**`int convert_length(string s) const`**
This function converts the string **`s`** representing a runway length to an **`int`** value in feet.

### Function objects
Two function objects must be defined. The **`Closer`** function object (to be defined in the file **`Closer.h`**) is used to sort facilities in order of proximity to the current location. The **`SiteNumber`** function object (to be defined in the file **`SiteNumber.h`**) is used to find a runway having a given site number.

### Test programs
The test programs **`testFacility.cpp`** and **`testRunway.cpp`** are provided and should not be modified. These programs will be used (with the corresponding input and output files) to test that your implementation of the **`Facility`** and **`Runway`** classes is correct.

### HW4 Assignment
Your task is to implement the files **`Facility.cpp`** , **`Runway.cpp`**. , **`Closer.h`** , **`SiteNumber.h`** , and complete the implementation in **`mayday.cpp`** . The files **`Facility.h`** and **`Runway.h`** are provided and should not be modified. The **`Makefile`** is provided and should not be modified. The **`mayday`** executable and other executables should be built using the command

```
$ make
```

Do not use C++11 or C++14 features in your source files.

### Test cases
Five test cases of the **`mayday`** program are provided with corresponding output files. Shell scripts files named **`testmayday1.sh`** to **`testmayday5.sh`** are provided and include the invocation of the **`mayday`** program with its command line arguments. The files **`testmayday1.out`** to **`testmayday5.out`** contain the corresponding output.

### 1) testmayday1.sh: Sacramento International Airport
You are flying over Sacramento International airport. Your position is (38.6954,  -121.5910) and you request a minimum landing length of 6000 ft.

### 2) testmayday2.sh: Helicopter on downtown Sacramento
You are flying a helicopter over downtown Sacramento. Your position is (38.55, -121.49) and you request a minimum landing length of 40 ft.

### 3) testmayday3.sh: Flight 1549
You are flying an Airbus A320 and just took off from New York La Guardia airport. You lost power after hitting a flock of birds. Your position is (40.8615, -73.8775) and you request a minimum landing length of 6000 ft.

### 4) testmayday4.sh: South Pacific Ocean

You are flying over the South Pacific Ocean. Your position is (-15, -134) and you request a minimum landing length of 6000 ft.

### 5) testmayday5.sh: Space Shuttle approach to CA

You are flying a Space Shuttle, approaching the California coast. Your position is (35, -123) and you request a minimum landing length of 15000 ft.

Two test cases for the programs **`testFacility`** and **`testRunway`** are also provided.

Use the **`mayday`** executable and the **`testFacility`** and **`testRunway`** executables together with the corresponding input and output files to check your implementation. Verify that your program reproduces the test output *exactly*. Use the **`diff`** command to compare your output files with the reference test output files. Note that other test files may also be used when grading your implementation.

### Submission

Create a tar file named **`hw4.tar`** containing all the files needed to build the **`mayday`** , **`testFacility`** and **`testRunway`** programs. In order to limit file size, do NOT include the files **`Facilities.txt`** and **`Runways.txt`** . Submit your project using:

```
$ handin cs40 hw4 hw4.tar
```