COMP 1631 Assignment 2 ("Airplane Geometry")

Given: Wednesday, September 23, 2020
Electronic Copy Due: Friday, October 2, 2020 before 11:59 pm sharp

Objectives

- To build problem solving and algorithm development skills.
- To learn basic C++ syntax for variable declarations, statements and arithmetic expressions.
- To gain experience with simple I/O.
- To learn how to invoke functions found in the standard library.
- To practice editing, compiling and testing a complete program.

The Problem

Air Cha–Cha is the latest fledgling airline company trying to grab a share of the megabucks Canadians spend annually on air travel. As part of their pricing strategy, they want to use the ground distance between two airports to calculate two things: 1) the in-air flying distance and 2) travel time.

When a plane takes off, it ascends at a constant rate to its cruising altitude, at which time it levels off and flies in a horizontal path. As the plane nears its destination, it descends at a constant rate to ground level. For simplicity, this assumes that the plane flies at a constant speed and that the flight path an airplane takes when ascending is specified simply as an angle of deviation from horizontal (a.ka. The Ground). Currently, Air Cha–Cha pilots land at the same angle used for take–off.

The in-air distance to fly from Point A to Point B is larger than the surface distance between the two points because of the extra distance associated with ascending and descending.

Air Cha–Cha has hired you to write a series of programs to transform ground distances to in-air distances. The prototype version #1 of the program will compute the distance by air, given the surface distance, the flying altitude and the angle for the ascent (the angle of descent is identical to that of the ascent). It is common to specify the flying altitude in feet, which will be how the user inputs it, but since all output and hence calculations are in kilometers you will need to convert this value appropriately. In addition, given the constant velocity of travel, the program will report the total travel time in decimal hours. Currently, all Air Cha–Cha planes travel at a default constant rate of 400 kilometers per hour.

However, Air Cha–Cha has received a significant number of complaints from passengers who "felt like they were diving" when their plane started its descent toward the destination airport. In the interests of traveler confidence, Air Cha–Cha is considering implementing "smooth descent" – pilots land planes at a smaller angle than that used to climb. So they have asked you to provide an improved version #2 of your program that will perform in the same way as version #1 except for the following:

- the angle of descent may differ from the angle of ascent
- the velocity will be input by the user
- (optional extra – do this **only** if you have completed everything else as specified) the take–off velocity, cruising velocity and landing velocity are to be obtained from the user and may all differ. Typically, the take–off and landing velocities will be slower than the cruising velocity but this need not be the case

Input (Version #1)

The program should prompt the user for (and then read from the keyboard) the ground distance between two airports (in kilometers), the flying altitude (in feet) and the angle for the ascent (in degrees). For example, the following illustrates what should appear on the screen as input for a calculation (the user input is shown in bold):

```
Welcome to the Air Cha-Cha Distance Finder.  Ready to calculate...
Enter ground distance (in kilometers): 1000
Enter flying altitude (in feet): 40000
Enter angle for take-off (in degrees): 45
```

You can assume that all of the input will be valid.

Output

The output should be formatted similarly to the examples below (i.e. nicely spaced). Marks will be awarded for readable output with whitespace and blank lines as shown in the output example. Your output should **not** attempt to restrict the number of decimal places when printing.

For the input given above, the corresponding output is:

```
Trip data:
Ground distance = 1000 km
Altitude = 40000 feet (12.192 km)
Take-off angle = 45 degrees (0.785398 radians)
Default velocity = 400 km/hr

In-air distance = 1010.1 km
Traveling time = 2.52525 hours
```

Input (Version #2)

This program needs extra input values from the user. In addition to the distance between two airports (in kilometers) and the flying altitude (in feet), this version will require two angles (both in degrees) and one velocity (the **optional** extra requires three velocities). For example, the following illustrates what should appear on the screen as input for a calculation in version #2 without the optional extra – the extra will obviously ask for 2 more velocities (the user will type the entries in bold):

```
Welcome to the Air Cha-Cha Distance Finder.  Ready to calculate...
Enter ground distance (in kilometers): 2000.5
Enter flying altitude (in feet): 37000.125
Enter angle for take-off (in degrees): 45.1
Enter angle for landing (in degrees): 30.9
Enter flight velocity (in km/hr): 500.25
```

The corresponding output is:

```
Trip data:
Ground distance = 2000.5 km
Altitude = 37000.1 feet (11.2776 km)
Take-off angle = 45.1 degrees (0.787143 radians)
Landing angle = 30.9 degrees (0.539307 radians)
Flight velocity = 500.25 km/hr

In-air distance = 2008.3 km
Traveling time = 4.01459 hours
```

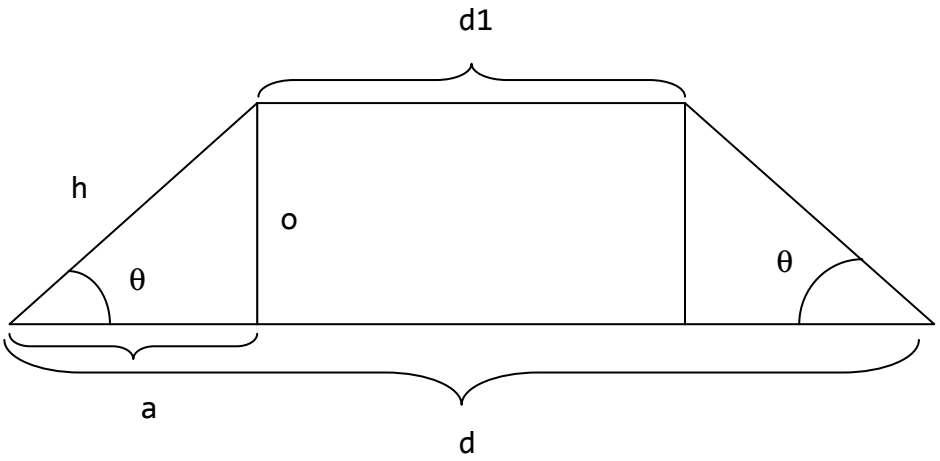
Restrictions

You may not use any program statements that have not been covered in class. In particular, you may not use any "if" statements or loops in this assignment. You may call functions in the `math` library and the `iostream` library. You may not use any programmer–defined functions in this assignment.

Calculations – Getting Started

In a normal problem solving phase for version 1 you would start by creating a labeled diagram for the plane's flight path. Determining how to compute the labeled values would entail researching basic triangle geometry and simple physics. However, due to the short duration of the assignment we will provide a review of this information for you.

The flight plan is shown below:



where:

d is the horizontal distance between cities
o is the altitude
a is the horizontal distance flown during take-off
θ is the take-off angle
d1 is the distance flown at the specified altitude
h is the actual distance flown during take-off

Realize that in version 1 the two triangles are identical, but in version 2 they can be different.

From trigonometry:

```
sin θ = o / h          cos θ = a / h       tan  θ = o / a
```

And from general physics: `distance = rate * time`

Determine which values will be input by the user and which the values need to be computed. Then, using the provided information, determine how to compute the required values.

Notes about the Math Library

A variety of mathematical functions, including some for simple planar geometry, can be found in the math library. To include this library in your program, use the following compiler directive:

```
#include <cmath>
```

Geometry functions which take angles as input use the **angle in radians, not in degrees.** The formula to convert degrees to radians is:

```
radians = degrees * 2π / 360
```

A system–defined constant value for π is available in the math library. It is called **`M_PI`**.

Detailed Specifications:

Design and implement a program which solves the problem as follows:

1. Include the header files for the I/O stream library and the math library so that you may invoke any required system functions.
2. Write out the introductory message.
3. Read in the data required by the program.
4. Perform all required calculations.
5. Write the output.
6. Be sure to follow the standards described in the handout **Coding Standards** (to be discussed in class).
7. Be sure to design your algorithm before you code the program.

Approach To Writing The Program

First break the problem into pieces (divide and conquer). Even though all of the work must be done in the main program, think of it as a collection of major tasks, then deal with the major tasks individually. This is referred to as top–down design. It is important to get into the habit, right from the beginning, of designing an algorithm in pseudocode before starting to write any code. Your first draft might look like:

```
get inputs
calculate in-air distance
print results
```

You now have the overall logical structure for version 1 of your program and can focus on each major step by itself. Break each of these big steps down into smaller and smaller tasks until you are at the level of pseudocode.

After you have developed the algorithm, translate it into a complete C++ program (including all syntax such as ; {}, and the like). Details such as specifying the wording of prompts can be worked out at this time. Do not hesitate to refer to examples in the text and your notes, but aim toward being able to do the translation into C++ entirely from memory. For the first few programs you design, you will want to do this translation on paper before you go to the lab to create the program source file.

If you type in your whole program at once, you may get an overwhelming number of error messages. You can avoid this by entering your code one chunk at a time (input chunk, processing chunk, output chunk) and compiling and debugging each chunk as you go. Naturally, you will have to include enough in each chunk to maintain a valid program (for example, don't add { without the closing }).

Hint: Use extra cout statements to print out your calculated values so you can check that they are correct. Once your program is working correctly, **be sure** to remove these statements.

Note: Start work on the assignment early. Don't wait until the last minute; leave yourself plenty of time to solve unexpected problems with the C++ language.

Official test data will be given out close to the due date. You will want to avoid the last–minute panic which will result if your program bombs on the assigned test data so you will have to come up with a number of sets of test data on your own. By giving your program a thorough workout in advance, you be confident of its correctness.

Only when version #1 is working perfectly should you start working on version #2. Since only minor modifications will be required, you should begin by making a copy of your first program under a different file name. Revamp your algorithm to encompass the necessary changes and make the changes in your pseudocode. Finally, make the required C++ changes, debug and test the modified program.

Hand In

You will need to hand in **two** programs; one for version 1 and one for version 2. Specific submission instructions and test data will be provided a couple of days before the assignment is due.

Should you be unable to complete version 2, be sure that version 1 is complete in all aspects, both in terms of function and documentation. This will be worth roughly 70% of the assignment.