SYDE 121
Lab Number 6

Notes:

1) Remember to review the SD121 Style Guide to know how to properly document your function declarations. Note that the Style Guide refers to "declarations" as "prototypes". Both terms are acceptable, but Savitch uses "declarations".
2) Now that we have started function-based coding, you are allowed to use carefully selected global constant variables e.g., you will probably need an expression for pi in this lab – this can be created as a global constant.
3) Remember to provide feedback to the TA in your header as to the success of your program.

Exercise 1: Quadratic equation solver

Learning Objectives: Practice implementing functions. Practice with call-by-reference function parameters.

Read This First

As you should already know, the solutions to the quadratic equation

$$ax^2 + bx + c = 0$$

are given by

$$\frac{-b+\sqrt{b^2-4ac}}{2a} \quad \text{and} \quad \frac{-b-\sqrt{b^2-4ac}}{2a}$$

What to do

Write a program, *lab0601.cpp*, to compute the solutions of a quadratic equation. The user should be prompted (in a separate function) for the values of the coefficients *a, b,* and *c* and the data should be entered within that same function. Assume that the values *a*, *b*, and *c* are all real. The program should then compute and display the roots. You should be able to calculate and display the roots whether or not they are real or complex. Use separate functions for calculating the roots and for displaying the roots.

Create a "*README_Lab0601.txt"* file, and create detailed instructions on how to run your *lab0601.cpp* program, including what input should be entered.

Bonus: Use structs for representing the root components

Use a `struct` to represent and store the real and imaginary components of each root in a more concise manner. Reminder, make sure you indicate in your header file if you have attempted the bonus part of this question.

What to Hand In

Submit both your *lab0601.cpp* files and *README_Lab0601.txt* files to the Lab Assignment 6 LEARN dropbox.

Exercise 2: Processing Student Grades

Learning Objectives: Practice with functions, formatting output, and using loops.

Read This First

A common function that course instructors perform is to process student grades at various points in the term, for instance, after a midterm or at the end of the term. It is helpful to know both the class average to understand how the students are performing overall and also to know the distribution of the student grades to understand any trends that may be occurring. For instance, in a class such as SYDE 121 where many students have never programmed before, and many have already taken a course in programming, we would expect to see a bimodal distribution on some course deliverables, especially early on before the first group has a chance to catch up to the second. A common way we can understand this grade distribution is to create a histogram from the input grades, as shown below for a set of input grades.

```
  0-49: ****
 50-59: **
 60-69: ****
 70-79: *****
 80-89: ********
90-100: ***
```

Grade histograms display the accumulation of grades across a set of "bin" values. The most common bins used for processing grades are 0-49, 50-59, 60-69, 70-79, 80-89, and 90-100. Each asterisk "*" represents one grade that falls into that bin.

What to do

Write a program, *lab0602.cpp*, which reads a list of final student grades input by the user and outputs the class average and a histogram showing the grade distribution for the class, using the bins mentioned above. The program should include two functions, one to populate the histogram bins, and one to output the histogram. It may be useful to use some call-by-reference formal parameters in the function that updates the histogram bins.

Create a "*README_Lab0602.txt"* file, and create detailed instructions on how to run your *lab0602.cpp* program, including what input should be entered.

What to Hand In

Submit both your *lab0602.cpp* files and *README_Lab0602.txt* files to the Lab Assignment 6 LEARN dropbox.

Exercise 3: Computing Flight Heading Adjustments

Learning Objectives: Practice using and implementing functions, practice implementing mathematical concepts.

Read This First

Pilots often have to make complex computations to adjust their aircraft trajectories due to external factors, such as wind speed and direction in order to fly the most efficient route to their intended destination. The "Example 3" provided on page 571-572 of your SYDE 111/113 textbook (Adams, *Calculus: A Complete Course, 8th ed.*), describes a method for determining such flight trajectory (i.e. heading) adjustments using relative velocities.

What to do

Write a program, *lab0603.cpp*, to compute the necessary aircraft heading adjustments discussed in the "Example 3" problem, given the following information:

- the aircraft's cruising speed with no wind (i.e. in still air) (all speeds should be given in km/h),
- the wind speed and direction (assume positive numbers represent wind blowing from the west, and negative numbers represent wind blowing from the east – assume no other wind direction is possible for the purposes of this assignment), and
- distance and relative compass direction from originating to destination cities (all distances should be given in km).

The program should obtain the above information from the user, and compute the heading adjustment, and estimated flight time. The program should then output the computed heading adjustment in angular degrees (e.g. 45°), as well as the computed flight time in hours (e.g. 1.5 hours).

The program should include at least five (5) functions, one for obtaining the user input, one for converting the original flight compass direction to a heading, one to compute the necessary heading adjustment, one to compute the estimated flight time, and one to output the results (i.e. heading adjustment and flight time).

Sample output for the program is provided on the next page.

Sample output (bold indicates user input):

```
This program will compute the necessary heading adjustment for
your flight, and provide the estimated flight time.

Enter the aircraft cruising speed in still air (in km/h): 300
[Enter]
  cruising speed = 300

Enter the wind speed in km/h: 100 [Enter]
  wind speed = 100

Enter 1 if the wind is blowing from the West and -1 if wind is
blowing from the East: -1 [Enter]
  The wind is blowing from the East.

Enter the distance between the originating and destination
cities, in km: 400 [Enter]
  flight distance = 400

Enter the compass direction of the destination city, relative
to the originating cities, using the following values:
1: ENE
2: NE
3: NNE
4: N
5: NNW
6: NW
7: WNW
8: W
9: WSW
10: SW
11: SWS
12: S
13: SSE
14: SE
15: ESE
16: E
3 [Enter]
 flight direction = 3: NNE
 original flight heading = 67.5

Your adjusted flight heading is: 49.5637 degrees.
Your estimated flight time is: 1.61844 hours.
```

Hints: The `cmath` library contains all of the trigonometry functions you will need to implement this question. The `asin` function computes the arcsin. All trigonometry functions take in radians as arguments and return radian angles. It may be useful to store heading angles in radians, to simplify computations, and then convert them during output. The `cmath` library contains a global constant named, `M_PI`, that may be helpful.

Create a "*README_Lab0603.txt*" file, and create detailed instructions on how to run your *lab0603.cpp* program, including what input should be entered.

Due Date

All materials for this lab are due **Friday, October 31** by **1:30pm.**