Amy Yu
UID: 905138432
CEE/MAE M20
October 8, 2019

Homework 1

1. I love M20

1.1 Introduction

The goal in this problem is to connect two strings entered by the user. However, the second string should be flipped, such that the last letter becomes the first letter and so on. The result should be a complete sentence that is printed on the screen.

1.2 Model and Methods

The script first prompts the user to enter two strings using the `input` function. In order to print the strings, the `fprintf` function was used to print the first string. Then, it was used again to print a flipped version of the second string. The formatSpec `' %s'` was also used to ensure that a space was printed before the first and second strings. This makes the output a sentence. The second string was flipped using the `fliplr` function. An example of how this is done is:

```
fprintf(' %s', fliplr(secondString))
```

1.3 Calculations and Results

When the program is executed with "I love" as the first string and "02M" as the second string, the following output is printed to the screen.

```
Enter first string: I love
Enter second string: 02M
I love M20
```

The program successfully flips the second string and combines the two strings into one sentence.

1.4 Discussion

This problem demonstrated the use of the `input` and `fprintf` functions.

2. Ellipse Calculations

2.1 Introduction

The goal in this problem is to determine the perimeter of an ellipse. Since there is no simple formula, we will compare eight different methods for approximating the perimeter of an ellipse. The user will input the values of a and b, which are the lengths of the semiaxes. The first four results will be printed in the first row and the last four results will be printed in another row, with a `tab` between each value.

2.2 Models and Methods

The script first prompts the user to enter the values of a and b, which are the lengths of the semiaxes using the `input` function. Then, $h = \left(\frac{a-b}{a+b}\right)^2$ is stored as a variable in order to make it easier to access later. The eight different methods of approximating the perimeter given in the problem are also stored in variables. Two matrices are then created, with the first containing the first four methods and the second containing the last four methods. The first matrix is printed as follows:

```
fprintf('%4.4f\t', A)
```

The `%4.4f` ensures that a minimum of 4 digits are printed and that four digits after the decimal point are printed. The `\t` puts a tab between each value. The second matrix is printed in a similar way. Four decimal places were chosen, because for most inputs, it is enough to see a difference between most of the eight different methods.

In between printing the two matrices, the following line is written:

```
fprintf('\n')
```

This ensures that there is a new line between the two matrices.

2.3 Calculations and Results

When the program is run with the value of a as 2 and the value of b as 4, the following is printed;

```
Enter value of a:
2
Enter value of b:
4

18.8496     19.8692     19.3661     19.3768
19.3769     19.3769     19.3769     19.3886
```

Most of the methods have similar outputs, and the last digit in method four is the only one that is different from methods five through seven, which is why four digits after the decimal point was chosen. For methods five through seven, all of the decimal points would be the same if more were displayed, so no more than four is shown.

For larger values of a and b, such as 1234 and 5678, respectively, there is a larger difference between the methods.

```
Enter value of a:
1234
Enter value of b:
5678

21714.6884  25815.5738  23853.4227  24016.7242
24023.9275  24023.4063  24023.8793  24256.2114
```

However, many of the results are still relatively close, especially methods five through seven.

2.4 Discussion

The eight results are all approximations of the same thing: the perimeter of an ellipse. However, there is a large difference between the methods, especially between methods one through three. This is because there is no simple formula for the perimeter of an ellipse.

One way of improving the code can be to have the program throw an error and display an error message using the `error` function when the values entered cause a fraction to be undefined. `if` statements can be used to throw an error when the value of both a and b are zero, or when they add up to zero. Currently, `NaN` is displayed when this happens.

3. Trigonometric Calculation

3.1 Introduction

The goal in this problem is to use $\cos(60°)$, $\cos(16°)$, and $\sin^2\theta + \cos^2\theta = 1$ to determine the values of $\cos(11°)$ and $\sin(11°)$. Trigonometric identities are used to achieve this.

3.2 Models and Methods

The script first stores the given values in variables to make them easier to access later and prevent accidental changes to the givens. Then, $\sin^2\theta + \cos^2\theta = 1$ was used to find the values of $\sin(60°)$ and $\sin(16°)$, which were also stored in variables. The negative angle and sum identities were then used to find $\cos(44°)$ and stored in a variable. The half angle formula was then used twice to find $\cos(11°)$, and the value of $\cos(22°)$ was used to find $\sin(11°)$. The `fprintf` function was used to print the values as follows:

```
fprintf('%s', "cos(11) = ")
fprintf('%6.6f', ans1)
```

The first line prints a string to make the output neater and clearer, and the second line prints the answer with a minimum of six digits and six digits after the decimal point. Six digits was

chosen because it allows the answer to be compared to that of a calculator. A new line is printed, and then the sine value was printed in the same way.

3.3 Calculations and Results

When the program is run, the following is outputted:

```
cos(11) = 0.981627
sin(11) = 0.190809
```

These values are the same as that outputted by a calculator, which shows that the use of trigonometric identities was accurate.

3.4 Discussion

Although the trigonometric identities should be accurate, there are rounding errors since the given value of $\cos(16°)$ was rounded. The computer also cannot store infinite values. This is why six decimal places were chosen to be displayed, since those digits are not affected by the rounding. It is very important to store all the intermediate values in variables, as this prevents the given values from accidentally being changed or mis entered.