1 Introduction

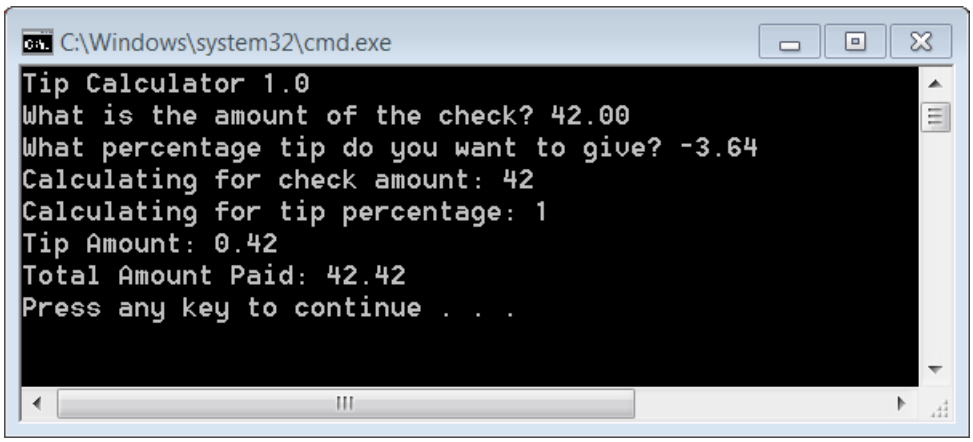
With knowledge of variables and conditionals we can write a simple program to automate a tedious calculation. In this assignment, you will program a tip calculator. The rest of the handout describes what the program should do and gives some hints on how to go about writing it.

- This is an **individual assignment**. That means that while you are encouraged to help each other and talk about the assignment, the code that you submit must be your own. You also must acknowledge who you worked with, and what their contributions to your solution were. You may not submit code that you do not understand.
- Submit your `.cpp` code file on canvas. This assignment is due on **Wednesday, September 14 at 11:59pm**. Late submissions will not be accepted.
- Your grade will based on both (1) whether your code meets the specifications below, and (2) whether it is properly documented. For this assignment, the standard for documentation is: "Have I sufficiently cited all of my sources, and are my comments clear enough that someone else who has a similar amount of programming experience as I do, but hasn't seen this assignment before, could easily understand what is going on?"

2 Specifications

1. Your code file will be named `yourname_HW1.cpp`. For example, our files would be called `shaunramsey_HW1.cpp` and `kylewilson_HW1.cpp`.
2. The program will begin by outputting the message `Tip Calculator 1.0` to the screen.
3. The program will ask the user for a check amount.
   (a) If the user enters a check amount that is < 0, the program should print a message like `Warning: this program is unsure how to tip a negative check amount.` Then the program should immediately quit.
4. The program will ask the user for a tip percentage.
   (a) If the user enters a tip percentage that is < 0, the program will immediately change it to 1%.
5. The program will print a summary of what amount and percentage it is using.
6. Then the program will print out the tip amount.
7. Then the program will print out the total amount paid.

Here is a sample of what a correct program will look like:



3 Tips for Doing the Assignment

Incremental development. A good way to write a program is to build it *incrementally*. Rather than writing a program that satisfies all of the specifications above, start with just one. Then add to the program one piece at a time until every requirement is met. The most important piece of this is testing: after each change, try running your program again, and make sure that each new piece works.

Sample Incremental Steps The steps below are an example of one way to complete this assignment incrementally. Feel free to follow these steps, or to take your own approach.

1. Start with spec 1. Make a program that has the right name and runs. It won't do anything yet. Start with the short hello program from the first lab.
2. Now add spec 2. Make it print the welcome message. *(Try to run it!)*
3. We'll save the user input and the input checking for later. For now, create variables called `checkAmount` and `tipPercentage`. Since we haven't written the user input code yet, initialize them to sample values, like `35` and `20`. *(Try to run it!)*
4. Write code to summarize the input values (spec 5.) *(Try to run it!)*
5. Make a variable called `tipAmount` and calculate the amount of the tip. (Remember that `tipPercentage` is a percentage, so you will need a division by `100.0`). *(Try to run it!)*
6. Print out the tip amount and the total amount paid (specs 6 and 7). *(Try to run it!)*
7. Go back and use `cin` and `cout` to prompt the user for the values of `checkAmount` and `tipPercentage`. *(Try to run it!)*
8. Finally, use `if` statements to check the user inputs. Immediately after you ask the user for the check amount, check if that amount is valid. Similarly do your check for the tip percentage right after asking the user for the number.

Incremental Testing. Developing program sincrementally works because it lets us test our program at every step, so we find out about our mistakes right away.

At each step, think about what sort of testing you need to do to know that your program is working correctly. For instance, to test the code that checks for negative tip percentages, you would want to try typing in a negative tip, and probably a 0% tip as well, to make sure that the program does the right thing.

Testing constantly may seem a little tedious, but it saves far more time and frustration than quickly hacking out line after line of broken code.