**Summary Report**

In summary, I did it! 2,147 lines of code later, I believe my budget project is done!! But in seriousness, this was the largest coding project I have ever done, and I have learned a lot more about C++ than I expected.

To begin with, my thoughts during planning this project were to break it up into many smaller pieces, and start from there, thus I first started coding the enums such as Currency and Spending\_Cat. My first difficulty came from not being able to find a C++ standard library that could represent a date, and after stack overflow confirmed that C++ does not have one like Java does, I chose to make my own Date class, complete with a Month enum as well. My next challenge came from implementing the cin operator >> for all the enums that could also gracefully do error handling for non-number input, but fortunately stack overflow suggested the great idea of a static\_cast. I personally did not consider this project terribly hard at any time – to me it was mainly just tedious and partially because of my perfectionism that I spent a lot more time than intended on it trying to make input as smooth as possible and output as pretty as possible. Implementing the cin >> operator for each class forced me to think of as many irrational inputs as possible in order to write code around it and had much more of a coding challenge, whereas implementing the cout << operator was a crash course into setw in order to make the output look half-decent. In error handling, I learned the power of stringstream, and chose to use it for all numerical variables with a very large or infinite range such as monetary amounts and conditional percentages, and I used strings more-or-less for everything else so that I could force user input to completely match the intended input and avoid any runtime errors if it did not. For input with small ranges such as Currency (integer 1 to 4 inclusive), that was great! But for Month (integer 1 to 31 inclusive)…that was a lot of code. I learned that for error handling, stringstream is good when you are more trying to avoid crashes and runtime errors, and strings are good when you want to raise the standard of accepted input (such as inputting “1.1” for Date will not work when Date only accepts “1”).

The Event class itself posed a few interesting problems to me. This was the first of two “mega” classes I implemented, the other being Budget, where that class had a large amount of private data members, either of primitive types or of previously implemented classes of mine. Specifically, trying to initialize Date from Event’s cin >> took me awhile to decide how exactly I wanted to do it. Additionally, I realized that an Event name and note may contain multiple words with multiple spaces, so I learned that in order to make >> and getline() coexist, you need to also use cin.ignore().

I know this project would be one of numbers, so it’s no surprise to me that the biggest and most consistent headache for me was the weirdness of floating point math in small decimals. In the Budget class, I tried to do some of the modification methods in O(1) time and either += or -= very tiny percentages that for some reason would be off by .01 and would only be fixed after I called calculate() to run in O(n) time. It was not an arithmetic error – I double and triple checked with my calculator – it was just an unexpected loss of precision in the decimal values that I had to cover up every time with my O(n) calculate method. For all my attempts to make some methods in constant time, floating point math would hold me back. Additionally, in trying to cout print doubles, I also found out that being printed by itself, a double will only show a certain amount of digits and decimal places before it arbitrarily truncates the rest. The math was not wrong, the double value calculation this time was correct, but the problem lied within cout itself. As a result, I had to write the numDigits() method that would count how many digits are in a number, and use that number with setprecision() every time I wanted to cout print a decimal. Even the round method from time to time would have weird floating point math results that would be off by .01, but then it would all be covered up after the calculate() method ran.

Another point of interest to me is that while it does have an integer ceiling and floor function, C++ does not have a round function, forcing me to implement that by myself as well. For every monetary numerical input and percent, I would have to also call the round method to enforce one of my simplifying assumptions of rounding those inputs to the nearest hundredth for calculations.

Within Budget, one difficulty I had was trying to resize an array like an ArrayList in Java, the allow for a budget with a technically infinite number of spending events (or at the least, very large) and its capacity relatively fit the number of spaces taken in memory (instantiating an array with size 1000 is quite a waste when there are only 5 events in the budget). However, I learned the syntax of how to dynamically store an array and was able to implement array resizing as an O(n) operation.

One constant difficulty present when trying to implement cin >> for each class was having to write in a data confirmation check every time. As a programmer, I feel we often unconsciously assume that input is a fast and painless process that is always correct, but not every human inputted data is correct (ah yes, the human factor), thus the need for these periodic data checks was always there.

My favorite feature of this project was being able to implement the for-ex foreign exchange operation. Granted, I took the exchange rates from the internet and after a few round trips of exchanging, my budget suddenly gained 10 cents, but it helped remind me why I decided to do this project in the first place – to help save and stretch money so you can spend it learning, traveling, and enjoying life.

Lastly, this project took a looooonnnggg time! I probably spent about 40 hours coding it and another 10 hours doing the report. I’m surprised Tsinghua has given me a project like this before Georgia Tech has, though if I think about a place like Tsinghua I honestly shouldn’t be. I am very tired right now and C++ in Chinese has been a wild ride, but this couldn’t have been a more fitting end to the craziest class in my craziest semester. To whoever is reading this:刘老师或者李老师，非常感谢！