C3C Ameen Khan

Final Project Design

User Interface:

The user will start the program through a command line interface and provide any information that it needs through command line arguments. However, it will send an email to the user whenever it predicts that there will be a significant price movement.

List of Requirements:

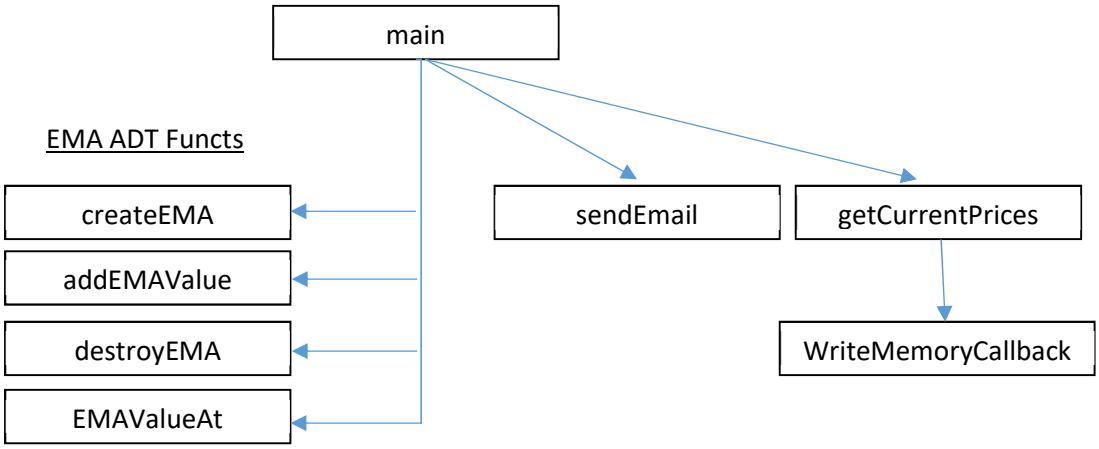
1. (25%) SHALL – Get price data live from an API
2. (15%) SHALL – Notify the user and terminate if API cannot be reached
3. (25%) SHALL – Keep track of two sets of exponential moving averages with different period sizes
4. (15%) SHALL – Notify the user by email when a price movement is predicted
    a. MAY – Automatically buy or sell the currency
5. (10%) SHALL – Do all of the above for multiple currencies specified by the user
6. (10%) SHALL – Get settings for which currencies to track and which period sizes of EMAs to use from a configuration file

Functional Decomposition:

1. main
    a. Read the config file to determine which cryptocurrencies to track and at what periods
        i. Also read the list of users to email
    b. Use createEMA to make two EMA datasets for each cryptocurrency
    c. Call getCurrentPrices to find the prices for each tracked cryptocurrency
    d. Use the addEMAValue to continue to build each EMA
    e. If the program predicts a price movement, sendEmail with the name, price information, and time to the mailingList
    f. If the program encounters an error, use sendEmail to relay the details to the user
    g. Continuously run until stopped by user or error
2. EMA* createEMA(int period, double startValue);
    a. Sets up a pointer to a struct containing the values of the exponential moving averages of a set of data
3. void addEMAValue(EMA* ema, double price);
    a. Adds a value to the the list of values in the struct
    b. Uses realloc to dynamically change the size of the size of the EMA struct
4. void destroyEMA(EMA* ema);
    a. Safely frees all dynamically allocated memory
5. double EMAValueAt(EMA* ema, int index);
    a. Returns the value of the EMA values array at the specified index
6. size_t WriteMemoryCallback(void *contents, size_t size, size_t nmemb, void *userp);

    a. Gets https response information from cURL and writes it to memory
    b. Memory is dynamically allocated
    c. Returns the amount of bytes written
7. Response getCurrentPrices(char* apiKey, char* tokens);
    a. Calls the Normics cryptocurrency price API for the specified currencies
    b. Parses the JSON data and creates a struct containing the price and datetime information
    c. Returns this struct back to the main
8. void sendEmail(FILE* mailingList, char* message);
    a. Open the file and read the emails into a dynamically allocated 2D array of characters
    b. Send each user in the mailingList the specified message

Structure Chart:



Estimates:

1. Number of lines of code for the product: 300
2. Number of lines of code to test the product: 50
3. Number of hours to complete the project: 20