Project Implement Iteration 01

**Goals:**

* Implement structured programming using the C language and continue building a foundation in programming fundamentals
* Understand and implement secure coding practices.
* Implement, and evaluate a computing-based solution to meet a given set of computing requirements (ABET standard 2)
* Communicate effectively in a variety of professional contexts - written. (ABET standard 3)
* Apply computer science theory and software development fundamentals to produce computing-based solutions. (ABET standard 6)

To demonstrate ability to implement a solution using c concepts learned in class so far that is maintainable, testable and flexible.

**Effort**: Individual ([Academic Integrity](https://docs.google.com/document/d/1vfCpxFGn-7Ru5fplmD7lK1n_cSFvhtMtsVQg4SCs2OU/edit?usp=sharing))

* Reminder if you share code/solutions, use someone else's code or use AI you will receive a 0 and be reported to the CS chair and UCCS Dean of Students

**Points:**  100 (Can be turned in up to 24 hours late for 10% deduction) **Read Rubric in canvas**

**Due date:** Look in canvas

**Deliverables:**

* Code solution file named lastname\_ITER01.c file
* Separate PDF or Word document named lastname reflection that contains the answer to the learnings and reflections
* Submit files separately and **not in a zip file.**

[Assignment Overview](#_ovecnf7ghu9r)

[Problem statement:](#_ku7cv63bko8w)

[Requirements](#_fj7ptgxbwzfr)

[Sample Output](#_f5dlu7urpz4f)

[Specification](#_5uj4g9kq40lh)

[Learnings and Reflection (In your own words)](#_gz0lj6ntibsf)

# Assignment Overview

Implement a solution that uses concepts covered in class so far and is based on the requirements and specification. The solution should be

1. Functionally Complete and Correct
2. Maintainable, Testable and Extendable
3. Secure
4. Tested

Reminders

* Highlight when you complete part of the assignments so you can track your progress compared to the deadline.Try to submit a day early.
* Ask clarifying questions if something doesn’t make sense.
* Review Rubric for this canvas assignment and make sure you create quality code and include comments as discussed in [Agile Software Development CS2060](https://docs.google.com/presentation/d/1HMbE4nr7_ShiO1UXbrnhf3NB6DGZc_wmPCFR9kveZM0/edit#slide=id.g110ce7b431a_0_0)
* Test the completed program with different values. Make sure it is functionally complete (satisfies all requirements) and functionally correct(satisfies acceptance criteria).

## Problem statement:

UCCS wants to provide ride sharing for the University that will hire students. You are hired to make a program for the ride share business. For example, here is some information on how [Lyft/Uber calculate ride fares](https://ride.guru/content/newsroom/how-is-my-uber-fare-calculated). The program will provide a calculated fare for all riders and track all the fare details for each driver and a summary of all drivers for the UCCS business.

After the quote the riders can decide to take the ride. The riders will pay the fare and have the option to tip.

The pricing is based on the following formula. If the ride cost doesn’t meet a certain minimum rate a flat rate ride fare will be charged.

Base (or initial) fare – A flat fee charged at the beginning of every ride

Cost per minute – How much you are charged for each minute you are inside the ride

Cost per mile – How much you are charged for each mile of the ride

Minimum flat rate - minimum charge for any ride

Ride Fare = Base Fare + (Cost per minute \* time in ride) + (Cost per mile \* ride distance)

If the ride fare is less than the minimum flat rate, customers will be charged that flat rate.

To simulate the ride miles and time the rider will enter the number of miles needed to get to destination. The program will calculate a random number based on the miles for the number of minutes the ride took.

Random minutes based on number of miles formula:

* Min minutes = 1.2 \* miles
* Max minutes = 1.5 \* miles
* Generate random integer between min and max minutes

Example Values

|  |
| --- |
| **Pricing**  Base (or initial) fare = 1.8  Cost per minute = .25  Cost per mile = 1.2  Minimum flat rate = $20  **Miles to destination for rider**  Miles = 10  **Find random integer between**  Min minutes 12 = 10 \* 1.2  Max minutes 15 = 10 \* 1.5  Estimated Ride Minutes (Integer Randomly generated by computer) = 13  Calculated Ride Fare $17.05 = 1.8 + (.25 \*13) + (1.2 \* 10)  Ride Fare Amount = $20 |

## Requirements

**User Stories and Acceptance Criteria**

Here are user stories for the first sprint iteration for this UCCS Ride Share software program.

**User story 1: Rider Mode**

As a riderI want to be able to get a ride using the UCCS ride share so I can get to my destination.

**Acceptance criteria**

1.1 Get the number of miles, “Welcome to the UCCS Ride Share. We can only provide services for rides from 1 to 100 miles. Enter the number of miles to your destination.” Repeat prompt until a valid number (decimal number) is entered.

* If the number is not a valid number greater then display error “Error: you didn’t enter the number of miles correctly.” Repeat the prompt. Here are the criteria
  + Greater than or equal to minimum number of miles of 1
  + Less than or equal to the maximum number of miles of 100
  + **NOTE:** You can accept 40f as a valid value since scanf will read the 40 and leave the f character in the buffer.
* If the number is valid then
  + Display number of estimated minutes (randomly created)
    - Min minutes = 1.2 \* miles
    - Max minutes = 1.5 \* miles
    - Generate random number between min and max minutes
  + Calculate the charge based on formula
  + Display ride fare charge
  + Display “Thanks for riding with us”
  + Program will repeat for next rider
* If the sentinel value of -1 is entered see user story 2.1 to print summary and end the ride share program.

**User Story 2: UCCS Ride Share Report Mode**

As the UCCS Ride Share business owner I need to be able to shutdown the program and get a report so I have the details of the ride fares.

**Acceptance criteria**

2.1 When the UCCS Ride Share business owner enters -1 when the program prompts in User Story 1.1 for the number of miles

* If there were no rides display “There were no rides”
* If there were rides display the totals

## Sample Output

Note updates to output highlighted in blue

|  |
| --- |
| Welcome to the UCCS Ride Share. We can only provide services for rides from 1 to 100 miles.  Enter the number of miles to your destination: x  Error: You did not enter a number.  Enter the number of miles to your destination: -2  Error: Not within 1 and 100 miles.  Enter the number of miles to your destination: 0  Error: Not within 1 and 100 miles.  Enter the number of miles to your destination: 7  Current Ride Information  Rider Number of Miles Number of Minutes Ride Fare Amount  1 45.0 61 $71.05  Welcome to the UCCS Ride Share. We can only provide services for rides from 1 to 100 miles.  Enter the number of miles to your destination: 7m (will be accepted as 7)  Current Ride Information  Rider Number of Miles Number of Minutes Ride Fare Amount  2 7.0 9 $20.00  Welcome to the UCCS Ride Share. We can only provide services for rides from 1 to 100 miles.  Enter the number of miles to your destination: -1  UCCS Ride Share Business Summary  Rider Number of Miles Number of Minutes Ride Fare Amount  3 52.0 70 $91.05 |

## Specification

Implement a maintainable and extendable solution that is modularized and does not hard code.

Implement the following in your code. Use functions (return, name, parameters) exactly as given.

You can have more functions, variables and constants but you must include what is listed below.

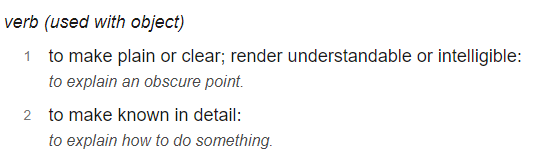
We will change **the value of the variables** in main when we run your code to test it. This is helping you develop algorithms and functions that can be reused for different information without hardcoding.

|  |
| --- |
| **USE THESE PREPROCESSOR SYMBOLIC CONSTANTS**  // used to generate random number of minutes based on number of miles  #define MIN\_RAND\_MINUTES\_FACTOR 1.2  #define MAX\_RAND\_MINUTES\_FACTOR 1.5  // sentinel value to end rider mode  #define SENTINEL\_VALUE -1 |
| **USE THESE VARIABLES IN MAIN**  **YOU WILL NEED MORE VARIABLES BUT THESE SHOULD BE USED WITH THESE NAMES AND WE WILL CHANGE THE VALUES WHEN TESTING YOUR CODE**  **int main(void)**  {  // inItialize variables  //min and max miles range  int minMiles = 1;  int maxMiles = 100;  //example ride calculation values  double baseFare = 1.8;  double costPerMinute = .25;  double costPerMile = 1.2;  double minFlatRate = 20.0; |
| **AFTER CALCULATING THE MIN AND MAX RANDOM MINUTES TO GET YOUR RANDOM NUMBER CAST THEM TO INTEGERS**   * Min random minutes = 1.2 \* miles * Max random minutes = 1.5 \* miles |
| **DEFINE THESE FUNCTIONS IN YOUR SOLUTION**  /\*  \* Returns only valid input from the user based on the min and max values inclusively and sentinel value  \* Parameters:  int min range, int max range, int sentinel value  \* Returns: An double representing a valid number based in the min and max  \*/  double getValidDouble(int min, int max, int sentinel)  /\*  \* Calculates rider fare  \* Parameters:  double base – A flat fee charged at the beginning of every ride  double minuteCost – How much you are charged for each minute you are inside the ride  double mileCost – How much you are charged for each mile of the ride  double minRate- minimum charge for any ride  int minutes - number of minutes of ride  double miles - ride distance  \* Returns: A double representing the fare charge  \*/  **double calculateFare(double base, double minuteCost, double mileCost, double minRate , double miles, int minutes)**  {  }  /\*  \* Prints the riders, miles, minutes and fare  Can be used for one rider or for the summary  \* Parameters: int count, double miles, int minutes, double fare  \* Returns: Nothing  \*/  **void printFare(int count, double miles, int minutes, double fare)**  {  } |

# Learnings and Reflection (In your own words)

**Create a separate Learnings and Reflection Document**

**Explain: In your own words as if you are teaching someone**



* Explanations provide elaboration of the concepts using supporting evidence and vocabulary
* Thoroughly reflects making clear the connection(s) with the experience and assignment.
* Communicate effectively in a variety of professional contexts - written.

|  |
| --- |
| 1. Reflect: What are you proud of from this assignment? What was the most difficult part for you?  - I am proud that I got my code to run properly. The most difficult part of the assignment was calculating the random values correctly and getting my code to accept characters after an integer value was entered. |
| 3. **Include related examples from either your design document or your code for each explanation. You can take screenshots or copy and paste your examples.**  // seed for the random generator with current time  srand((**unsigned** **int**) time(**NULL**));  **double** getValidDouble(**int** min, **int** max, **int** sentinel) {  **double** value;  **int** validInput = 0; // flag to track if input is valid    **while**(!validInput) {  **if** (scanf("%lf", &value) != 1) {  puts("Error: You did not enter a number.\n");  **while**(getchar() != '\n'); // clear buffer    } **else** **if** (value == sentinel) {  validInput = 1; // set flag to indicate valid input    } **else** **if** (value < min || value > max) {  printf("Error: Not within %d and %d miles.\n", min, max);    } **else** {  validInput = 1; // set flag to indicate valid input  }    } // end of while  **return** value;  **3.1 Explain** functional completeness and what you did to ensure that in your solution.  Functional completeness is when I can get my code to build and run successfully with the requirements I needed to follow.  I did quite a few things to ensure that in my solution by handling error input, doing test cases, and displaying prompts for users to understand.  **3.2 Explain** functional correctness and what you did to ensure that in your solution.  Functional correctness is getting the correct output with input.  I did used my debugger, test cases, and error handling to ensure this.  **3.3 Explain** what you did to make your code maintainable and flexible.  I made my code maintainable and flexible by creating functions, putting comments throughout my code and testing my code.  **3.4 Explain** the secure coding standards you implemented in your solution.  The secure coding standards I implemented in my solution is getting valid input, using methods for the random number generator, clearing the buffer after input, and handling errors.  A screenshot of a computer  Description automatically generated |
| 6. Take a screenshot showing your code in your github **remote** repository.  A screenshot of a computer  Description automatically generated |