1. **Program Statement**

Program has different and specific functions for a driver depending on the type of vehicle involved; Car, Truck, Boat, or Plane. Driver will not only have the ability to see and store data of one or more of these types of vehicles, but also be able command the turns vehicles can take and trace the route of each vehicle.

1. **Requirements**
   1. **Assumptions**
      1. User enters lowercase letters for menu selection
      2. User enters integer value for age variable
      3. User selects appropriate option in Menu Selections, without giving out of bound inputs.
      4. Speed is in miles per hour
      5. Land Vehicle transmission is automatic
      6. Truck involved is a big cargo truck
      7. Altitude 0 means plane is at the airport on ground (not in air)
         1. Measured in feet
      8. Lights can only be turned on / off while engine is running
      9. Fuel is measured in gallons
      10. Boat propeller level can only be changed while engine is on
      11. User can only go on at most of two trips per vehicle
      12. User types in 1 for true and 0 for false
   2. **Specifications**
      1. Welcome message to driver
      2. Vector to store type of vehicle driven each time changing it
      3. Vector to store turns that user makes
      4. Vector to store each place of arrival in order
      5. Menu
         1. Menu for Type of Vehicle
            1. Car

1. Turn on Car

2. Accelerate

3. Decelerate

4. Turn off Car

5. Turn right

6. Turn left

* + - * 1. Truck

1. Turn on Truck

2. Accelerate

3. Decelerate

4. Turn off Truck

5. Turn right

6. Turn left

* + - * 1. Boat

1. Turn on Boat

2. Accelerate

3. Decelerate

4. Turn off Boat

5. Launch Boat

6. Dock Boat

7. Turn right

8. Turn left

* + - * 1. Plane

1. Turn on Plane

2. Take off

3. Accelerate

4. Decelerate

5. Land

6. Turn off Plane

7. Turn right

8. Turn left

* + - * 1. Taxi (same as Car)
    1. Store data
       1. Age of vehicle (default = 0)
       2. Price of vehicle (default = 0.0)
       3. True or false, is car a race car? (default = false)
       4. True or false, is truck a diesel type? (default = false)
       5. Number of passengers
          1. <= 4 in Car
          2. <= 1 in Truck
          3. <= 8 in Boat
          4. <= 200 in Plane
       6. Fuel Capacity
          1. 45 in Car
          2. 5,000 in Truck
          3. 30 in Boat
          4. 50,000 in Plane
       7. Cargo capacity of truck is 100
       8. Propeller level of boat is between 0 and 5
       9. Starting mileage of land vehicle is 1000 miles
       10. Highest altitude of plane: 35,000 ft
           1. 0 = Not in water
           2. 5 = Maximum depth
       11. Cost
           1. Plane = $1,000/trip

Gas price = $4/gal

* + - * 1. Car = $100/trip if taxi car

Gas price = $3/gal

* + - * 1. Boat = $500/trip for renting

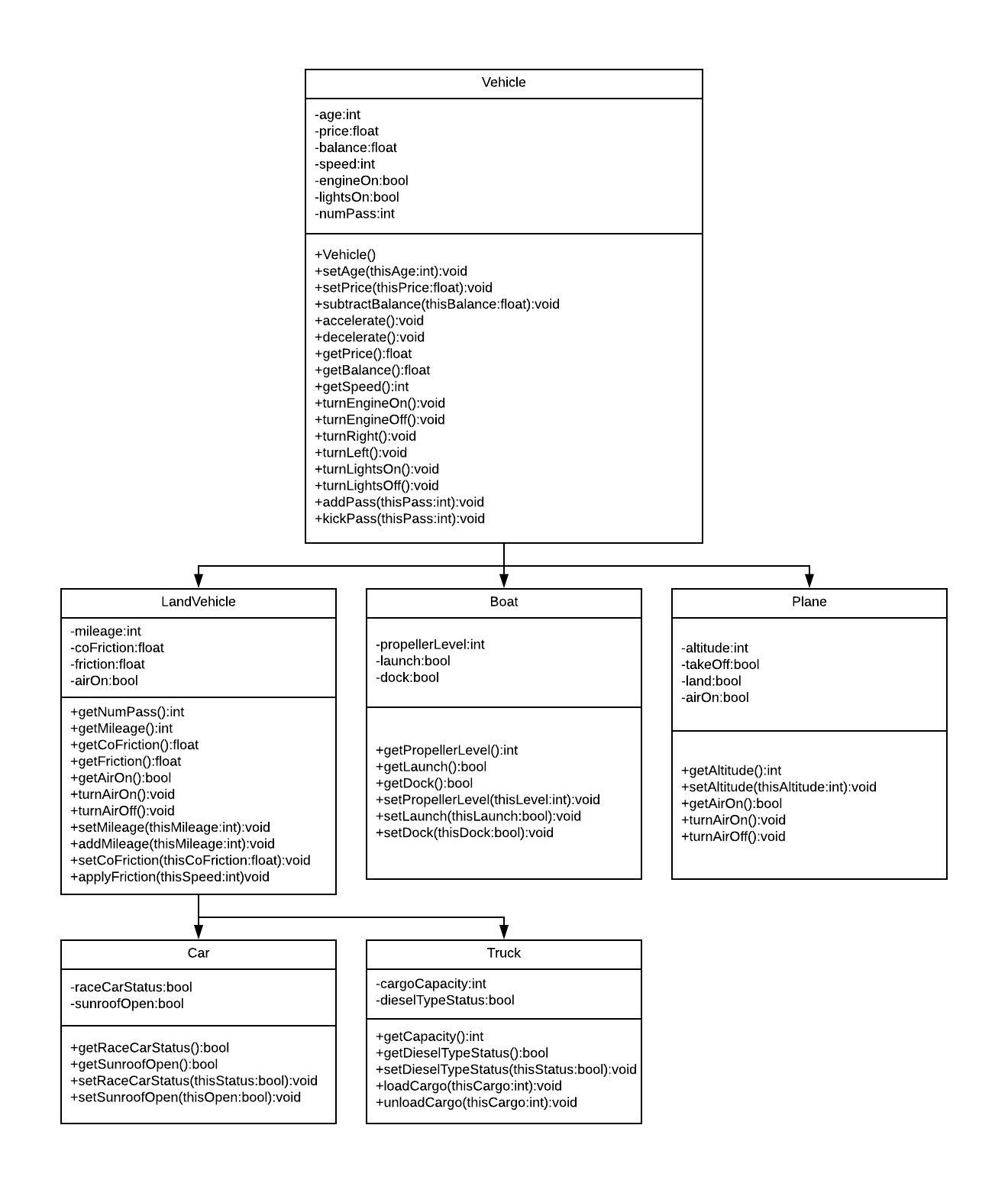
Gas price = $3/gal

* + - * 1. Truck = -$5/trip for delivering cargo

Gas price = $3/gal

* + - 1. Coefficient of friction of both car and truck is between 0 and 1
    1. Manipulate data
       1. Get and set age of each vehicle
       2. Get and set price of each vehicle
       3. Get and set true or false whether or not car is a race car
       4. Get and set true of false whether or not truck is a diesel type
    2. Inheritance
       1. Class *Car* and class *Truck* inherit class *LandVehicle* which inherits class *Vehicle*
       2. Class *Boat* and class *Plane* inherit class *Vehicle*
    3. Thank you message to the driver
    4. Perform checks
       1. Only accept values for age more than what was previously stored and should always be positive
       2. Only accept values for price less than what was previously stored and should always be positive
    5. Store everything in an output file

1. **UML Diagram**



1. **Decomposition Diagram**

|  |  |  |
| --- | --- | --- |
| **Main** | | |
| **Input** | **Process** | **Output** |
| Age | Check if it is more than previously stored value. Store it in the variable age | “Age of vehicle stored” |
| Price | Check if it is less than previously stored value. Store it in variable price | “Price of vehicle stored” |
| Menu selection to get age | Make an appropriate print statement with variable price | Print the statement with age |
| Menu selection to get price | Make an appropriate print statement with variable print | Print the statement with price |
| Race car status | Set the passed in status to the variable raceCarStatus | “Race Car Status stored” |
| Menu selections to get race car status | Make an appropriate print statement with variable raceCarStatus | Print the statement with race car status |
| Diesel type status | Set the passed in status to the variable dieselTypeStatus | “Diesel Type Status stored” |
| Menu Selection of owning which type of vehicle | Make an object and call functions of that particular class selected by user | Confirmation saying user selected the particular vehicle |
| Menu selection of turn lights on / off | Set value of turn lights on to true / false | “Lights are on” |
| Menu selection of turn engine on | Set value of engine on to true if it is previously false | “Lights are off” |
| Menu selection of turn engine off | Set value of engine on to false if it is previously true | “Vehicle turned off” |
| Add this # of passengers | Only add if total # of passengers do not exceed maximum capacity | “Passengers added” |
| Subtract this # of passengers | Only subtract if final # of passengers do not go below 0 | “Passengers kicked out” |
| Balance | Subtract this amount of balance from stored balance | “Balance deducted” |
| Menu selection to accelerate | Set speed to a greater number | “Vehicle is at full speed” |
| Menu selection to decelerate | Set speed to a smaller number | “Vehicle is at a lower speed” |
| Menu selection to turn right | Assign value 2 in a vector at correct location | “You have turned right” |
| Menu selection to turn left | Assign value 1 in a vector at correct location | “You have turned left” |
| Menu selection to turn air on | Set value of air on to true if it is previously false | “Air is on” |
| Menu selection to turn air off | Set value of air on to false if it is previously true | “Air is off” |
| Mileage | Add this mileage to car’s overall mileage | “Mileage added” |
| Coefficient of friction | Set this value to the variable of coefficient of friction and call function to apply friction | “Friction applied” |
| Menu selection to open sunroof | Set value of sun roof open to true if it is previously false | “Sunroof is open” |
| Menu selection to close sunroof | Set value of sun roof open to false if it is previously true | “Sunroof is closed” |
| # of cargo to be loaded | Add this # of cargo to current # of cargo only if total # of cargo does not exceed maximum capacity | “Cargo loaded” |
| # of cargo to be unloaded | Subtract this # of cargo from current # of cargo only if it does not end up with less than 0 cargo | “Cargo unloaded” |
| Propeller level | If the input is between 1 and 5, then assign the input to the variable | “Propeller level set” |
| Menu selection to launch boat | Set value of launch to true and value of dock to false if launch is previously false | “Boat launched” |
| Menu selection to dock boat | Set value of dock to true and value of launch to false if dock is previously false | “Boat docked” |

1. **Test Strategy**
   1. **Valid Data**
   2. **Invalid Data**
2. **Test Plan Version 1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Valid | 1 | Age of vehicle is greater than previously stored value |  |  |  |  |
| Valid | 2 | Price of vehicle is less than previously stored value unless storing it 1st time |  |  |  |  |
| Valid | 3 | Price value is always positive |  |  |  |  |
| Valid | 4 | User enters corresponding number for choosing either true or false for race car status |  |  |  |  |
| Valid | 5 | User enters corresponding number for choosing either true or false for diesel type status |  |  |  |  |
| Valid | 6 | User enters corresponding number for choosing from all vehicles on list |  |  |  |  |
| Valid | 7 | Speed is between 0 mph and 45 mph |  |  |  |  |
| Valid | 8 | Speed is changed while engine is on, speed is not the same as previous, propeller is under water for boat, and boat is not docked |  |  |  |  |
| Valid | 9 | User turns on engine while it was previously off |  |  |  |  |
| Valid | 10 | User turns off engine while speed is 0 mph, engine is previously on, air is off, sun roof is closed, propeller level for boat is 0, boat is docked, and plane altitude is 0 ft |  |  |  |  |
| Valid | 11 | User turns on the lights while they are previously off, and while engine is on |  |  |  |  |
| Valid | 12 | User turns off the lights while they are previously on, and while engine is on |  |  |  |  |
| Valid | 13 | Passengers are added to have total of between 1 and 200, while speed = 0 |  |  |  |  |
| Valid | 14 | Passengers are kicked out to have total of between 0 and 200 |  |  |  |  |
| Valid | 15 | User enters original mileage 1000 |  |  |  |  |
| Valid | 16 | Coefficient of friction is between 0 and 1 |  |  |  |  |
| Valid | 17 | User turns air off while it is previously on and while engine is on |  |  |  |  |
| Valid | 18 | User turns air on while it is previously off, while engine is on, and while sunroof is closed |  |  |  |  |
| Valid | 19 | User opens sunroof while it is previously closed, while air is off, and while engine is on |  |  |  |  |
| Valid | 20 | User closes sunroof while it is previously open and while engine is on |  |  |  |  |
| Valid | 21 | User sets cargo capacity in beginning to be between 1 and 100 |  |  |  |  |
| Valid | 22 | User adds cargo to have total of 100 or less cargo if number of cargo added is not 0 while truck speed = 0 |  |  |  |  |
| Valid | 23 | User enters propeller level to be between 1 and 5 while boat while boat engine is on |  |  |  |  |
| Valid | 24 | User selects to launch boat if it is not previously launched |  |  |  |  |
| Valid | 25 | User selects to dock boat if it is not previously docked |  |  |  |  |
| Valid | 26 | User sets altitude of plane to be between 15,000 ft to 35,000 ft while speed 100 – 140 mph and while altitude is previously 0 |  |  |  |  |
| Valid | 27 | User sets altitude of plane to be 0 while speed 100 – 140 mph and while altitude is previously 15,000 ft to 35,000 |  |  |  |  |
| Invalid | 1 | Age of vehicle is more than previously stored value |  |  |  |  |
| Invalid | 2 | Price of vehicle is more than previously stored value unless storing it 1st time |  |  |  |  |
| Invalid | 3 | Price value is negative |  |  |  |  |
| Invalid | 4 | User enters number not corresponding to choosing either true or false for race car status |  |  |  |  |
| Invalid | 5 | User enters number not corresponding to choosing either true or false for race diesel type status |  |  |  |  |
| Invalid | 6 | User enters number not corresponding to choosing either Car or Truck |  |  |  |  |
| Invalid | 7 | Speed of car / truck is less than 0 |  |  |  |  |
| Invalid | 8 | Speed of car / truck is more than 45 |  |  |  |  |
| Invalid | 9 | User selects to change speed while engine is off |  |  |  |  |
| Invalid | 10 | User selects to change speed to the previously stored speed value |  |  |  |  |
| Invalid | 11 | User selects to change speed while boat propeller level is 0 |  |  |  |  |
| Invalid | 12 | User selects to change speed while boat is not docked |  |  |  |  |
| Invalid | 13 | User selects to turn engine on while it’s on |  |  |  |  |
| Invalid | 14 | User selects to turn engine off while it’s off |  |  |  |  |
| Invalid | 15 | User selects to turn engine off while speed is more than 0 |  |  |  |  |
| Invalid | 16 | User selects to turn engine off while air is on |  |  |  |  |
| Invalid | 17 | User selects to turn engine off while sun roof is open |  |  |  |  |
| Invalid | 18 | User selects to turn engine off while boat propeller level is more than 0 |  |  |  |  |
| Invalid | 19 | User selects to turn engine off while boat is not docked |  |  |  |  |
| Invalid | 20 | User selects to turn engine off while plane altitude is more than 0 |  |  |  |  |
| Invalid | 21 | User selects to turn lights on while they are on |  |  |  |  |
| Invalid | 22 | User selects to turn lights on while engine is off |  |  |  |  |
| Invalid | 23 | User selects to turn lights off while are off |  |  |  |  |
| Invalid | 24 | User selects to add passengers to have total of 0 |  |  |  |  |
| Invalid | 25 | User selects to add passengers to have total of more than 4 passengers in car |  |  |  |  |
| Invalid | 26 | User selects to add passengers to have total of more than 1 passengers in truck |  |  |  |  |
| Invalid | 27 | User selects to add passengers to have total of more than 8 passengers in boat |  |  |  |  |
| Invalid | 28 | User selects to add passengers to have total of more than 200 passengers in plane |  |  |  |  |
| Invalid | 29 | User selects to kick 0 passengers out |  |  |  |  |
| Invalid | 30 | User selects to kick passengers out to have total of negative passengers |  |  |  |  |
| Invalid | 31 | User enters a negative number for mileage |  |  |  |  |
| Invalid | 32 | Coefficient of friction is less than 0 |  |  |  |  |
| Invalid | 33 | Coefficient of friction is more than 1 |  |  |  |  |
| Invalid | 34 | User selects to turn air off while it is off |  |  |  |  |
| Invalid | 35 | User selects to turn air on while it is on |  |  |  |  |
| Invalid | 36 | User selects to turn air on while engine is off |  |  |  |  |
| Invalid | 37 | User selects to turn air on while sunroof is open |  |  |  |  |
| Invalid | 38 | User selects to open sunroof while it is on |  |  |  |  |
| Invalid | 39 | User selects to open sunroof while engine is off |  |  |  |  |
| Invalid | 40 | User selects to open sunroof while air is on |  |  |  |  |
| Invalid | 41 | User selects to close sunroof while it is closed |  |  |  |  |
| Invalid | 42 | User sets cargo capacity in the beginning to be negative |  |  |  |  |
| Invalid | 43 | User sets cargo capacity in the beginning to be greater than 100 |  |  |  |  |
| Invalid | 44 | User adds cargo to have total of more than 100 cargo |  |  |  |  |
| Invalid | 45 | User adds cargo to have total of the same amount as before |  |  |  |  |
| Invalid | 46 | User selects to add cargo while speed of vehicle is greater than 0 |  |  |  |  |
| Invalid | 47 | User enters propeller level to be the same level as before |  |  |  |  |
| Invalid | 48 | User enters propeller level to be negative |  |  |  |  |
| Invalid | 49 | User enters propeller level to be more than 5 |  |  |  |  |
| Invalid | 50 | User selects to change propeller level while engine is off |  |  |  |  |
| Invalid | 51 | User selects to launch boat while it is already launched |  |  |  |  |
| Invalid | 52 | User selects to dock boat while it is already docked |  |  |  |  |
| Invalid | 53 | User selects to set altitude of plane to be less than 15,000 ft while speed is 100 – 140 mph |  |  |  |  |
| Invalid | 54 | User selects to set altitude of plane to be more than 35,000 ft while speed is 100 – 140 mph |  |  |  |  |
| Invalid | 55 | User selects to set altitude of plane to be between 15,000 ft to 35,000 ft while speed is less than 100 |  |  |  |  |
| Invalid | 56 | User selects to set altitude of plane to be between 15,000 ft to 35,000 ft while speed is more than 140 |  |  |  |  |
| Invalid | 57 | User selects to increase speed of plane while it is previously 500 mph (max) |  |  |  |  |
| Invalid | 58 | User selects to decrease speed of plane while it is previously 120 mph (min) |  |  |  |  |
| Invalid | 59 | User selects to set altitude of plane to be 0 while speed is more than 140 mph |  |  |  |  |
| Invalid | 60 | User selects to set altitude of plane to be 0 while plane is already on land |  |  |  |  |
| Invalid | 61 | User selects to change altitude while engine is off |  |  |  |  |

1. **Initial Algorithm**
   1. In class Vehicle
      1. Make private variables
         1. Age (int)
         2. Price (float)
         3. Balance (float)
         4. Speed (int)
         5. engineOn (Boolean)
         6. lightsOn (Boolean)
         7. numPass (int)
      2. Make a vector called turns to store turns that users take
      3. Constructor
         1. Set age to 0
         2. Set price to 0.0
         3. Set balance to $4,000
         4. Set speed = 0
         5. Set engineOn to false
         6. Set lightsOn to false
         7. Set numPass to 0
      4. In *setAge()* function
         1. Set vehicle’s age to be the value passed in
            1. Validate the vehicle’s age passed in is greater than previously stored value if not setting the age for 1st time
            2. If setting the age for 1st time, validate that value passed in is positive
      5. In *setPrice()* function
         1. Set vehicle’s price to be the value passed in
            1. Validate the vehicle’s price passed in is smaller than previously stored value if not setting the price for 1st time
            2. If setting the price for 1st time, validate that value passed in is positive
      6. In *getAge()* function
         1. Return vehicle’s age
      7. In *getPrice()* function
         1. Return vehicle’s price
      8. In *subtractBalance()* function
         1. Subtract passed in value from current balance
      9. In *accelerate()* function
         1. Add 5 to speed
      10. In *decelerate()* function
          1. Subtract 5 from speed
      11. In *getBalance()* function
          1. Return balance
      12. In *getSpeed()* function
          1. Return speed
      13. In *turnEngineOn()* function
          1. Set engineOn variable to true
      14. In *turnEngineOff()* function
          1. Set entingOn variable o false
      15. In *turnRight()* function
          1. Store number 3 in first empty space in the vector
      16. In *turnLeft()* function
          1. Store number 1 in the first empty space in the vector
      17. In *turnLightsOn()* function
          1. Set lightsOn variable to true
      18. In *turnLightsOff()* function
          1. Set lightsOn variable to false
      19. In *addPass()* function
          1. Add passed in value to current number of passengers
      20. In *kickPass()* function
          1. Subtract passed in value from current number of passengers
   2. In class LandVehicle (inherits class Vehicle)
      1. Private variables
         1. Mileage (int) to add mileage of each vehicle
         2. Coefficient of friction (int) to calculate friction
         3. Friction (float) to be calculated by multiplying coefficient of friction by speed
         4. Air on (Boolean) to hold status of A/C on or off
      2. Constructor
         1. Set mileage to 1000
         2. Set coefficient of friction to 0.5
         3. Call function to calculate friction
         4. Set air on status to false
      3. In *getNumPass()* function
         1. Return number of passengers currently in the car / truck
      4. In *getMileage* function
         1. Return current mileage of car / truck
      5. In *getCoFriction* function
         1. Return coefficient of friction
      6. In *getFriction()* function
         1. Return calculated friction
      7. In *getAirOn()* function
         1. Return air on status
      8. In *turnAirOn()* function
         1. Set air on status to true
      9. In *turnAirOff()* function
         1. Set air on status to false
      10. In *addMileage()* function
          1. Add passed in value to current mileage of vehicle
      11. In *setCoFriction()* function
          1. Set the passed in value to the variable of coefficient of friction
      12. In *applyFriction()* function
          1. Friction = coefficient of friction \* speed
   3. In class Car (inherits class LandVehicle)
      1. Private variable for race car status (bool)
      2. Private variable for sun roof open (bool)
      3. Constructor
         1. Set race car status to false
         2. Sets sun roof open status to false
      4. In *setRaceCarStatus()* function
         1. Set race car status to be the value passed in
      5. In *getRaceCarStatus()* function
         1. Return race car status
      6. In *getSunroofOpen()* function
         1. Return sunroof open status
      7. In *setSunroofOpen()* function
         1. Set sunroof status to be the value passed in
   4. In class Truck (inherits class LandVehicle)
      1. Private variable for diesel type status (bool)
      2. Private variable for cargo capacity (int)
      3. Public variable for maximum cargo capacity (int)
      4. Constructor
         1. Set diesel type status to false
         2. Set cargo capacity to 0
         3. Set maximum cargo capacity to 100
      5. In *setDieselTypeStatus()* function
         1. Set diesel type status to be the value passed in
      6. In *getDieselTypeStatus()* function
         1. Return diesel type status
      7. In *getCapacity()* function
         1. Return the current number of cargo in the truck
      8. In *loadCargo()* function
         1. Add the value passed in to the current number of cargo stored
      9. In *unloadCargo()* function
         1. Subtract the value passed in from the current number of cargo stored
   5. In class Boat (inherits class Vehicle)
      1. Private variables
         1. Propeller level (int)
            1. 0 = Outside of water
            2. 1 to 5 = under water
         2. Launch (Boolean)
         3. Dock (Boolean)
      2. Constructor
         1. Set propeller level to 0
         2. Set launch to false
         3. Set dock to true
      3. In *getPropellerLevel()* function
         1. Return current propeller level
      4. In *getLaunch()* function
         1. Return current launch status
      5. In *getDock()* function
         1. Return current dock status
      6. In *setPropellerLevel()* function
         1. Set the current propeller level to be the passed in value
      7. In *setLaunch()* function
         1. Set the current launch status to be the status passed in
      8. In *setDock()* function
         1. Set the current dock status to be the status passed in
   6. In class Plane (inherits class Vehicle)
      1. Private variables
         1. Altitude (int)
         2. Take off status (bool) to store that plane is in air
         3. Land status (bool) to store that plane is in air
      2. Constructor
         1. Set altitude to 0
         2. Set take off status to false
         3. Set land status to true
      3. In *getAltitude()* function
         1. Return current altitude of plane
      4. In *setAltitude()* function
         1. Set current altitude to the value passed in
      5. In *getAirOn()* function
         1. Return air on status
      6. In *turnAirOn()* function
         1. Set air on status to true
      7. In *turnAirOff()* function
         1. Set air on status to false
   7. In class Main
      1. Make variables to be used later
      2. Make vector for turns a car, truck or boat makes.
      3. Make vector for storing locations user goes
      4. Make vector for storing which vehicle user uses each time
      5. Do following, then loop while user chooses to quit program
         1. If user is at home, print a menu with options Car, Truck, Quit
            1. If user selects Car

Using switch case, print and call functions for all the different things user can do and loop until user says they have reached destination

Set age if user gives an age more than or equal to previous age

Set price if user gives a price less than or equal to previous price

Return balance

Turn engine on after performing checks

Turn engine off after performing checks

Turn lights on / off after performing checks

Add / kick passengers after performing checks

Return mileage

Set coefficient of friction after performing checks

Return friction

Turn air on / off after performing checks

Set race car status after performing checks

Open / close sunroof after performing checks

Turn right – insert 3 in the first empty box in vector

Turn left – insert 1 in the first empty box in vector

Reached destination – ask if user wants to return to last location. If yes, quit loop and print returning messages

* + - * 1. If user selects Truck

Using switch case, print and call functions for all the different things user can do and loop until user says they have reached destination

Set age if user gives an age more than or equal to previous age

Set price if user gives a price less than or equal to previous price

Return balance

Turn engine on after performing checks

Turn engine off after performing checks

Turn lights on / off after performing checks

Add / kick passengers after performing checks

Return cargo capacity

Add / Subtract cargo after performing checks

Return mileage

Set coefficient of friction after performing checks

Return friction

Turn air on / off after performing checks

Set diesel type status after performing checks

Turn right – insert 3 in the first empty box in vector

Turn left – insert 1 in the first empty box in vector

Reached destination – ask if user wants to return to last location. If yes, quit loop and print returning messages

* + - * 1. If user selects quit, then break out the loop
      1. If user is at airport, print a menu with options Plane, Car, Truck, Quit
         1. If user selects Plane

Using switch case, print and call functions for all the different things user can do and loop until user says they have reached destination

Set age if user gives an age more than or equal to previous age

Set price if user gives a price less than or equal to previous price

Return balance

Turn engine on after performing checks

Turn engine off after performing checks

Turn lights on / off after performing checks

Add / kick passengers after performing checks

Turn air on / off after performing checks

Reached destination – ask if user wants to return to last location. If yes, quit loop and print returning messages

Return altitude

Set altitude after performing checks

Return air on status

* + - * 1. If user selects Car, using switch case, print and call functions for all the different things user can do and loop until user says they have reached destination
        2. If user selects Truck, using switch case, print and call functions for all the different things user can do and loop until user says they have reached destination
        3. If user selects Quit, break out of the loop and ask which location is the user at and store it in the vector
      1. If user is at marina, print a menu with options Boat, Car, Truck, Quit
         1. If user selects Boat

Using switch case, print and call functions for all the different things user can do and loop until user says they have reached destination

Set age if user gives an age more than or equal to previous age

Set price if user gives a price less than or equal to previous price

Return balance

Turn engine on after performing checks

Turn engine off after performing checks

Turn lights on / off after performing checks

Add / kick passengers after performing checks

Set / change launch / dock status after performing checks

Set propeller level after performing checks

Reached destination – ask if user wants to return to last location. If yes, quit loop and print returning messages

* + - * 1. If user selects Car, using switch case, print and call functions for all the different things user can do and loop until user says they have reached destination
        2. If user selects Truck, using switch case, print and call functions for all the different things user can do and loop until user says they have reached destination
        3. If user selects Quit, break out of the loop and ask which location is the user at and store it in the vector

1. **Test Plan Version 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Valid | 1 | Age of vehicle is greater than previously stored value | Previous: “5”  New: “10” | Age stored successfully |  |  |
| Valid | 2 | Price of vehicle is less than previously stored value unless storing it 1st time | Previous: “15000”  New: “10000” | Price stored successfully |  |  |
| Valid | 3 | Price value is always positive | “15000” | “Price stored successfully” |  |  |
| Valid | 4 | User enters corresponding number for choosing either true or false for race car status | “1” for true | “Race car status updated” |  |  |
| Valid | 5 | User enters corresponding number for choosing either true or false for diesel type status | “1” for true | “Diesel type status updated” |  |  |
| Valid | 6 | User enters corresponding number for choosing from all vehicles on list | “1” for Car when 1: Car  2: Truck  3: Quit | “You are driving car now” |  |  |
| Valid | 7 | Speed is between 0 mph and 45 mph | “40” | “Speed is now 40” |  |  |
| Valid | 8.1 | Speed of car / truck is changed while engine is on, speed is not the same as previous | Engine: On  Speed previous: 40  Speed new: “45” | “Speed is now 45” |  |  |
| Valid | 8.2 | Speed of boat is changed while engine is on, speed is not the same as previous, propeller is under water for boat, and boat is not docked | Engine: On  Speed previous: 40  Speed new: “45”  Propeller level: 2  Boat: launched | “Speed is now 45” |  |  |
| Valid | 9 | User turns on engine while it was previously off | Previous: engine off  New: engine on | “Engine is now on” |  |  |
| Valid | 10.1 | User turns off engine while speed is 0 mph, engine is previously on, air is off, sun roof is closed | Speed: 0  Previous: engine on  New: engine off  Air: off  Sunroof: closed | “Engine is now off” |  |  |
| Valid | 10.2 | User turns off engine while speed is 0 mph, engine is previously on, propeller level for boat is 0, boat is docked | Speed: 0  Previous: engine on  New: engine off  Propeller level: 0  Boat: docked | “Engine is now off” |  |  |
| Valid | 10.3 | User turns off engine while speed is 0 mph, engine is previously on, air is off, and plane altitude is 0 ft | Speed: 0  Previous: engine on  New: engine off  Air: off  Altitude: 0 | “Engine is now off” |  |  |
| Valid | 11 | User turns on the lights while they are previously off, and while engine is on | Previous: Lights off  New: lights on  Engine: on | “Lights are now on” |  |  |
| Valid | 12 | User turns off the lights while they are previously on, and while engine is on | Previous: Lights on  New: lights off  Engine: on | “Lights are now off” |  |  |
| Valid | 13.1 | Passengers are added to have total of between 1 and 200 for plane, while speed = 0 | Previous: 50  Add: 100  Total: 150  Speed: 0 | “Passengers added successfully” |  |  |
| Valid | 13.2 | Passengers are added to have total of between 1 and 4 for Car, while speed = 0 | Previous: 2  Add: 2  Total: 4  Speed: 0 | “Passengers added successfully” |  |  |
| Valid | 13.3 | Passengers are added to have total of 1 for Truck, while speed = 0 | Previous: 0  Add: 1  Total: 1  Speed: 0 | “Passengers added successfully” |  |  |
| Valid | 13.4 | Passengers are added to have total of 8 for Boat, while speed = 0 | Previous: 4  Add: 4  Total: 8  Speed: 0 | “Passengers added successfully” |  |  |
| Valid | 14 | Passengers are kicked out to have total of between 0 and 200 for plane, while speed = 0 | Previous: 150  Kick: 50  Total:100  Speed = 0 | “Passengers kicked out successfully” |  |  |
| Valid | 15 | User enters original mileage 1000 | “1000” | “Mileage see successfully” |  |  |
| Valid | 16 | Coefficient of friction is between 0 and 1 | “0.5” | “Coefficient is updated successfully” |  |  |
| Valid | 17 | User turns air off while it is previously on and while engine is on | Previous: air on  New: air off  Engine: on | “Air is not off” |  |  |
| Valid | 18 | User turns air on while it is previously off, while engine is on, and while sunroof is closed | Previous: air off  New: air on  Engine: on  Sunroof: closed | “Air is now on” |  |  |
| Valid | 19 | User opens sunroof while it is previously closed, while air is off, and while engine is on | Previous: sunroof closed  New: sunroof open  Air: off  Engine on | “Sunroof is now open” |  |  |
| Valid | 20 | User closes sunroof while it is previously open and while engine is on | Previous: sunroof open  New: sunroof closed  Engine: on | “Sunroof is now closed” |  |  |
| Valid | 21 | User sets cargo capacity in beginning to be between 1 and 100 | “50” | “Current cargo capacity set” |  |  |
| Valid | 22 | User adds cargo to have total of 100 or less cargo if number of cargo added is not 0 while truck speed = 0 | Previous: 50  Add: 25  Total: 75  Speed: 0 | “Cargo added successfully” |  |  |
| Valid | 23 | User enters propeller level to be between 1 and 5 for boat while boat engine is on | Level: 2  Engine: on | “Propeller level updated successfully” |  |  |
| Valid | 24 | User selects to launch boat if it is not previously launched | Select “launch”  Previous: docked | “Boat is now launched” |  |  |
| Valid | 25 | User selects to dock boat if it is not previously docked | Select “dock”  Previous: launched | “Boat is now docked” |  |  |
| Valid | 26 | User sets altitude of plane to be between 15,000 ft to 35,000 ft while speed 100 – 140 mph and while altitude is previously 0 | Speed: 130  Set new altitude: “30000”  Previous altitude: 0 | “Altitude of plane updated successfully” |  |  |
| Valid | 27 | User sets altitude of plane to be 0 while speed 100 – 140 mph and while altitude is previously 15,000 ft to 35,000 | Speed: 110  Set new altitude: “0”  Previous altitude: 18000 | “Altitude of plane updated successfully” |  |  |
| Invalid | 1 | Age of vehicle is less than previously stored value | Previous: 15  New: 10 | “Invalid input: Age cannot be less than last one” |  |  |
| Invalid | 2 | Price of vehicle is more than previously stored value unless storing it 1st time | Previous: 15000  New: 20000 | “Invalid input: Price cannot be more than last one” |  |  |
| Invalid | 3 | Price value is negative | “-25000” | “Invalid input: Price cannot be negative” |  |  |
| Invalid | 4 | User enters number not corresponding to choosing either true or false for race car status | “3” | “Invalid input: Try again” |  |  |
| Invalid | 5 | User enters number not corresponding to choosing either true or false for race diesel type status | “100” | “Invalid input: Try again” |  |  |
| Invalid | 6 | User enters number not corresponding to choosing either Car or Truck | “-4” | “Invalid input: Try again” |  |  |
| Invalid | 7 | Speed of car / truck is less than 0 | “-15” | “Invalid input: Speed cannot be less than 0” |  |  |
| Invalid | 8 | Speed of car / truck is more than 45 | “55” | “Invalid input: Speed cannot be more than 45” |  |  |
| Invalid | 9 | User selects to change speed while engine is off | Engine: off  Previous speed: 0  New: 45 | “Invalid input: Cannot change speed while engine is off” |  |  |
| Invalid | 10 | User selects to change speed to the previously stored speed value | Previous: 40  New: 40 | “Invalid input: Cannot change speed to the same value as before” |  |  |
| Invalid | 11 | User selects to change speed while boat propeller level is 0 | Previous: 0  New: 30  Propeller level: 0 | “Invalid input: Cannot change speed while boat propeller level is not in water” |  |  |
| Invalid | 12 | User selects to change speed while boat is not launched | Previous: 0  New: 20  Boat: docked | “Invalid input: Cannot change speed while boat is docked” |  |  |
| Invalid | 13 | User selects to turn engine on while it’s on | Previous:  Engine on  New:  Engine on | “Invalid input: Cannot turn engine on while it is already on” |  |  |
| Invalid | 14 | User selects to turn engine off while it’s off | Previous: Engine off  New: Engine off | “Invalid input: Cannot turn engine off while is already off” |  |  |
| Invalid | 15 | User selects to turn engine off while speed is more than 0 | Previous: engine on  New: engine off  Speed: 14 | “Invalid input: Cannot turn engine off while vehicle is moving” |  |  |
| Invalid | 16 | User selects to turn engine off while air is on | Air: on  Previous: engine on  New: engine off | “Invalid input: Cannot turn engine off while A/C is on” |  |  |
| Invalid | 17 | User selects to turn engine off while sun roof is open | Sunroof: open  Previous: engine on  New: engine off | “Invalid input: Cannot turn engine of while sunroof is open” |  |  |
| Invalid | 18 | User selects to turn engine off while boat propeller level is more than 0 | Propeller level: 1  Previous: engine on  New: engine off | “Invalid input: Cannot turn engine off while boat propeller is inside water” |  |  |
| Invalid | 19 | User selects to turn engine off while boat is not docked | Boat: launched  Previous: engine on  New: engine off | “Invalid input: Cannot turn engine off while boat is launched” |  |  |
| Invalid | 20 | User selects to turn engine off while plane altitude is more than 0 | Altitude: 20,000  Previous: engine on  New: engine off | “Invalid input: Cannot turn engine off while plane is in air” |  |  |
| Invalid | 21 | User selects to turn lights on while they are on | Previous: lights on  New: lights on | “Invalid input: Cannot turn lights on if they already on” |  |  |
| Invalid | 22 | User selects to turn lights on while engine is off | Engine: off  Previous: lights off  New: lights on | “Invalid input: Cannot turn lights on if engine is off” |  |  |
| Invalid | 23 | User selects to turn lights off while are off | Previous: lights off  New: lights off | “Invalid input: Cannot turn lights off while they are already off” |  |  |
| Invalid | 24 | User selects to add passengers to have total of 0 | Previous passengers: 0  Add: “0”  Total: 0 | “Invalid input: Cannot add 0 passengers” |  |  |
| Invalid | 25 | User selects to add passengers to have total of more than 4 passengers in car | Previous passengers: 2  Add: 3  Total: 5 | “Invalid input: Cannot add passengers to have total of more than 4” |  |  |
| Invalid | 26 | User selects to add passengers to have total of more than 1 passengers in truck | Previous passengers: 0  Add: 2  Total: 2 | “Invalid input: Cannot add passengers to have total of more than 1” |  |  |
| Invalid | 27 | User selects to add passengers to have total of more than 8 passengers in boat | Previous passengers: 3  Add: 10  Total: 13 | “Invalid input: Cannot add passengers to have total of more than 8” |  |  |
| Invalid | 28 | User selects to add passengers to have total of more than 200 passengers in plane | Previous passengers: 100  Add: 150  Total: 250 | “Invalid input: Cannot add passengers to have total of more than 200” |  |  |
| Invalid | 29 | User selects to kick 0 passengers out | Kick “0” passengers out | “Invalid input: Cannot kick 0 passengers out” |  |  |
| Invalid | 30 | User selects to kick passengers out to have total of negative passengers | Previous passengers: 2  Kick: 3  Total: -1 | “Invalid input: Cannot kick out more passengers than there are” |  |  |
| Invalid | 31 | User enters a negative number for mileage | “-1000” | “Invalid input: Cannot have negative mileage” |  |  |
| Invalid | 32 | Coefficient of friction is less than 0 | “-1” | “Invalid input: Cannot have negative coefficient of friction” |  |  |
| Invalid | 33 | Coefficient of friction is more than 1 | “2” | “Invalid input: Cannot have coefficient of friction more than 2” |  |  |
| Invalid | 34 | User selects to turn air off while it is off | Previous air: off  New air: off | “Invalid input: Cannot turn engine off if it is already off” |  |  |
| Invalid | 35 | User selects to turn air on while it is on | Previous air: on  New air: on | “Invalid input: Cannot turn A/C on while it is already on” |  |  |
| Invalid | 36 | User selects to turn air on while engine is off | Engine: off  Previous air: off  New air: on | “Invalid input: Cannot turn air on if engine is off” |  |  |
| Invalid | 37 | User selects to turn air on while sunroof is open | Sunroof: on  Previous air: off  New air: on | “Invalid input: Cannot turn air on if sunroof is open” |  |  |
| Invalid | 38 | User selects to open sunroof while it is on | Previous sunroof: open  New sunroof: open | “Invalid input: Cannot open sunroof if it is already open” |  |  |
| Invalid | 39 | User selects to open sunroof while engine is off | Engine: off  Previous sunroof: closed  New sunroof: open | “Invalid input: Cannot open sunroof if engine is off” |  |  |
| Invalid | 40 | User selects to open sunroof while air is on | Air: on  Previous sunroof: closed  New sunroof: open | “Invalid input: Cannot open sunroof if A/C is on” |  |  |
| Invalid | 41 | User selects to close sunroof while it is closed | Previous sunroof: closed  New sunroof: closed | “Invalid input: Cannot close sunroof if it is already closed” |  |  |
| Invalid | 42 | User sets cargo capacity in the beginning to be negative | “-50” | “Invalid input: Cannot have a negative cargo” |  |  |
| Invalid | 43 | User sets cargo capacity in the beginning to be greater than 100 | “150” | “Invalid input: Cannot have more than 100 cargo” |  |  |
| Invalid | 44 | User adds cargo to have total of more than 100 cargo | Previous cargo: 10  Add: “100”  Total: 110 | “Invalid input: Cannot add cargo to have total of more than 100 cargo” |  |  |
| Invalid | 45 | User adds cargo to have total of the same amount as before | Previous cargo: 75  Add: 0  Total: 75 | “Invalid input: Cannot add 0 cargo” |  |  |
| Invalid | 46 | User selects to add cargo while speed of vehicle is greater than 0 | Speed: 40  Select add cargo | “Invalid input: Cannot add cargo if vehicle is moving” |  |  |
| Invalid | 47 | User enters propeller level to be the same level as before | Previous level: 1  New: “1” | “Invalid input: Cannot set propeller level to be same as before” |  |  |
| Invalid | 48 | User enters propeller level to be negative | “-4” | “Invalid input: Propeller level cannot be negative” |  |  |
| Invalid | 49 | User enters propeller level to be more than 5 | “10” | “Invalid input: Propeller level cannot be more than 5” |  |  |
| Invalid | 50 | User selects to change propeller level while engine is off | Engine: off  Previous level: 4  New: 5 | “Invalid input: Cannot change propeller level if engine is off” |  |  |
| Invalid | 51 | User selects to launch boat while it is already launched | Previous boat: launched  New: Launch | “Invalid input: Cannot launch boat if it is already launched” |  |  |
| Invalid | 52 | User selects to dock boat while it is already docked | Pervious boat: docked  New: docked | “Invalid input: Cannot dock boat if it is already docked” |  |  |
| Invalid | 53 | User selects to set altitude of plane to be less than 15,000 ft while speed is 141-500 mph while landing / taking off | Speed: 135  Previous altitude: 15000  Set altitude: “1” | “Invalid input: Cannot change altitude to be less than 15,000 if speed is 141 – 500 mph ” |  |  |
| Invalid | 54 | User selects to set altitude of plane to be more than 35,000 ft while speed is 100-140 mph while landing / taking off | Speed: 135  Previous altitude: 0  Set altitude: “99000” | “Invalid input: Plane cannot go above altitude of 35,000” |  |  |
| Invalid | 55 | User selects to set altitude of plane to be between 15,000 ft to 35,000 ft while speed is less than 100 while landing / taking off | Speed: 99  Previous altitude: 0  Set altitude: “30150” | “Invalid input: Cannot change altitude if speed is less than 100 mph” |  |  |
| Invalid | 56 | User selects to set altitude of plane to be between 15,000 ft to 35,000 ft while speed is more than 140 while landing / taking off | Speed: 141  Previous altitude: 0  Set altitude: “30150” | “Invalid input: Cannot change altitude if speed is more than 140 mph” |  |  |
| Invalid | 57 | User selects to increase speed of plane while it is previously 500 mph (max) | Previous speed: 500  Add: 1  Total: 501 | “Invalid input: Plane cannot go more than 500 mph” |  |  |
| Invalid | 58 | User selects to decrease speed of plane while it is previously 120 mph (min) | Previous speed: 120  Subtract: 1  Total: 119 | “Invalid input: Plane cannot go less than 120 mph to maintain altitude” |  |  |
| Invalid | 59 | User selects to set altitude of plane to be 0 while speed is more than 140 mph | Speed: 145  Previous altitude: 15001  Set altitude: 0 | “Invalid input: Cannot land plane if speed is more than 140 mph” |  |  |
| Invalid | 60 | User selects to set altitude of plane to be 0 while plane is already on land | Previous altitude: 0  Speed 120  Set altitude: 0 | “Invalid input: Cannot land plane if it is already on ground” |  |  |
| Invalid | 61 | User selects to change altitude while engine is off | Engine: off  Set altitude: “15001” | “Invalid input: Cannot change altitude if engine is off” |  |  |

1. **Code**

#include "Car.h"

#include "Truck.h"

#include "Boat.h"

#include "Plane.h"

#include <string>

#include <fstream>

#include <algorithm>

#include <vector>

#include <iostream>

using namespace std;

void printCarMenu();

void printTruckMenu();

void printBoatMenu();

void printPlaneMenu();

int main()

{

//Output file

ofstream fout;

fout.open("Outputs.txt");

//Variables...Vectors

int userMenu, userVehicle, userInput, count, userSelection, numPass, tempStat, userCapacity, userAlt, userPropeller;

float coFriction;

bool endOfMenu = false;

int userLocation; //1.Home, 2.Airport, 3.Marina

vector<int> location;

vector<int> turnsTripOne;

vector<int> turnsTripTwo;

vector<int> turnsTripThree;

vector<int> vehicle;

//Vehicles

Car Lambo; //Vehicle 1

Car Taxi; //...2

Truck HaulTruck; //...3

Truck TankTruck; //...4

Boat Yacht; //...5

Boat Hovercraft; //...6

Plane AirbusA310; //...7

Plane Boeing747; //...8

//Welcome message

cout << "Welcome, user to an amazing program where you will get to drive and fly many different types of vehicles.\nDON'T FORGET THE SEATBELT!!!" << endl;

fout << "Welcome message" << endl;

system("pause");

system("CLS");

location.push\_back(1);

fout << "User starts at home" << endl;

//................... . . . . . . . . . . . . . . . . . . . . . . . . .

do {

if (location.back() == 1) //If user is at home

{

cout << "You are at home right now" << endl;

cout << "1. Take the Car" << endl;

cout << "2. Take the Truck" << endl;

cin >> userMenu;

if (userMenu == 1) //If user selected Car

{

vehicle.push\_back(1);

fout << "User takes the car" << endl;

do {

system("CLS");

cout << "You are driving Car" << endl;

printCarMenu();

cin >> userInput;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

switch (userInput) {

case 1: //Set age of car

cout << "What is the age of car now?: ";

cin >> userSelection;

if (userSelection < Lambo.getAge())

{

cout << "Invalid input. Age cannot be less than last one" << endl;

system("pause");

}

else if (userSelection == Lambo.getAge())

{

cout << "The age you entered is the same age stored earlier" << endl;

system("pause");

}

else {

Lambo.setAge(userSelection);

cout << "Age stored successfully" << endl;

fout << "Age stored as " << userSelection << endl;

system("pause");

}

break;

case 2: //Set price of car

cout << "What is the price of your car now?: ";

cin >> userSelection;

if (userSelection > Lambo.getPrice())

{

cout << "Invalid input. Price cannot be more than last one" << endl;

system("pause");

}

else if (userSelection == Lambo.getPrice())

{

cout << "The price you enteres it the same price stored earlier" << endl;

system("pause");

}

else if (userSelection < 0)

{

cout << "Invalid input: Price cannot be negative" << endl;

system("Pause");

}

else {

Lambo.setPrice(userSelection);

cout << "Price stored successfully" << endl;

fout << "Price is stored as " << userSelection << endl;

system("pause");

}

break;

case 3: //Check balance of user

if (vehicle.at(0) == 1)

{

cout << "Your balance is " << Lambo.getBalance() << endl;

}

else if (vehicle.at(0) == 4)

{

cout << "Your balance is " << TankTruck.getBalance() << endl;

}

system("pause");

break;

case 4: //Turn engine on

if (Lambo.getEngineOn())

{

cout << "Invalid input: Cannot turn engine on while it is already on" << endl;

}

else {

Lambo.turnEngineOn();

cout << "Engine is now on" << endl;

fout << "User turned on the engine" << endl;

}

system("pause");

break;

case 5: //Turn engine off

if (Lambo.getEngineOn() == false || Lambo.getSpeed() != 0 || Lambo.getAirOn() == true || Lambo.getSunroofOpen() == true || Lambo.getLightsOn())

{

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn engine off while it is already off" << endl;

}

if (Lambo.getSpeed() != 0)

{

cout << "Invalid input: Cannot turn engine off while vehicle is moving" << endl;

}

if (Lambo.getAirOn() == true)

{

cout << "Invalid input: Cannot turn engine off while A/C is on" << endl;

}

if (Lambo.getSunroofOpen() == true)

{

cout << "Invalid input: Cannot turn engine off while sunroof is open" << endl;

}

if (Lambo.getLightsOn())

{

cout << "Invalid input: Cannot turn engine off while lights are on" << endl;

}

}

else {

Lambo.turnEngineOff();

cout << "Engine is now off" << endl;

fout << "User turned off the engine" << endl;

}

system("pause");

break;

case 6: //Turn lights on

if (Lambo.getLightsOn() || Lambo.getEngineOn() == false)

{

if (Lambo.getLightsOn())

{

cout << "Invalid input: Cannot turn lights on if they are already on" << endl;

}

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn lights on if engine is off" << endl;

}

}

else {

Lambo.turnLightsOn();

cout << "Lights are now on" << endl;

fout << "User turned on the lights" << endl;

}

system("pause");

break;

case 7: //Turn lights off

if (Lambo.getEngineOn() == false || Lambo.getLightsOn() == false)

{

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn turn lights off while engine is off" << endl;

}

if (Lambo.getLightsOn() == false)

{

cout << "Invalid input: Cannot turn lights off while they are already off" << endl;

}

}

else {

Lambo.turnLightsOff();

cout << "Lights are now off" << endl;

fout << "User turned off the lights" << endl;

}

system("pause");

break;

case 8: //Add Passengers

if (Lambo.getSpeed() != 0)

{

cout << "Invalid input: Cannot add passengers while vehicle is moving" << endl;

}

else {

cout << "How many passengers do you want to add? ";

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot add 0 passengers" << endl;

}

else if (Lambo.getNumPass() + numPass > 4)

{

cout << "Invlalid input: Cannot add passengers to have total of more than 4" << endl;

}

else {

Lambo.addPass(numPass);

cout << "Passengers added successfully" << endl;

fout << "User added " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 9: //Kick Passengers

if (Lambo.getSpeed() != 0)

{

cout << "Invalid input: Cannot kick passengers while vehicle is moving\nThey will die" << endl;

}

else {

cout << "How many passengers do you want to kick out" << endl;

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot kick 0 passengers" << endl;

}

else if (numPass > Lambo.getNumPass())

{

cout << "Invalid input: Cannot kick more passengers than there are" << endl;

}

else {

Lambo.kickPass(numPass);

cout << "Passengers kicked out successfully successfully" << endl;

fout << "User kicked out " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 10: //Check mileage

cout << "You car's mileage is: " << Lambo.getMileage() << " miles" << endl;

system("pause");

break;

case 11: //Set coefficient of friction

cout << "What is the current coefficient of friction" << endl;

cin >> coFriction;

if (coFriction < 0)

{

cout << "Invalid input: Cannot have negative coefficient of friction" << endl;

}

else if (coFriction > 1)

{

cout << "Invalid input: Cannot have coefficient of friction more than 1" << endl;

}

else {

Lambo.setCoFriction(coFriction);

cout << "Coefficient is updated successfully" << endl;

fout << "Coefficient is updated to " << coFriction << endl;

}

system("pause");

break;

case 12: //Check friction

cout << "Current friction on your car is: " << Lambo.getFriction();

system("pause");

break;

case 13: //Turn air on

if (Lambo.getAirOn() || Lambo.getEngineOn() == false || Lambo.getSunroofOpen())

{

if (Lambo.getAirOn())

{

cout << "Invalid input: Cannot turn A/C on while it is already on" << endl;

}

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn air on if engine is off" << endl;

}

if (Lambo.getSunroofOpen())

{

cout << "Invalid input: Cannot turn air on if sunroof is open" << endl;

}

}

else {

Lambo.turnAirOn();

cout << "Air is now on" << endl;

fout << "User turned on the air" << endl;

}

system("pause");

break;

case 14: //Turn air off

if (Lambo.getAirOn() == false)

{

cout << "Invalid input: Cannot turn air off if it is already off" << endl;

}

else {

Lambo.turnAirOff();

cout << "Air is now off" << endl;

fout << "User turned off the air" << endl;

}

system("pause");

break;

case 15: //Change race car status

cout << "Type 1 if it IS a race car and type 0 if it IS NOT a race car: ";

cin >> tempStat;

if (tempStat < 0 || tempStat > 1)

{

cout << "Invalid input: Only accepted values are 0 and 1" << endl;

}

else {

Lambo.setRaceCarStatus(tempStat);

cout << "Race Car status updated" << endl;

if (tempStat == true)

{

fout << "Race car status updated to false" << endl;

}

else {

fout << "Race car status updated to true" << endl;

}

}

system("pause");

break;

case 16: //Open sunroof

if (Lambo.getSunroofOpen() || Lambo.getEngineOn() == false || Lambo.getAirOn())

{

if (Lambo.getSunroofOpen())

{

cout << "Invalid input: Cannot open sunroof if it is already open" << endl;

}

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot open sunroof if engine is off" << endl;

}

if (Lambo.getAirOn())

{

cout << "Invalid input: Cannot open sunroof if A/C is on" << endl;

}

}

else {

Lambo.setSunRoofStatus(true);

cout << "Sunroof is now open" << endl;

fout << "User opened the sunroof" << endl;

}

system("pause");

break;

case 17: //Close sunroof

if (Lambo.getSunroofOpen() == false || Lambo.getEngineOn() == false)

{

if (Lambo.getSunroofOpen() == false)

{

cout << "Invalid input: Cannot close sunroof if it is already closed" << endl;

}

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot close sunroof if engine is off" << endl;

}

}

else {

Lambo.setSunRoofStatus(false);

cout << "Sunroof is now closed" << endl;

fout << "User closed the sunroof" << endl;

}

system("pause");

break;

case 18: //Turn right

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (Lambo.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else if (Lambo.getSpeed() > 10)

{

cout << "Invalid input: Cannot turn if speed is greater than 10" << endl;

}

else {

Lambo.turnRight(turnsTripOne);

cout << "You have turned right" << endl;

fout << "User turned right" << endl;

Lambo.addMileage(10);

}

system("pause");

break;

case 19: //Turn left

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (Lambo.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else if (Lambo.getSpeed() > 10)

{

cout << "Invalid input: Cannot turn if speed is greater than 10" << endl;

}

else {

Lambo.turnLeft(turnsTripOne);

cout << "You have turned left" << endl;

fout << "User turned left" << endl;

Lambo.addMileage(10);

}

system("pause");

break;

case 20: //Accelerate

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot accelerate if engine is off" << endl;

}

else if (Lambo.getSpeed() == 45)

{

cout << "Invalid input: Cannot accelerate to more than 45 mph" << endl;

}

else {

Lambo.accelerate();

cout << "Speed is now increased to " << Lambo.getSpeed() << endl;

fout << "User accelerated vehicle to " << Lambo.getSpeed() << endl;

Lambo.addMileage(10);

}

system("pause");

break;

case 21: //Decelerate

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot decelerate if engine is off" << endl;

}

else if (Lambo.getSpeed() == 0)

{

cout << "Invalid input: Cannot decelerate to less than 0 mph" << endl;

}

else {

Lambo.decelerate();

cout << "Speed is now decreased to " << Lambo.getSpeed() << endl;

fout << "User decelerated vehicle to " << Lambo.getSpeed() << endl;

Lambo.addMileage(10);

}

system("pause");

break;

case 22: //Return to last location

for (int i = 0; i < turnsTripOne.size(); i++)

{

if (turnsTripOne.at(i) == 1)

{

turnsTripOne[i] = 3;

}

else {

turnsTripOne[i] = 1;

}

}

reverse(turnsTripOne.begin(), turnsTripOne.end());

for (int i = 0; i < turnsTripOne.size(); i++)

{

if (turnsTripOne.at(i) == 1)

{

cout << "Turn left..." << endl;

fout << "User turned left" << endl;

}

else {

cout << "Turn right..." << endl;

fout << "User turned right" << endl;

}

}

location[location.size() - 1] = NULL; //Deletes last location

cout << "You are at home now" << endl;

fout << "User has gone back to last location" << endl;

system("pause");

break;

case 23: //Reached destination

if (Lambo.getEngineOn())

{

cout << "Invalid input: Cannot get out of vehicle if engine is on" << endl;

}

else {

cout << "What location is this? Type 2 for Airport. Type 3 for Marina: ";

cin >> userLocation;

if (userLocation == 2)

{

location.push\_back(2);

}

else if (userLocation == 3)

{

location.push\_back(3);

}

else {

cout << "Invalid input: Only values acceptable are 2 or 3" << endl;

}

fout << "User has ended this trip" << endl;

Lambo.addMileage(20);

}

system("pause");

break;

case 0: //Quit

cout << "Sorry to see you go" << endl;

endOfMenu = true;

system("pause");

break;

};

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

if (userInput == 0)

{

break;

}

} while (userInput != 23);

}

else if (userMenu == 2) //If user selected Truck

{

vehicle.push\_back(4);

fout << "User is driving a Truck now" << endl;

do {

system("CLS");

cout << "You are driving Truck" << endl;

printTruckMenu();

cin >> userInput;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

switch (userInput) {

case 1: //Set age of Truck

cout << "What is the age of truck now?: ";

cin >> userSelection;

if (userSelection < TankTruck.getAge())

{

cout << "Invalid input. Age cannot be less than last one" << endl;

system("pause");

}

else if (userSelection == TankTruck.getAge())

{

cout << "The age you entered is the same age stored earlier" << endl;

system("pause");

}

else if (userSelection < 0)

{

cout << "Invalid input: Price cannot be negative" << endl;

system("Pause");

}

else {

TankTruck.setAge(userSelection);

cout << "Age stored successfully" << endl;

fout << "Age is stored as " << TankTruck.getAge() << endl;

system("pause");

}

break;

case 2: //Set price of Truck

cout << "What is the price of your truck now?: ";

cin >> userSelection;

if (userSelection > TankTruck.getPrice())

{

cout << "Invalid input. Price cannot be more than last one" << endl;

system("pause");

}

else if (userSelection == TankTruck.getPrice())

{

cout << "The price you enteres it the same price stored earlier" << endl;

system("pause");

}

else {

TankTruck.setPrice(userSelection);

cout << "Price stored successfully" << endl;

fout << "Price is stored as " << TankTruck.getPrice() << endl;

system("pause");

}

break;

case 3: //Check balance of user

if (vehicle.at(0) == 1)

{

cout << "Your balance is " << Lambo.getBalance() << endl;

}

else if (vehicle.at(0) == 4)

{

cout << "Your balance is " << TankTruck.getBalance() << endl;

}

system("pause");

break;

case 4: //Turn engine on

if (TankTruck.getEngineOn())

{

cout << "Invalid input: Cannot turn engine on while it is already on" << endl;

}

else {

TankTruck.turnEngineOn();

cout << "Engine is now on" << endl;

fout << "User turned on the engine" << endl;

}

system("pause");

break;

case 5: //Turn engine off

if (TankTruck.getEngineOn() == false || TankTruck.getSpeed() != 0 || TankTruck.getAirOn() == true || TankTruck.getLightsOn())

{

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn engine off while it is already off" << endl;

}

if (TankTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot turn engine off while vehicle is moving" << endl;

}

if (TankTruck.getAirOn() == true)

{

cout << "Invalid input: Cannot turn engine off while A/C is on" << endl;

}

if (TankTruck.getLightsOn())

{

cout << "Invalid input: Cannot turn engine off while lights are on" << endl;

}

}

else {

TankTruck.turnEngineOff();

cout << "Engine is now off" << endl;

fout << "User turned off the engine" << endl;

}

system("pause");

break;

case 6: //Turn lights on

if (TankTruck.getLightsOn() || TankTruck.getEngineOn() == false)

{

if (TankTruck.getLightsOn())

{

cout << "Invalid input: Cannot turn lights on if they are already on" << endl;

}

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn lights on if engine is off" << endl;

}

}

else {

TankTruck.turnLightsOn();

cout << "Lights are now on" << endl;

fout << "User turned on the lights" << endl;

}

system("pause");

break;

case 7: //Turn lights off

if (TankTruck.getEngineOn() == false || TankTruck.getLightsOn() == false)

{

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn turn lights off while engine is off" << endl;

}

if (TankTruck.getLightsOn() == false)

{

cout << "Invalid input: Cannot turn lights off while they are already off" << endl;

}

}

else {

TankTruck.turnLightsOff();

cout << "Lights are now off" << endl;

fout << "User turned off the lights" << endl;

}

system("pause");

break;

case 8: //Add Passengers

if (TankTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot add passengers while vehicle is moving" << endl;

}

else {

cout << "How many passengers do you want to add? ";

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot add 0 passengers" << endl;

}

else if (TankTruck.getNumPass() + numPass > 1)

{

cout << "Invlalid input: Cannot add passengers to have total of more than 1" << endl;

}

else {

TankTruck.addPass(numPass);

cout << "Passengers added successfully" << endl;

fout << "User added " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 9: //Kick Passengers

if (TankTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot kick passengers while vehicle is moving\nThey will die" << endl;

}

else {

cout << "How many passengers do you want to kick out" << endl;

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot kick 0 passengers" << endl;

}

else if (numPass > TankTruck.getNumPass())

{

cout << "Invalid input: Cannot kick more passengers than there are" << endl;

}

else {

TankTruck.kickPass(numPass);

cout << "Passengers kicked out successfully" << endl;

fout << "User has kicked " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 10: //Check mileage

cout << "You car's mileage is: " << TankTruck.getMileage() << " miles" << endl;

system("pause");

break;

case 11: //Set coefficient of friction

cout << "What is the current coefficient of friction" << endl;

cin >> coFriction;

if (coFriction < 0)

{

cout << "Invalid input: Cannot have negative coefficient of friction" << endl;

}

else if (coFriction > 1)

{

cout << "Invalid input: Cannot have coefficient of friction more than 1" << endl;

}

else {

TankTruck.setCoFriction(coFriction);

cout << "Coefficient is updated successfully" << endl;

fout << "Coefficient is updated to " << coFriction << endl;

}

system("pause");

break;

case 12: //Check friction

cout << "Current friction on your car is: " << TankTruck.getFriction();

system("pause");

break;

case 13: //Turn air on

if (TankTruck.getAirOn() || TankTruck.getEngineOn() == false)

{

if (TankTruck.getAirOn())

{

cout << "Invalid input: Cannot turn A/C on while it is already on" << endl;

}

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn air on if engine is off" << endl;

}

}

else {

TankTruck.turnAirOn();

cout << "Air is now on" << endl;

fout << "User turned on the air" << endl;

}

system("pause");

break;

case 14: //Turn air off

if (TankTruck.getAirOn() == false)

{

cout << "Invalid input: Cannot turn air off if it is already off" << endl;

}

else {

TankTruck.turnAirOff();

cout << "Air is now off" << endl;

fout << "User turned off the air" << endl;

}

system("pause");

break;

case 15: //Change diesel type status

cout << "Type 1 if it IS a diesel type and type 0 if it IS NOT a diesel type: ";

cin >> tempStat;

if (tempStat < 0 || tempStat > 1)

{

cout << "Invalid input: Only accepted values are 0 and 1" << endl;

}

else {

TankTruck.setDieselTypeStatus(tempStat);

cout << "Diesel type status updated" << endl;

if (tempStat)

{

fout << "Diesel type status is updated to true" << endl;

}

else

{

fout << "Diesel type status is updated to false" << endl;

}

}

system("pause");

break;

case 16: //Load cargo

cout << "How much do you want to load?: " << endl;

cin >> userCapacity;

if (userCapacity < 0 || userCapacity == 0 || userCapacity > 100 || TankTruck.getCapacity() + userCapacity > 100 || TankTruck.getSpeed() != 0)

{

if (100 - TankTruck.getCapacity() < userCapacity)

{

cout << "Invalid input: Cannot add to end up with more than 100 cargo" << endl;

}

if (userCapacity < 0)

{

cout << "Invalid input: Cannot add 0 cargo" << endl;

}

if (userCapacity == 0)

{

cout << "Invalid input: Cannot add 0 cargo" << endl;

}

if (TankTruck.getCapacity() > 100)

{

cout << "Invalid input: Cannot add more than 100 cargo" << endl;

}

if (TankTruck.getCapacity() + userCapacity > 100)

{

cout << "Invalid input: Cannot add cargo to have total of more than 100 cargo" << endl;

}

if (TankTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot add cargo while vehicle is moving" << endl;

}

}

else {

TankTruck.loadCargo(userCapacity);

cout << "Cargo loaded successfully" << endl;

fout << "User added " << userCapacity << " cargo" << endl;

}

system("pause");

break;

case 17: //Unload Cargo

cout << "How much do you want to unload?: " << endl;

cin >> userCapacity;

if (userCapacity > TankTruck.getCapacity() || userCapacity < 0 || userCapacity == 0 || userCapacity > 100 || TankTruck.getSpeed() != 0)

{

if (userCapacity > TankTruck.getCapacity())

{

cout << "Invalid input: Cannot unload to end up with negative cargo" << endl;

}

if (userCapacity < 0)

{

cout << "Invalid input: Cannot have a negative cargo" << endl;

}

if (userCapacity == 0)

{

cout << "Invalid input: Cannot unload 0 cargo" << endl;

}

if (userCapacity > 100)

{

cout << "Invalid input: Cannot unload more than 100 cargo" << endl;

}

if (TankTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot unload cargo while vehicle is moving" << endl;

}

}

else {

TankTruck.loadCargo(userCapacity);

cout << "Cargo loaded successfully" << endl;

fout << "User took out " << userCapacity << " cargo" << endl;

}

system("pause");

break;

case 18: //Check cargo

cout << "Current cargo is " << TankTruck.getCapacity() << endl;

system("pause");

break;

case 19: //Turn right

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (TankTruck.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else if (TankTruck.getSpeed() > 10)

{

cout << "Invalid input: Cannot turn if speed is greater than 10" << endl;

}

else {

TankTruck.turnRight(turnsTripOne);

cout << "You have turned right" << endl;

fout << "User turned right" << endl;

TankTruck.addMileage(10);

}

system("pause");

break;

case 20: //Turn left

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (TankTruck.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else if (TankTruck.getSpeed() > 10)

{

cout << "Invalid input: Cannot turn if speed is greater than 10" << endl;

}

else {

TankTruck.turnLeft(turnsTripOne);

cout << "You have turned left" << endl;

fout << "User turned left" << endl;

TankTruck.addMileage(10);

}

system("pause");

break;

case 21: //Accelerate

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot accelerate if engine is off" << endl;

}

else if (TankTruck.getSpeed() == 45)

{

cout << "Invalid input: Cannot accelerate to more than 45 mph" << endl;

}

else {

TankTruck.accelerate();

cout << "Speed is now increased to " << TankTruck.getSpeed() << endl;

fout << "User accelerated vehicle to " << TankTruck.getSpeed() << endl;

TankTruck.addMileage(10);

}

system("pause");

break;

case 22: //Decelerate

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot decelerate if engine is off" << endl;

}

else if (TankTruck.getSpeed() == 0)

{

cout << "Invalid input: Cannot decelerate to less than 0 mph" << endl;

}

else {

TankTruck.decelerate();

cout << "Speed is now decreased to " << TankTruck.getSpeed() << endl;

fout << "User decelerated vehicle to " << TankTruck.getSpeed() << endl;

TankTruck.addMileage(10);

}

system("pause");

break;

case 23: //Return to last location

for (int i = 0; i < turnsTripOne.size(); i++)

{

if (turnsTripOne.at(i) == 1)

{

turnsTripOne[i] = 3;

}

else {

turnsTripOne[i] = 1;

}

}

reverse(turnsTripOne.begin(), turnsTripOne.end());

for (int i = 0; i < turnsTripOne.size(); i++)

{

if (turnsTripOne.at(i) == 1)

{

cout << "Turn left..." << endl;

fout << "User turned left" << endl;

}

else {

cout << "Turn right..." << endl;

fout << "User turned right" << endl;

}

}

location[location.size() - 1] = NULL; //Deletes last location

cout << "You are at home now" << endl;

fout << "User has returned to last location" << endl;

TankTruck.addMileage(20);

system("pause");

break;

case 24: //Reached destination

if (TankTruck.getEngineOn())

{

cout << "Invalid input: Cannot get out of vehicle if engine is on" << endl;

}

else {

cout << "What location is this? Type 2 for Airport. Type 3 for Marina: ";

cin >> userLocation;

if (userLocation == 2)

{

location.push\_back(2);

}

else if (userLocation == 3)

{

location.push\_back(3);

}

else {

cout << "Invalid input: Only values acceptable are 2 or 3" << endl;

}

fout << "User has reached the destination" << endl;

TankTruck.addMileage(20);

}

system("Pause");

break;

case 0: //Quit

cout << "Sorry to see you go" << endl;

endOfMenu = true;

system("pause");

break;

};

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

if (userInput == 0)

{

break;

}

} while (userInput != 24);

}

else {

cout << "Invalid input. Neither Car or Truck was selected" << endl;

}

}

//................... . . . . . . . . . . . . . . . . . . . . . . . . .

else if (location.back() == 2) //If user is at Airport

{

cout << "You are at Airport right now" << endl;

cout << "1. Take the Car" << endl;

cout << "2. Take the Truck" << endl;

cout << "3. Take the Plane" << endl;

cin >> userMenu;

if (userMenu == 1) //If user selected Car

{

vehicle.push\_back(2);

fout << "User is driving a car" << endl;

do {

system("CLS");

cout << "You are driving Car" << endl;

printCarMenu();

cin >> userInput;

if (location.at(location.size() - 2) == 1) //Car

{

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

switch (userInput) {

case 1: //Set age of car

cout << "What is the age of car now?: ";

cin >> userSelection;

if (userSelection < Lambo.getAge())

{

cout << "Invalid input. Age cannot be less than last one" << endl;

system("pause");

}

else if (userSelection == Lambo.getAge())

{

cout << "The age you entered is the same age stored earlier" << endl;

system("pause");

}

else {

Lambo.setAge(userSelection);

cout << "Age stored successfully" << endl;

fout << "Age is stored as " << userSelection << endl;

system("pause");

}

break;

case 2: //Set price of car

cout << "What is the price of your car now?: ";

cin >> userSelection;

if (userSelection > Lambo.getPrice())

{

cout << "Invalid input. Price cannot be more than last one" << endl;

system("pause");

}

else if (userSelection == Lambo.getPrice())

{

cout << "The price you enteres it the same price stored earlier" << endl;

system("pause");

}

else if (userSelection < 0)

{

cout << "Invalid input: Price cannot be negative" << endl;

system("Pause");

}

else {

Lambo.setPrice(userSelection);

cout << "Price stored successfully" << endl;

fout << "Price is stored as " << userSelection << endl;

system("pause");

}

break;

case 3: //Check balance of user

if (vehicle.at(0) == 1)

{

cout << "Your balance is " << Lambo.getBalance() << endl;

}

else if (vehicle.at(0) == 4)

{

cout << "Your balance is " << TankTruck.getBalance() << endl;

}

system("pause");

break;

case 4: //Turn engine on

if (Lambo.getEngineOn())

{

cout << "Invalid input: Cannot turn engine on while it is already on" << endl;

}

else {

Lambo.turnEngineOn();

cout << "Engine is now on" << endl;

fout << "User turned on the engine" << endl;

}

system("pause");

break;

case 5: //Turn engine off

if (Lambo.getEngineOn() == false || Lambo.getSpeed() != 0 || Lambo.getAirOn() == true || Lambo.getSunroofOpen() == true || Lambo.getLightsOn())

{

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn engine off while it is already off" << endl;

}

if (Lambo.getSpeed() != 0)

{

cout << "Invalid input: Cannot turn engine off while vehicle is moving" << endl;

}

if (Lambo.getAirOn() == true)

{

cout << "Invalid input: Cannot turn engine off while A/C is on" << endl;

}

if (Lambo.getSunroofOpen() == true)

{

cout << "Invalid input: Cannot turn engine off while sunroof is open" << endl;

}

if (Lambo.getLightsOn())

{

cout << "Invalid input: Cannot turn engine off while lights are on" << endl;

}

}

else {

Lambo.turnEngineOff();

cout << "Engine is now off" << endl;

fout << "User turned off the engine" << endl;

}

system("pause");

break;

case 6: //Turn lights on

if (Lambo.getLightsOn() || Lambo.getEngineOn() == false)

{

if (Lambo.getLightsOn())

{

cout << "Invalid input: Cannot turn lights on if they are already on" << endl;

}

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn lights on if engine is off" << endl;

}

}

else {

Lambo.turnLightsOn();

cout << "Lights are now on" << endl;

fout << "User turned on the ligths" << endl;

}

system("pause");

break;

case 7: //Turn lights off

if (Lambo.getEngineOn() == false || Lambo.getLightsOn() == false)

{

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn turn lights off while engine is off" << endl;

}

if (Lambo.getLightsOn() == false)

{

cout << "Invalid input: Cannot turn lights off while they are already off" << endl;

}

}

else {

Lambo.turnLightsOff();

cout << "Lights are now off" << endl;

fout << "User turned off the lights" << endl;

}

system("pause");

break;

case 8: //Add Passengers

if (Lambo.getSpeed() != 0)

{

cout << "Invalid input: Cannot add passengers while vehicle is moving" << endl;

}

else {

cout << "How many passengers do you want to add? ";

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot add 0 passengers" << endl;

}

else if (Lambo.getNumPass() + numPass > 4)

{

cout << "Invlalid input: Cannot add passengers to have total of more than 4" << endl;

}

else {

Lambo.addPass(numPass);

cout << "Passengers added successfully" << endl;

fout << "User added " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 9: //Kick Passengers

if (Lambo.getSpeed() != 0)

{

cout << "Invalid input: Cannot kick passengers while vehicle is moving\nThey will die" << endl;

}

else {

cout << "How many passengers do you want to kick out" << endl;

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot kick 0 passengers" << endl;

}

else if (numPass > Lambo.getNumPass())

{

cout << "Invalid input: Cannot kick more passengers than there are" << endl;

}

else {

Lambo.kickPass(numPass);

cout << "Passengers kicked out successfully" << endl;

fout << "User kicked " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 10: //Check mileage

cout << "You car's mileage is: " << Lambo.getMileage() << " miles" << endl;

system("pause");

break;

case 11: //Set coefficient of friction

cout << "What is the current coefficient of friction" << endl;

cin >> coFriction;

if (coFriction < 0)

{

cout << "Invalid input: Cannot have negative coefficient of friction" << endl;

}

else if (coFriction > 1)

{

cout << "Invalid input: Cannot have coefficient of friction more than 1" << endl;

}

else {

Lambo.setCoFriction(coFriction);

cout << "Coefficient is updated successfully" << endl;

fout << "Coefficient is updated to " << coFriction << endl;

}

system("pause");

break;

case 12: //Check friction

cout << "Current friction on your car is: " << Lambo.getFriction();

system("pause");

break;

case 13: //Turn air on

if (Lambo.getAirOn() || Lambo.getEngineOn() == false || Lambo.getSunroofOpen())

{

if (Lambo.getAirOn())

{

cout << "Invalid input: Cannot turn A/C on while it is already on" << endl;

}

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn air on if engine is off" << endl;

}

if (Lambo.getSunroofOpen())

{

cout << "Invalid input: Cannot turn air on if sunroof is open" << endl;

}

}

else {

Lambo.turnAirOn();

cout << "Air is now on" << endl;

fout << "User turned on the air" << endl;

}

system("pause");

break;

case 14: //Turn air off

if (Lambo.getAirOn() == false)

{

cout << "Invalid input: Cannot turn air off if it is already off" << endl;

}

else {

Lambo.turnAirOff();

cout << "Air is now off" << endl;

fout << "User turned off the air" << endl;

}

system("pause");

break;

case 15: //Change race car status

cout << "Type 1 if it IS a race car and type 0 if it IS NOT a race car: ";

cin >> tempStat;

if (tempStat < 0 || tempStat > 1)

{

cout << "Invalid input: Only accepted values are 0 and 1" << endl;

}

else {

Lambo.setRaceCarStatus(tempStat);

cout << "Race Car status updated" << endl;

if (tempStat)

{

fout << "Race Car status updated to true" << endl;

}

else {

fout << "Race car status updated to false" << endl;

}

}

system("pause");

break;

case 16: //Open sunroof

if (Lambo.getSunroofOpen() || Lambo.getEngineOn() == false || Lambo.getAirOn())

{

if (Lambo.getSunroofOpen())

{

cout << "Invalid input: Cannot open sunroof if it is already open" << endl;

}

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot open sunroof if engine is off" << endl;

}

if (Lambo.getAirOn())

{

cout << "Invalid input: Cannot open sunroof if A/C is on" << endl;

}

}

else {

Lambo.setSunRoofStatus(true);

cout << "Sunroof is now open" << endl;

fout << "User opened sunroof" << endl;

}

system("pause");

break;

case 17: //Close sunroof

if (Lambo.getSunroofOpen() == false || Lambo.getEngineOn() == false)

{

if (Lambo.getSunroofOpen() == false)

{

cout << "Invalid input: Cannot close sunroof if it is already closed" << endl;

}

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot close sunroof if engine is off" << endl;

}

}

else {

Lambo.setSunRoofStatus(false);

cout << "Sunroof is now closed" << endl;

fout << "User closed sunroof" << endl;

}

system("pause");

break;

case 18: //Turn right

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (Lambo.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else if (Lambo.getSpeed() > 10)

{

cout << "Invalid input: Cannot turn if speed is greater than 10" << endl;

}

else {

Lambo.turnRight(turnsTripTwo);

cout << "You have turned right" << endl;

fout << "User turned right" << endl;

Lambo.addMileage(10);

}

system("pause");

break;

case 19: //Turn left

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (Lambo.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else if (Lambo.getSpeed() > 10)

{

cout << "Invalid input: Cannot turn if speed is greater than 10" << endl;

}

else {

Lambo.turnLeft(turnsTripTwo);

cout << "You have turned left" << endl;

fout << "User turned left" << endl;

Lambo.addMileage(10);

}

system("pause");

break;

case 20: //Accelerate

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot accelerate if engine is off" << endl;

}

else if (Lambo.getSpeed() == 45)

{

cout << "Invalid input: Cannot accelerate to more than 45 mph" << endl;

}

else {

Lambo.accelerate();

cout << "Speed is now increased to " << Lambo.getSpeed() << endl;

fout << "User accelerated vehicle to " << Lambo.getSpeed() << endl;

Lambo.addMileage(10);

}

system("pause");

break;

case 21: //Decelerate

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot decelerate if engine is off" << endl;

}

else if (Lambo.getSpeed() == 0)

{

cout << "Invalid input: Cannot decelerate to less than 0 mph" << endl;

}

else {

Lambo.decelerate();

cout << "Speed is now decreased to " << Lambo.getSpeed() << endl;

fout << "User decelerated vehicle to " << Lambo.getSpeed() << endl;

Lambo.addMileage(10);

}

system("pause");

break;

case 22: //Return to last location

for (int i = 0; i < turnsTripTwo.size(); i++)

{

if (turnsTripOne.at(i) == 1)

{

turnsTripOne[i] = 3;

}

else {

turnsTripOne[i] = 1;

}

}

reverse(turnsTripTwo.begin(), turnsTripTwo.end());

for (int i = 0; i < turnsTripTwo.size(); i++)

{

if (turnsTripTwo.at(i) == 1)

{

cout << "Turn left..." << endl;

fout << "User turned left" << endl;

Lambo.addMileage(10);

}

else {

cout << "Turn right..." << endl;

fout << "User turned right" << endl;

Lambo.addMileage(10);

}

}

if (location.back() == 1)

{

cout << "You are now home" << endl;

}

else if (location.back() == 2)

{

cout << "You are now at airport" << endl;

}

else if (location.back() == 3)

{

cout << "You are now at the Marina" << endl;

}

location[location.size() - 1] = NULL; //Deletes last location

vehicle[vehicle.size() - 1] = NULL; //Deletes last vehicle

fout << "User returned to last location" << endl;

system("pause");

break;

case 23: //Reached destination

if (Lambo.getEngineOn())

{

cout << "Invalid input: Cannot get out of vehicle if engine is on" << endl;

}

else {

cout << "What location is this? Type 2 for Airport. Type 3 for Marina: ";

cin >> userLocation;

if (userLocation == 2)

{

location.push\_back(2);

}

else if (userLocation == 3)

{

location.push\_back(3);

}

else {

cout << "Invalid input: Only values acceptable are 2 or 3" << endl;

}

fout << "Trip is ended" << endl;

Lambo.addMileage(20);

}

system("pause");

break;

case 0: //Quit

cout << "Sorry to see you go" << endl;

endOfMenu = true;

system("pause");

break;

};

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

else { //Taxi

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

switch (userInput) {

case 1: //Set age of car

cout << "What is the age of car now?: ";

cin >> userSelection;

if (userSelection < Taxi.getAge())

{

cout << "Invalid input. Age cannot be less than last one" << endl;

system("pause");

}

else if (userSelection == Taxi.getAge())

{

cout << "The age you entered is the same age stored earlier" << endl;

system("pause");

}

else {

Taxi.setAge(userSelection);

cout << "Age stored successfully" << endl;

fout << "Age is stored as " << Taxi.getAge();

system("pause");

}

break;

case 2: //Set price of car

cout << "What is the price of your car now?: ";

cin >> userSelection;

if (userSelection > Taxi.getPrice())

{

cout << "Invalid input. Price cannot be more than last one" << endl;

system("pause");

}

else if (userSelection == Taxi.getPrice())

{

cout << "The price you enteres it the same price stored earlier" << endl;

system("pause");

}

else if (userSelection < 0)

{

cout << "Invalid input: Price cannot be negative" << endl;

system("Pause");

}

else {

Taxi.setPrice(userSelection);

cout << "Price stored successfully" << endl;

fout << "Price is stored as " << userSelection << endl;

system("pause");

}

break;

case 3: //Check balance of user

if (vehicle.at(0) == 1)

{

cout << "Your balance is " << Lambo.getBalance() << endl;

}

else if (vehicle.at(0) == 4)

{

cout << "Your balance is " << TankTruck.getBalance() << endl;

}

system("pause");

break;

case 4: //Turn engine on

if (Taxi.getEngineOn())

{

cout << "Invalid input: Cannot turn engine on while it is already on" << endl;

}

else {

Taxi.turnEngineOn();

cout << "Engine is now on" << endl;

fout << "User turned the engine on" << endl;

}

system("pause");

break;

case 5: //Turn engine off

if (Taxi.getEngineOn() == false || Taxi.getSpeed() != 0 || Taxi.getAirOn() == true || Taxi.getSunroofOpen() == true || Taxi.getLightsOn())

{

if (Taxi.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn engine off while it is already off" << endl;

}

if (Taxi.getSpeed() != 0)

{

cout << "Invalid input: Cannot turn engine off while vehicle is moving" << endl;

}

if (Taxi.getAirOn() == true)

{

cout << "Invalid input: Cannot turn engine off while A/C is on" << endl;

}

if (Taxi.getSunroofOpen() == true)

{

cout << "Invalid input: Cannot turn engine off while sunroof is open" << endl;

}

if (Taxi.getLightsOn())

{

cout << "Invalid input: Cannot turn engine off while lights are on" << endl;

}

}

else {

Taxi.turnEngineOff();

cout << "Engine is now off" << endl;

fout << "User turned the engine off" << endl;

}

system("pause");

break;

case 6: //Turn lights on

if (Taxi.getLightsOn() || Taxi.getEngineOn() == false)

{

if (Taxi.getLightsOn())

{

cout << "Invalid input: Cannot turn lights on if they are already on" << endl;

}

if (Taxi.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn lights on if engine is off" << endl;

}

}

else {

Taxi.turnLightsOn();

cout << "Lights are now on" << endl;

fout << "User turned the lights on" << endl;

}

system("pause");

break;

case 7: //Turn lights off

if (Taxi.getEngineOn() == false || Taxi.getLightsOn() == false)

{

if (Taxi.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn turn lights off while engine is off" << endl;

}

if (Taxi.getLightsOn() == false)

{

cout << "Invalid input: Cannot turn lights off while they are already off" << endl;

}

}

else {

Taxi.turnLightsOff();

cout << "Lights are now off" << endl;

fout << "User turned off the lights" << endl;

}

system("pause");

break;

case 8: //Add Passengers

if (Taxi.getSpeed() != 0)

{

cout << "Invalid input: Cannot add passengers while vehicle is moving" << endl;

}

else {

cout << "How many passengers do you want to add? ";

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot add 0 passengers" << endl;

}

else if (Taxi.getNumPass() + numPass > 4)

{

cout << "Invlalid input: Cannot add passengers to have total of more than 4" << endl;

}

else {

Taxi.addPass(numPass);

cout << "Passengers added successfully" << endl;

fout << "User added " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 9: //Kick Passengers

if (Taxi.getSpeed() != 0)

{

cout << "Invalid input: Cannot kick passengers while vehicle is moving\nThey will die" << endl;

}

else {

cout << "How many passengers do you want to kick out" << endl;

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot kick 0 passengers" << endl;

}

else if (numPass > Taxi.getNumPass())

{

cout << "Invalid input: Cannot kick more passengers than there are" << endl;

}

else {

Taxi.kickPass(numPass);

cout << "Passengers kicked out successfully" << endl;

fout << "User kicked out " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 10: //Check mileage

cout << "You car's mileage is: " << Taxi.getMileage() << " miles" << endl;

system("pause");

break;

case 11: //Set coefficient of friction

cout << "What is the current coefficient of friction" << endl;

cin >> coFriction;

if (coFriction < 0)

{

cout << "Invalid input: Cannot have negative coefficient of friction" << endl;

}

else if (coFriction > 1)

{

cout << "Invalid input: Cannot have coefficient of friction more than 1" << endl;

}

else {

Taxi.setCoFriction(coFriction);

cout << "Coefficient is updated successfully" << endl;

fout << "Coefficient is updated to " << coFriction << endl;

}

system("pause");

break;

case 12: //Check friction

cout << "Current friction on your car is: " << Taxi.getFriction();

system("pause");

break;

case 13: //Turn air on

if (Taxi.getAirOn() || Taxi.getEngineOn() == false || Taxi.getSunroofOpen())

{

if (Taxi.getAirOn())

{

cout << "Invalid input: Cannot turn A/C on while it is already on" << endl;

}

if (Taxi.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn air on if engine is off" << endl;

}

if (Taxi.getSunroofOpen())

{

cout << "Invalid input: Cannot turn air on if sunroof is open" << endl;

}

}

else {

Taxi.turnAirOn();

cout << "Air is now on" << endl;

fout << "User turned on the air" << endl;

}

system("pause");

break;

case 14: //Turn air off

if (Taxi.getAirOn() == false)

{

cout << "Invalid input: Cannot turn air off if it is already off" << endl;

}

else {

Taxi.turnAirOff();

cout << "Air is now off" << endl;

fout << "User turned off the air" << endl;

}

system("pause");

break;

case 15: //Change race car status

cout << "Type 1 if it IS a race car and type 0 if it IS NOT a race car: ";

cin >> tempStat;

if (tempStat < 0 || tempStat > 1)

{

cout << "Invalid input: Only accepted values are 0 and 1" << endl;

}

else {

Taxi.setRaceCarStatus(tempStat);

cout << "Race Car status updated" << endl;

if (tempStat)

{

fout << "Race car status updated to true" << endl;

}

else {

fout << "Race car status updated to false" << endl;

}

}

system("pause");

break;

case 16: //Open sunroof

if (Taxi.getSunroofOpen() || Taxi.getEngineOn() == false || Taxi.getAirOn())

{

if (Taxi.getSunroofOpen())

{

cout << "Invalid input: Cannot open sunroof if it is already open" << endl;

}

if (Taxi.getEngineOn() == false)

{

cout << "Invalid input: Cannot open sunroof if engine is off" << endl;

}

if (Taxi.getAirOn())

{

cout << "Invalid input: Cannot open sunroof if A/C is on" << endl;

}

}

else {

Taxi.setSunRoofStatus(true);

cout << "Sunroof is now open" << endl;

fout << "User opened sunroof" << endl;

}

system("pause");

break;

case 17: //Close sunroof

if (Taxi.getSunroofOpen() == false || Taxi.getEngineOn() == false)

{

if (Taxi.getSunroofOpen() == false)

{

cout << "Invalid input: Cannot close sunroof if it is already closed" << endl;

}

if (Taxi.getEngineOn() == false)

{

cout << "Invalid input: Cannot close sunroof if engine is off" << endl;

}

}

else {

Taxi.setSunRoofStatus(false);

cout << "Sunroof is now closed" << endl;

fout << "User closed sunroof" << endl;

}

system("pause");

break;

case 18: //Turn right

if (Taxi.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (Taxi.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else if (Taxi.getSpeed() > 10)

{

cout << "Invalid input: Cannot turn if speed is greater than 10" << endl;

}

else {

Taxi.turnRight(turnsTripTwo);

cout << "You have turned right" << endl;

fout << "User turned right" << endl;

Taxi.addMileage(10);

}

system("pause");

break;

case 19: //Turn left

if (Taxi.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (Taxi.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else if (Taxi.getSpeed() > 10)

{

cout << "Invalid input: Cannot turn if speed is greater than 10" << endl;

}

else {

Taxi.turnLeft(turnsTripTwo);

cout << "You have turned left" << endl;

fout << "User turned left" << endl;

Taxi.addMileage(10);

}

system("pause");

break;

case 20: //Accelerate

if (Taxi.getEngineOn() == false)

{

cout << "Invalid input: Cannot accelerate if engine is off" << endl;

}

else if (Taxi.getSpeed() == 45)

{

cout << "Invalid input: Cannot accelerate to more than 45 mph" << endl;

}

else {

Taxi.accelerate();

cout << "Speed is now increased to " << Taxi.getSpeed() << endl;

fout << "User accelerated vehicle to " << Taxi.getSpeed() << endl;

Taxi.addMileage(10);

}

system("pause");

break;

case 21: //Decelerate

if (Taxi.getEngineOn() == false)

{

cout << "Invalid input: Cannot decelerate if engine is off" << endl;

}

else if (Taxi.getSpeed() == 0)

{

cout << "Invalid input: Cannot decelerate to less than 0 mph" << endl;

}

else {

Taxi.decelerate();

cout << "Speed is now decreased to " << Taxi.getSpeed() << endl;

fout << "User decelerated vehicle to " << Taxi.getSpeed() << endl;

Taxi.addMileage(10);

}

system("pause");

break;

case 22: //Return to last location

for (int i = 0; i < turnsTripTwo.size(); i++)

{

if (turnsTripOne.at(i) == 1)

{

turnsTripOne[i] = 3;

}

else {

turnsTripOne[i] = 1;

}

}

reverse(turnsTripTwo.begin(), turnsTripTwo.end());

for (int i = 0; i < turnsTripTwo.size(); i++)

{

if (turnsTripTwo.at(i) == 1)

{

cout << "Turn left..." << endl;

fout << "User turned left" << endl;

Taxi.addMileage(10);

}

else {

cout << "Turn right..." << endl;

fout << "User turned right " << endl;

Taxi.addMileage(10);

}

}

location[location.size() - 1] = NULL; //Deletes last location

cout << "You are at home now" << endl;

fout << "User has returned to last location" << endl;

system("pause");

break;

case 23: //Reached destination

if (Taxi.getEngineOn())

{

cout << "Invalid input: Cannot get out of vehicle if engine is on" << endl;

}

else {

cout << "What location is this? Type 2 for Airport. Type 3 for Marina: ";

cin >> userLocation;

if (userLocation == 2)

{

location.push\_back(2);

}

else if (userLocation == 3)

{

location.push\_back(3);

}

else {

cout << "Invalid input: Only values acceptable are 2 or 3" << endl;

}

fout << "User has ended this trip" << endl;

Taxi.addMileage(20);

}

system("pause");

break;

case 0: //Quit

cout << "Sorry to see you go" << endl;

endOfMenu = true;

system("pause");

break;

};

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

if (userInput == 0)

{

break;

}

} while (userInput != 23);

}

else if (userMenu == 2) //If user selected Truck

{

vehicle.push\_back(3);

do {

system("CLS");

cout << "You are driving Truck" << endl;

printTruckMenu();

cin >> userInput;

if (location.at(location.size() - 2) == 1) //Tank Truck

{

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

switch (userInput) {

case 1: //Set age of Truck

cout << "What is the age of truck now?: ";

cin >> userSelection;

if (userSelection < TankTruck.getAge())

{

cout << "Invalid input. Age cannot be less than last one" << endl;

system("pause");

}

else if (userSelection == TankTruck.getAge())

{

cout << "The age you entered is the same age stored earlier" << endl;

system("pause");

}

else {

TankTruck.setAge(userSelection);

cout << "Age stored successfully" << endl;

fout << "Age is stored as " << userSelection << endl;

system("pause");

}

break;

case 2: //Set price of Truck

cout << "What is the price of your truck now?: ";

cin >> userSelection;

if (userSelection > TankTruck.getPrice())

{

cout << "Invalid input. Price cannot be more than last one" << endl;

system("pause");

}

else if (userSelection == TankTruck.getPrice())

{

cout << "The price you enteres it the same price stored earlier" << endl;

system("pause");

}

else if (userSelection < 0)

{

cout << "Invalid input: Price cannot be negative" << endl;

system("Pause");

}

else {

TankTruck.setPrice(userSelection);

cout << "Price stored successfully" << endl;

fout << "Price is stored as " << userSelection << endl;

system("pause");

}

break;

case 3: //Check balance of user

if (vehicle.at(0) == 1)

{

cout << "Your balance is " << Lambo.getBalance() << endl;

}

else if (vehicle.at(0) == 4)

{

cout << "Your balance is " << TankTruck.getBalance() << endl;

}

system("pause");

break;

case 4: //Turn engine on

if (TankTruck.getEngineOn())

{

cout << "Invalid input: Cannot turn engine on while it is already on" << endl;

}

else {

TankTruck.turnEngineOn();

cout << "Engine is now on" << endl;

fout << "User turned on the engine" << endl;

}

system("pause");

break;

case 5: //Turn engine off

if (TankTruck.getEngineOn() == false || TankTruck.getSpeed() != 0 || TankTruck.getAirOn() == true || TankTruck.getLightsOn())

{

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn engine off while it is already off" << endl;

}

if (TankTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot turn engine off while vehicle is moving" << endl;

}

if (TankTruck.getAirOn() == true)

{

cout << "Invalid input: Cannot turn engine off while A/C is on" << endl;

}

if (TankTruck.getLightsOn())

{

cout << "Invalid input: Cannot turn engine off while lights are on" << endl;

}

}

else {

TankTruck.turnEngineOff();

cout << "Engine is now off" << endl;

fout << "User turned off the engine" << endl;

}

system("pause");

break;

case 6: //Turn lights on

if (TankTruck.getLightsOn() || TankTruck.getEngineOn() == false)

{

if (TankTruck.getLightsOn())

{

cout << "Invalid input: Cannot turn lights on if they are already on" << endl;

}

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn lights on if engine is off" << endl;

}

}

else {

TankTruck.turnLightsOn();

cout << "Lights are now on" << endl;

fout << "User turned on the lights" << endl;

}

system("pause");

break;

case 7: //Turn lights off

if (TankTruck.getEngineOn() == false || TankTruck.getLightsOn() == false)

{

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn turn lights off while engine is off" << endl;

}

if (TankTruck.getLightsOn() == false)

{

cout << "Invalid input: Cannot turn lights off while they are already off" << endl;

}

}

else {

TankTruck.turnLightsOff();

cout << "Lights are now off" << endl;

fout << "User turned ooff the ligths" << endl;

}

system("pause");

break;

case 8: //Add Passengers

if (TankTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot add passengers while vehicle is moving" << endl;

}

else {

cout << "How many passengers do you want to add? ";

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot add 0 passengers" << endl;

}

else if (TankTruck.getNumPass() + numPass > 1)

{

cout << "Invlalid input: Cannot add passengers to have total of more than 1" << endl;

}

else {

TankTruck.addPass(numPass);

cout << "Passengers added successfully" << endl;

fout << "User added " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 9: //Kick Passengers

if (TankTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot kick passengers while vehicle is moving\nThey will die" << endl;

}

else {

cout << "How many passengers do you want to kick out" << endl;

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot kick 0 passengers" << endl;

}

else if (numPass > TankTruck.getNumPass())

{

cout << "Invalid input: Cannot kick more passengers than there are" << endl;

}

else {

TankTruck.kickPass(numPass);

cout << "Passengers kicked out successfully" << endl;

fout << "User kicked out " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 10: //Check mileage

cout << "You car's mileage is: " << TankTruck.getMileage() << " miles" << endl;

system("pause");

break;

case 11: //Set coefficient of friction

cout << "What is the current coefficient of friction" << endl;

cin >> coFriction;

if (coFriction < 0)

{

cout << "Invalid input: Cannot have negative coefficient of friction" << endl;

}

else if (coFriction > 1)

{

cout << "Invalid input: Cannot have coefficient of friction more than 1" << endl;

}

else {

TankTruck.setCoFriction(coFriction);

cout << "Coefficient is updated successfully" << endl;

fout << "Coefficient is updated to " << coFriction << endl;

}

system("pause");

break;

case 12: //Check friction

cout << "Current friction on your car is: " << TankTruck.getFriction();

system("pause");

break;

case 13: //Turn air on

if (TankTruck.getAirOn() || TankTruck.getEngineOn() == false)

{

if (TankTruck.getAirOn())

{

cout << "Invalid input: Cannot turn A/C on while it is already on" << endl;

}

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn air on if engine is off" << endl;

}

}

else {

TankTruck.turnAirOn();

cout << "Air is now on" << endl;

fout << "User turned on the air" << endl;

}

system("pause");

break;

case 14: //Turn air off

if (TankTruck.getAirOn() == false)

{

cout << "Invalid input: Cannot turn air off if it is already off" << endl;

}

else {

TankTruck.turnAirOff();

cout << "Air is now off" << endl;

fout << "User turned off the air" << endl;

}

system("pause");

break;

case 15: //Change diesel type status

cout << "Type 1 if it IS a diesel type and type 0 if it IS NOT a diesel type: ";

cin >> tempStat;

if (tempStat < 0 || tempStat > 1)

{

cout << "Invalid input: Only accepted values are 0 and 1" << endl;

}

else {

TankTruck.setDieselTypeStatus(tempStat);

cout << "Diesel type status updated" << endl;

if (tempStat)

{

fout << "Diesel type status updated to true" << endl;

}

else {

fout << "Diesel type status updated to false" << endl;

}

}

system("pause");

break;

case 16: //Load cargo

cout << "How much do you want to load?: " << endl;

cin >> userCapacity;

if (userCapacity < 0 || userCapacity == 0 || userCapacity > 100 || TankTruck.getCapacity() + userCapacity > 100 || TankTruck.getSpeed() != 0)

{

if (100 - TankTruck.getCapacity() < userCapacity)

{

cout << "Invalid input: Cannot add to end up with more than 100 cargo" << endl;

}

if (userCapacity < 0)

{

cout << "Invalid input: Cannot add 0 cargo" << endl;

}

if (userCapacity == 0)

{

cout << "Invalid input: Cannot add 0 cargo" << endl;

}

if (TankTruck.getCapacity() > 100)

{

cout << "Invalid input: Cannot add more than 100 cargo" << endl;

}

if (TankTruck.getCapacity() + userCapacity > 100)

{

cout << "Invalid input: Cannot add cargo to have total of more than 100 cargo" << endl;

}

if (TankTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot add cargo while vehicle is moving" << endl;

}

}

else {

TankTruck.loadCargo(userCapacity);

cout << "Cargo loaded successfully" << endl;

fout << "User added " << userCapacity << " cargo" << endl;

}

system("pause");

break;

case 17: //Unload Cargo

cout << "How much do you want to unload?: " << endl;

cin >> userCapacity;

if (userCapacity > TankTruck.getCapacity() || userCapacity < 0 || userCapacity == 0 || userCapacity > 100 || TankTruck.getSpeed() != 0)

{

if (userCapacity > TankTruck.getCapacity())

{

cout << "Invalid input: Cannot unload to end up with negative cargo" << endl;

}

if (userCapacity < 0)

{

cout << "Invalid input: Cannot have a negative cargo" << endl;

}

if (userCapacity == 0)

{

cout << "Invalid input: Cannot unload 0 cargo" << endl;

}

if (userCapacity > 100)

{

cout << "Invalid input: Cannot unload more than 100 cargo" << endl;

}

if (TankTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot unload cargo while vehicle is moving" << endl;

}

}

else {

TankTruck.loadCargo(userCapacity);

cout << "Cargo loaded successfully" << endl;

fout << "User took out " << userCapacity << " cargo" << endl;

}

system("pause");

break;

case 18: //Check cargo

cout << "Current cargo is " << TankTruck.getCapacity() << endl;

system("pause");

break;

case 19: //Turn right

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (TankTruck.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else if (TankTruck.getSpeed() > 10)

{

cout << "Invalid input: Cannot turn if speed is greater than 10" << endl;

}

else {

TankTruck.turnRight(turnsTripOne);

cout << "You have turned right" << endl;

fout << "User turned right " << endl;

TankTruck.addMileage(10);

}

system("pause");

break;

case 20: //Turn left

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (TankTruck.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else if (TankTruck.getSpeed() > 10)

{

cout << "Invalid input: Cannot turn if speed is greater than 10" << endl;

}

else {

TankTruck.turnLeft(turnsTripOne);

cout << "You have turned left" << endl;

fout << "User turned left" << endl;

TankTruck.addMileage(10);

}

system("pause");

break;

case 21: //Accelerate

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot accelerate if engine is off" << endl;

}

else if (TankTruck.getSpeed() == 45)

{

cout << "Invalid input: Cannot accelerate to more than 45 mph" << endl;

}

else {

TankTruck.accelerate();

cout << "Speed is now increased to " << TankTruck.getSpeed() << endl;

fout << "User accelerated vehicle to " << TankTruck.getSpeed() << endl;

TankTruck.addMileage(10);

}

system("pause");

break;

case 22: //Decelerate

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot decelerate if engine is off" << endl;

}

else if (TankTruck.getSpeed() == 0)

{

cout << "Invalid input: Cannot decelerate to less than 0 mph" << endl;

}

else {

TankTruck.decelerate();

cout << "Speed is now decreased to " << TankTruck.getSpeed() << endl;

fout << "User decelerated vehicle to " << TankTruck.getSpeed() << endl;

}

system("pause");

break;

case 23: //Return to last location

for (int i = 0; i < turnsTripTwo.size(); i++)

{

if (turnsTripTwo.at(i) == 1)

{

turnsTripTwo[i] = 3;

}

else {

turnsTripTwo[i] = 1;

}

}

reverse(turnsTripTwo.begin(), turnsTripTwo.end());

for (int i = 0; i < turnsTripTwo.size(); i++)

{

if (turnsTripTwo.at(i) == 1)

{

cout << "Turn left..." << endl;

fout << "User turned left" << endl;

TankTruck.addMileage(10);

}

else {

cout << "Turn right..." << endl;

fout << "User turned right" << endl;

TankTruck.addMileage(10);

}

}

location[location.size() - 1] = NULL; //Deletes last location

cout << "You are at home now" << endl;

fout << "User has returned to last location" << endl;

TankTruck.addMileage(20);

system("pause");

break;

case 24: //Reached destination

if (TankTruck.getEngineOn())

{

cout << "Invalid input: Cannot get out of vehicle if engine is on" << endl;

}

else {

cout << "What location is this? Type 2 for Airport. Type 3 for Marina: ";

cin >> userLocation;

if (userLocation == 2)

{

location.push\_back(2);

}

else if (userLocation == 3)

{

location.push\_back(3);

}

else {

cout << "Invalid input: Only values acceptable are 2 or 3" << endl;

}

fout << "User has reached destionation" << endl;

TankTruck.addMileage(20);

}

system("pause");

break;

case 0: //Quit

cout << "Sorry to see you go" << endl;

endOfMenu = true;

system("pause");

break;

};

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

else //HaulTruck

{

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

switch (userInput) {

case 1: //Set age of Truck

cout << "What is the age of truck now?: ";

cin >> userSelection;

if (userSelection < HaulTruck.getAge())

{

cout << "Invalid input. Age cannot be less than last one" << endl;

system("pause");

}

else if (userSelection == HaulTruck.getAge())

{

cout << "The age you entered is the same age stored earlier" << endl;

system("pause");

}

else {

HaulTruck.setAge(userSelection);

cout << "Age stored successfully" << endl;

fout << "Age is stored as " << userSelection << endl;

system("pause");

}

break;

case 2: //Set price of Truck

cout << "What is the price of your truck now?: ";

cin >> userSelection;

if (userSelection > HaulTruck.getPrice())

{

cout << "Invalid input. Price cannot be more than last one" << endl;

system("pause");

}

else if (userSelection == HaulTruck.getPrice())

{

cout << "The price you enteres it the same price stored earlier" << endl;

system("pause");

}

else if (userSelection < 0)

{

cout << "Invalid input: Price cannot be negative" << endl;

system("Pause");

}

else {

HaulTruck.setPrice(userSelection);

cout << "Price stored successfully" << endl;

fout << "Price is stored as " << userSelection << endl;

system("pause");

}

break;

case 3: //Check balance of user

if (vehicle.at(0) == 1)

{

cout << "Your balance is " << Lambo.getBalance() << endl;

}

else if (vehicle.at(0) == 4)

{

cout << "Your balance is " << TankTruck.getBalance() << endl;

}

system("pause");

break;

case 4: //Turn engine on

if (HaulTruck.getEngineOn())

{

cout << "Invalid input: Cannot turn engine on while it is already on" << endl;

}

else {

HaulTruck.turnEngineOn();

cout << "Engine is now on" << endl;

fout << "User turned on the engine" << endl;

}

system("pause");

break;

case 5: //Turn engine off

if (HaulTruck.getEngineOn() == false || HaulTruck.getSpeed() != 0 || HaulTruck.getAirOn() == true || HaulTruck.getLightsOn())

{

if (HaulTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn engine off while it is already off" << endl;

}

if (HaulTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot turn engine off while vehicle is moving" << endl;

}

if (HaulTruck.getAirOn() == true)

{

cout << "Invalid input: Cannot turn engine off while A/C is on" << endl;

}

if (HaulTruck.getLightsOn())

{

cout << "Invalid input: Cannot turn engine off while lights are on" << endl;

}

}

else {

HaulTruck.turnEngineOff();

cout << "Engine is now off" << endl;

fout << "User turned off the engine" << endl;

}

system("pause");

break;

case 6: //Turn lights on

if (HaulTruck.getLightsOn() || HaulTruck.getEngineOn() == false)

{

if (HaulTruck.getLightsOn())

{

cout << "Invalid input: Cannot turn lights on if they are already on" << endl;

}

if (HaulTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn lights on if engine is off" << endl;

}

}

else {

HaulTruck.turnLightsOn();

cout << "Lights are now on" << endl;

fout << "User turned on the lights" << endl;

}

system("pause");

break;

case 7: //Turn lights off

if (HaulTruck.getEngineOn() == false || HaulTruck.getLightsOn() == false)

{

if (HaulTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn turn lights off while engine is off" << endl;

}

if (HaulTruck.getLightsOn() == false)

{

cout << "Invalid input: Cannot turn lights off while they are already off" << endl;

}

}

else {

HaulTruck.turnLightsOff();

cout << "Lights are now off" << endl;

fout << "User turned off the lights" << endl;

}

system("pause");

break;

case 8: //Add Passengers

if (HaulTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot add passengers while vehicle is moving" << endl;

}

else {

cout << "How many passengers do you want to add? ";

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot add 0 passengers" << endl;

}

else if (HaulTruck.getNumPass() + numPass > 8)

{

cout << "Invlalid input: Cannot add passengers to have total of more than 1" << endl;

}

else {

HaulTruck.addPass(numPass);

cout << "Passengers added successfully" << endl;

fout << "User added " << numPass << " passengers " << endl;

}

}

system("pause");

break;

case 9: //Kick Passengers

if (HaulTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot kick passengers while vehicle is moving\nThey will die" << endl;

}

else {

cout << "How many passengers do you want to kick out" << endl;

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot kick 0 passengers" << endl;

}

else if (numPass > HaulTruck.getNumPass())

{

cout << "Invalid input: Cannot kick more passengers than there are" << endl;

}

else {

HaulTruck.kickPass(numPass);

cout << "Passengers kicked out successfully" << endl;

fout << "User kicked out " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 10: //Check mileage

cout << "You car's mileage is: " << HaulTruck.getMileage() << " miles" << endl;

system("pause");

break;

case 11: //Set coefficient of friction

cout << "What is the current coefficient of friction" << endl;

cin >> coFriction;

if (coFriction < 0)

{

cout << "Invalid input: Cannot have negative coefficient of friction" << endl;

}

else if (coFriction > 1)

{

cout << "Invalid input: Cannot have coefficient of friction more than 1" << endl;

}

else {

HaulTruck.setCoFriction(coFriction);

cout << "Coefficient is updated successfully" << endl;

fout << "Coefficeient is updated to " << coFriction << endl;

}

system("pause");

break;

case 12: //Check friction

cout << "Current friction on your car is: " << HaulTruck.getFriction();

system("pause");

break;

case 13: //Turn air on

if (HaulTruck.getAirOn() || HaulTruck.getEngineOn() == false)

{

if (HaulTruck.getAirOn())

{

cout << "Invalid input: Cannot turn A/C on while it is already on" << endl;

}

if (HaulTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn air on if engine is off" << endl;

}

}

else {

HaulTruck.turnAirOn();

cout << "Air is now on" << endl;

fout << "User turned on the air" << endl;

}

system("pause");

break;

case 14: //Turn air off

if (HaulTruck.getAirOn() == false)

{

cout << "Invalid input: Cannot turn air off if it is already off" << endl;

}

else {

HaulTruck.turnAirOff();

cout << "Air is now off" << endl;

fout << "User turned off the air" << endl;

}

system("pause");

break;

case 15: //Change diesel type status

cout << "Type 1 if it IS a diesel type and type 0 if it IS NOT a diesel type: ";

cin >> tempStat;

if (tempStat < 0 || tempStat > 1)

{

cout << "Invalid input: Only accepted values are 0 and 1" << endl;

}

else {

HaulTruck.setDieselTypeStatus(tempStat);

cout << "Diesel type status updated" << endl;

if (tempStat)

{

fout << "Diesel type status updated to true" << endl;

}

else {

fout << "Diesel type status updated to false " << endl;

}

}

system("pause");

break;

case 16: //Load cargo

cout << "How much do you want to load?: " << endl;

cin >> userCapacity;

if (userCapacity < 0 || userCapacity == 0 || userCapacity > 100 || HaulTruck.getCapacity() + userCapacity > 100 || HaulTruck.getSpeed() != 0)

{

if (100 - HaulTruck.getCapacity() < userCapacity)

{

cout << "Invalid input: Cannot add to end up with more than 100 cargo" << endl;

}

if (userCapacity < 0)

{

cout << "Invalid input: Cannot add 0 cargo" << endl;

}

if (userCapacity == 0)

{

cout << "Invalid input: Cannot add 0 cargo" << endl;

}

if (HaulTruck.getCapacity() > 100)

{

cout << "Invalid input: Cannot add more than 100 cargo" << endl;

}

if (HaulTruck.getCapacity() + userCapacity > 100)

{

cout << "Invalid input: Cannot add cargo to have total of more than 100 cargo" << endl;

}

if (HaulTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot add cargo while vehicle is moving" << endl;

}

}

else {

HaulTruck.loadCargo(userCapacity);

cout << "Cargo loaded successfully" << endl;

fout << "User added " << userCapacity << " cargo" << endl;

}

system("pause");

break;

case 17: //Unload Cargo

cout << "How much do you want to unload?: " << endl;

cin >> userCapacity;

if (userCapacity > HaulTruck.getCapacity() || userCapacity < 0 || userCapacity == 0 || userCapacity > 100 || HaulTruck.getSpeed() != 0)

{

if (userCapacity > HaulTruck.getCapacity())

{

cout << "Invalid input: Cannot unload to end up with negative cargo" << endl;

}

if (userCapacity < 0)

{

cout << "Invalid input: Cannot have a negative cargo" << endl;

}

if (userCapacity == 0)

{

cout << "Invalid input: Cannot unload 0 cargo" << endl;

}

if (userCapacity > 100)

{

cout << "Invalid input: Cannot unload more than 100 cargo" << endl;

}

if (HaulTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot unload cargo while vehicle is moving" << endl;

}

}

else {

HaulTruck.loadCargo(userCapacity);

cout << "Cargo unloaded successfully" << endl;

fout << "User took out " << userCapacity << " cargo" << endl;

}

system("pause");

break;

case 18: //Check cargo

cout << "Current cargo is " << HaulTruck.getCapacity() << endl;

system("pause");

break;

case 19: //Turn right

if (HaulTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (HaulTruck.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else if (HaulTruck.getSpeed() > 10)

{

cout << "Invalid input: Cannot turn if speed is greater than 10" << endl;

}

else {

HaulTruck.turnRight(turnsTripOne);

cout << "You have turned right" << endl;

fout << "User turned right " << endl;

HaulTruck.addMileage(10);

}

system("pause");

break;

case 20: //Turn left

if (HaulTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (HaulTruck.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else if (HaulTruck.getSpeed() > 10)

{

cout << "Invalid input: Cannot turn if speed is greater than 10" << endl;

}

else {

HaulTruck.turnLeft(turnsTripOne);

cout << "You have turned left" << endl;

fout << "User turned left " << endl;

HaulTruck.addMileage(10);

}

system("pause");

break;

case 21: //Accelerate

if (HaulTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot accelerate if engine is off" << endl;

}

else if (HaulTruck.getSpeed() == 45)

{

cout << "Invalid input: Cannot accelerate to more than 45 mph" << endl;

}

else {

HaulTruck.accelerate();

cout << "Speed is now increased to " << HaulTruck.getSpeed() << endl;

fout << "User increased speed to " << HaulTruck.getSpeed() << endl;

HaulTruck.addMileage(10);

}

system("pause");

break;

case 22: //Decelerate

if (HaulTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot decelerate if engine is off" << endl;

}

else if (HaulTruck.getSpeed() == 0)

{

cout << "Invalid input: Cannot decelerate to less than 0 mph" << endl;

}

else {

HaulTruck.decelerate();

cout << "Speed is now decreased to " << HaulTruck.getSpeed() << endl;

fout << "Speed is decreased to " << HaulTruck.getSpeed() << endl;

HaulTruck.addMileage(10);

}

system("pause");

break;

case 23: //Return to last location

for (int i = 0; i < turnsTripTwo.size(); i++)

{

if (turnsTripTwo.at(i) == 1)

{

turnsTripTwo[i] = 3;

}

else {

turnsTripTwo[i] = 1;

}

}

reverse(turnsTripTwo.begin(), turnsTripTwo.end());

for (int i = 0; i < turnsTripTwo.size(); i++)

{

if (turnsTripTwo.at(i) == 1)

{

cout << "Turn left..." << endl;

fout << "User turned left " << endl;

HaulTruck.addMileage(10);

}

else {

cout << "Turn right..." << endl;

fout << "User turned right " << endl;

HaulTruck.addMileage(10);

}

}

location[location.size() - 1] = NULL; //Deletes last location

cout << "You are at home now" << endl;

fout << "User has returned to last lcoation" << endl;

HaulTruck.addMileage(20);

system("pause");

break;

case 24: //Reached destination

if (HaulTruck.getEngineOn())

{

cout << "Invalid input: Cannot get out of vehicle if engine is on" << endl;

}

else {

cout << "What location is this? Type 2 for Airport. Type 3 for Marina: ";

cin >> userLocation;

if (userLocation == 2)

{

location.push\_back(2);

}

else if (userLocation == 3)

{

location.push\_back(3);

}

else {

cout << "Invalid input: Only values acceptable are 2 or 3" << endl;

}

fout << "User has ended this trip" << endl;

HaulTruck.addMileage(20);

}

system("pause");

break;

case 0: //Quit

cout << "Sorry to see you go" << endl;

endOfMenu = true;

system("pause");

break;

};

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

if (userInput == 0)

{

break;

}

} while (userInput != 24);

}

else if (userMenu == 3) //If user selected Plane

{

vehicle.push\_back(7);

do {

system("CLS");

cout << "You are driving Plane" << endl;

printPlaneMenu();

cin >> userInput;

if (location.at(location.size() - 2) == 1) //Airbus

{

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

switch (userInput)

{

case 1: //Set Age

cout << "What is the age of plane now?: ";

cin >> userSelection;

if (userSelection < AirbusA310.getAge())

{

cout << "Invalid input. Age cannot be less than last one" << endl;

system("pause");

}

else if (userSelection == AirbusA310.getAge())

{

cout << "The age you entered is the same age stored earlier" << endl;

system("pause");

}

else {

AirbusA310.setAge(userSelection);

cout << "Age stored successfully" << endl;

fout << "Age stored as " << userSelection << endl;

system("pause");

}

break;

case 2: //Set price

cout << "What is the price of your plane now?: ";

cin >> userSelection;

if (userSelection > AirbusA310.getPrice())

{

cout << "Invalid input. Price cannot be more than last one" << endl;

system("pause");

}

else if (userSelection == AirbusA310.getPrice())

{

cout << "The price you enteres it the same price stored earlier" << endl;

system("pause");

}

else if (userSelection < 0)

{

cout << "Invalid input: Price cannot be negative" << endl;

system("Pause");

}

else {

AirbusA310.setPrice(userSelection);

cout << "Price stored successfully" << endl;

fout << "Price stored as " << userSelection << endl;

system("pause");

}

break;

case 3: //Check balance

if (vehicle.at(0) == 1)

{

cout << "Your balance is " << Lambo.getBalance() << endl;

}

else if (vehicle.at(0) == 4)

{

cout << "Your balance is " << TankTruck.getBalance() << endl;

}

system("pause");

break;

case 4: //Turn engine on

if (AirbusA310.getEngineOn())

{

cout << "Invalid input: Cannot turn engine on while it is already on" << endl;

}

else {

AirbusA310.turnEngineOn();

cout << "Engine is now on" << endl;

fout << "User turne don the engine" << endl;

}

system("pause");

break;

case 5: //Turn engine off

if (AirbusA310.getEngineOn() == false || AirbusA310.getSpeed() != 0 || AirbusA310.getAirOn() == true || AirbusA310.getLightsOn())

{

if (AirbusA310.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn engine off while it is already off" << endl;

}

if (AirbusA310.getSpeed() != 0)

{

cout << "Invalid input: Cannot turn engine off while vehicle is moving" << endl;

}

if (AirbusA310.getAirOn() == true)

{

cout << "Invalid input: Cannot turn engine off while A/C is on" << endl;

}

if (AirbusA310.getLightsOn())

{

cout << "Invalid input: Cannot turn engine off while lights are on" << endl;

}

}

else {

AirbusA310.turnEngineOff();

cout << "Engine is now off" << endl;

fout << "User turned off the engine" << endl;

}

system("pause");

break;

case 6: //Turn lights on

if (AirbusA310.getLightsOn() || AirbusA310.getEngineOn() == false)

{

if (AirbusA310.getLightsOn())

{

cout << "Invalid input: Cannot turn lights on if they are already on" << endl;

}

if (AirbusA310.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn lights on if engine is off" << endl;

}

}

else {

AirbusA310.turnLightsOn();

cout << "Lights are now on" << endl;

fout << "User turned on the lights" << endl;

}

system("pause");

break;

case 7: //Turn lights off

if (AirbusA310.getEngineOn() == false || AirbusA310.getLightsOn() == false)

{

if (AirbusA310.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn turn lights off while engine is off" << endl;

}

if (AirbusA310.getLightsOn() == false)

{

cout << "Invalid input: Cannot turn lights off while they are already off" << endl;

}

}

else {

AirbusA310.turnLightsOff();

cout << "Lights are now off" << endl;

fout << "User turned off the lights" << endl;

}

system("pause");

break;

case 8: //Add passengers

if (AirbusA310.getSpeed() != 0)

{

cout << "Invalid input: Cannot add passengers while vehicle is moving" << endl;

}

else {

cout << "How many passengers do you want to add? ";

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot add 0 passengers" << endl;

}

else if (AirbusA310.getNumPass() + numPass > 200)

{

cout << "Invlalid input: Cannot add passengers to have total of more than 200" << endl;

}

else {

AirbusA310.addPass(numPass);

cout << "Passengers added successfully" << endl;

fout << "User adds " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 9: //Kick passengers

if (AirbusA310.getSpeed() != 0)

{

cout << "Invalid input: Cannot kick passengers while vehicle is moving\nThey will die" << endl;

}

else {

cout << "How many passengers do you want to kick out" << endl;

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot kick 0 passengers" << endl;

}

else if (numPass > AirbusA310.getNumPass())

{

cout << "Invalid input: Cannot kick more passengers than there are" << endl;

}

else {

AirbusA310.kickPass(numPass);

cout << "Passengers kicked out successfully" << endl;

fout << "User kicked out " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 10: //Turn air on

if (AirbusA310.getAirOn() || AirbusA310.getEngineOn() == false)

{

if (AirbusA310.getAirOn())

{

cout << "Invalid input: Cannot turn A/C on while it is already on" << endl;

}

if (AirbusA310.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn air on if engine is off" << endl;

}

}

else {

AirbusA310.turnAirOn();

cout << "Air is now on" << endl;

fout << "User turned the air on" << endl;

}

system("pause");

break;

case 11: //Turn air off

if (AirbusA310.getAirOn() == false)

{

cout << "Invalid input: Cannot turn air off if it is already off" << endl;

}

else {

AirbusA310.turnAirOff();

cout << "Air is now off" << endl;

fout << "User turned the air off" << endl;

}

system("pause");

break;

case 12: //Check altitude

cout << "Your plane's altitude is " << AirbusA310.getAltitude() << " ft" << endl;

system("Pause");

break;

case 13: //Change altitude

if (AirbusA310.getAltitude() >= 15000 && AirbusA310.getAltitude() <= 35000) //If in air

{

if (AirbusA310.getSpeed() > 140)

{

cout << "Invalid input: Cannot change altitude if speed is 141 - 500 mph" << endl;

}

else {

cout << "What would you like to set altitude to: ";

cin >> userAlt;

if (userAlt < 15000)

{

cout << "Invalid input: Cannot set the altitude less than 15000 without selecting to land plane" << endl;

}

else if (userAlt > 35000)

{

cout << "Invalid input: Cannot set the altitude more than 35000" << endl;

}

else {

AirbusA310.setAltitude(userAlt);

cout << "Altitude is set successfully" << endl;

fout << "User changed altitude to " << userAlt << endl;

}

}

}

else { //If on land

cout << "Invalid input. Cannot change altitude if plane is on land. First take off" << endl;

}

system("pause");

break;

case 14: //Take off

if (AirbusA310.getEngineOn() == false)

{

cout << "Invalid input. Cannot fly plane if engine is off" << endl;

}

else {

if (AirbusA310.getAltitude() > 15000)

{

cout << "Invalid input. Cannot take off if already in air" << endl;

}

else if (AirbusA310.getSpeed() < 100)

{

cout << "Invalid input. Cannot take off if speed is less than 100 mph" << endl;

}

else if (AirbusA310.getSpeed() > 140)

{

cout << "Invalid input. Cannot take off if speed is more than 140 mph" << endl;

}

else {

cout << "Set altitude between 15,000 and 35,000" << endl;

cin >> userAlt;

if (userAlt < 15000)

{

cout << "Invalid input. Cannot change altitude to be less than 15,000" << endl;

}

else if (userAlt > 35000)

{

cout << "Invalid input. Cannot change altitude to be mroe than 35,000" << endl;

}

else {

AirbusA310.setAltitude(userAlt);

cout << "You are now in air" << endl;

fout << "Plane has taken off" << endl;

AirbusA310.addMileage(100);

}

}

}

system("pause");

break;

case 15: //Land

if (AirbusA310.getAltitude() != 0) {

if (AirbusA310.getSpeed() > 140)

{

cout << "Invalid input: Cannot land if speed is more than 140 mph" << endl;

}

else {

AirbusA310.setAltitude(0);

cout << "Landed successfully" << endl;

fout << "Plane has landed" << endl;

AirbusA310.addMileage(100);

}

}

else {

cout << "Invalid input: Cannot land if plane if it is already on ground" << endl;

}

system("Pause");

break;

case 16: //Turn right

if (AirbusA310.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (AirbusA310.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else {

AirbusA310.turnRight(turnsTripTwo);

cout << "You have turned right" << endl;

fout << "User turned right" << endl;

AirbusA310.addMileage(100);

}

system("pause");

break;

case 17: //Turn left

if (AirbusA310.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (AirbusA310.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else {

AirbusA310.turnLeft(turnsTripTwo);

cout << "You have turned left" << endl;

fout << "User turned left " << endl;

}

system("pause");

break;

case 18: //Accelerate

if (AirbusA310.getEngineOn() == false)

{

cout << "Invalid input: Cannot accelerate if engine is off" << endl;

}

else if (AirbusA310.getAltitude() == 0) {

if (AirbusA310.getSpeed() == 140)

{

cout << "Invalid input: Cannot accelerate to more than 140 mph" << endl;

}

else {

AirbusA310.accelerate();

cout << "Speed is now increased to " << AirbusA310.getSpeed() << endl;

fout << "User has increased speed to " << AirbusA310.getSpeed() << endl;

AirbusA310.addMileage(100);

}

}

else if (AirbusA310.getSpeed() == 500)

{

cout << "Invalid input: Cannot accelerate to more than 500 mph while in air" << endl;

}

else {

AirbusA310.accelerate();

cout << "Speed is now increased to " << AirbusA310.getSpeed() << endl;

fout << "User has increased speed to " << AirbusA310.getSpeed() << endl;

AirbusA310.addMileage(100);

}

system("pause");

break;

case 19: //Decelerate

if (AirbusA310.getEngineOn() == false)

{

cout << "Invalid input: Cannot decelerate if engine is off" << endl;

}

else if (AirbusA310.getSpeed() == 0)

{

cout << "Invalid input: Cannot decelerate to less than 0 mph" << endl;

}

else if (AirbusA310.getAltitude() >= 15000 && AirbusA310.getAltitude() <= 35000) {

if (AirbusA310.getSpeed() == 120)

{

cout << "Invalid input: Cannot decelerate to less than 120 mph while in air" << endl;

}

else

{

AirbusA310.decelerate();

cout << "Speed is now decreased to " << AirbusA310.getSpeed() << endl;

fout << "User has decreased speed to " << AirbusA310.getSpeed() << endl;

AirbusA310.addMileage(100);

}

}

else {

AirbusA310.decelerate();

cout << "Speed is now decreased to " << AirbusA310.getSpeed() << endl;

fout << "User has decreased speed to " << AirbusA310.getSpeed() << endl;

AirbusA310.addMileage(100);

}

system("pause");

break;

case 20: //Return to last location

for (int i = 0; i < turnsTripTwo.size(); i++)

{

if (turnsTripTwo.at(i) == 1)

{

turnsTripTwo[i] = 3;

}

else {

turnsTripTwo[i] = 1;

}

}

reverse(turnsTripTwo.begin(), turnsTripTwo.end());

for (int i = 0; i < turnsTripTwo.size(); i++)

{

if (turnsTripTwo.at(i) == 1)

{

cout << "Turn left..." << endl;

fout << "User turned left" << endl;

AirbusA310.addMileage(100);

}

else {

cout << "Turn right..." << endl;

fout << "User turned right " << endl;

AirbusA310.addMileage(100);

}

}

location[location.size() - 1] = NULL; //Deletes last location

cout << "You are at home now" << endl;

fout << "User has returned to last location" << endl;

AirbusA310.addMileage(100);

system("pause");

break;

case 21: //Reached destination

if (AirbusA310.getEngineOn())

{

cout << "Invalid input: Cannot get out of vehicle if engine is on" << endl;

}

else {

location.push\_back(2);

fout << "User has ended this trip" << endl;

}

system("pause");

break;

case 22: //Quit

cout << "Sorry to see you go" << endl;

endOfMenu = true;

system("pause");

break;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

else //Boeing

{

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

switch (userInput)

{

case 1: //Set Age

cout << "What is the age of plane now?: ";

cin >> userSelection;

if (userSelection < Boeing747.getAge())

{

cout << "Invalid input. Age cannot be less than last one" << endl;

system("pause");

}

else if (userSelection == Boeing747.getAge())

{

cout << "The age you entered is the same age stored earlier" << endl;

system("pause");

}

else {

Boeing747.setAge(userSelection);

cout << "Age stored successfully" << endl;

fout << "Age is stored as " << userSelection << endl;

system("pause");

}

break;

case 2: //Set price

cout << "What is the price of your plane now?: ";

cin >> userSelection;

if (userSelection > Boeing747.getPrice())

{

cout << "Invalid input. Price cannot be more than last one" << endl;

system("pause");

}

else if (userSelection == Boeing747.getPrice())

{

cout << "The price you enteres it the same price stored earlier" << endl;

system("pause");

}

else if (userSelection < 0)

{

cout << "Invalid input: Price cannot be negative" << endl;

system("Pause");

}

else {

Boeing747.setPrice(userSelection);

cout << "Price stored successfully" << endl;

fout << "Price is stored as " << userSelection << endl;

system("pause");

}

break;

case 3: //Check balance

if (vehicle.at(0) == 1)

{

cout << "Your balance is " << Lambo.getBalance() << endl; //Don't change

}

else if (vehicle.at(0) == 4)

{

cout << "Your balance is " << TankTruck.getBalance() << endl; //Don't change

}

system("pause");

break;

case 4: //Turn engine on

if (Boeing747.getEngineOn())

{

cout << "Invalid input: Cannot turn engine on while it is already on" << endl;

}

else {

Boeing747.turnEngineOn();

cout << "Engine is now on" << endl;

fout << "User turned on the engine " << endl;

}

system("pause");

break;

case 5: //Turn engine off

if (Boeing747.getEngineOn() == false || Boeing747.getSpeed() != 0 || Boeing747.getAirOn() == true || Boeing747.getLightsOn())

{

if (Boeing747.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn engine off while it is already off" << endl;

}

if (Boeing747.getSpeed() != 0)

{

cout << "Invalid input: Cannot turn engine off while vehicle is moving" << endl;

}

if (Boeing747.getAirOn() == true)

{

cout << "Invalid input: Cannot turn engine off while A/C is on" << endl;

}

if (Boeing747.getLightsOn())

{

cout << "Invalid input: Cannot turn engine off while lights are on" << endl;

}

}

else {

Boeing747.turnEngineOff();

cout << "Engine is now off" << endl;

fout << "User turned off the engine" << endl;

}

system("pause");

break;

case 6: //Turn lights on

if (Boeing747.getLightsOn() || Boeing747.getEngineOn() == false)

{

if (Boeing747.getLightsOn())

{

cout << "Invalid input: Cannot turn lights on if they are already on" << endl;

}

if (Boeing747.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn lights on if engine is off" << endl;

}

}

else {

Boeing747.turnLightsOn();

cout << "Lights are now on" << endl;

fout << "User turned on the lights " << endl;

}

system("pause");

break;

case 7: //Turn lights off

if (Boeing747.getEngineOn() == false || Boeing747.getLightsOn() == false)

{

if (Boeing747.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn turn lights off while engine is off" << endl;

}

if (Boeing747.getLightsOn() == false)

{

cout << "Invalid input: Cannot turn lights off while they are already off" << endl;

}

}

else {

Boeing747.turnLightsOff();

cout << "Lights are now off" << endl;

fout << "User turned off the lights " << endl;

}

system("pause");

break;

case 8: //Add passengers

if (Boeing747.getSpeed() != 0)

{

cout << "Invalid input: Cannot add passengers while vehicle is moving" << endl;

}

else {

cout << "How many passengers do you want to add? ";

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot add 0 passengers" << endl;

}

else if (Boeing747.getNumPass() + numPass > 200)

{

cout << "Invlalid input: Cannot add passengers to have total of more than 200" << endl;

}

else {

Boeing747.addPass(numPass);

cout << "Passengers added successfully" << endl;

fout << "User added " << numPass << " passengers " << endl;

}

}

system("pause");

break;

case 9: //Kick passengers

if (Boeing747.getSpeed() != 0)

{

cout << "Invalid input: Cannot kick passengers while vehicle is moving\nThey will die" << endl;

}

else {

cout << "How many passengers do you want to kick out" << endl;

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot kick 0 passengers" << endl;

}

else if (numPass > Boeing747.getNumPass())

{

cout << "Invalid input: Cannot kick more passengers than there are" << endl;

}

else {

Boeing747.kickPass(numPass);

cout << "Passengers kicked out successfully" << endl;

fout << "User kicked out " << numPass << " passengers " << endl;

}

}

system("pause");

break;

case 10: //Turn air on

if (Boeing747.getAirOn() || Boeing747.getEngineOn() == false)

{

if (Boeing747.getAirOn())

{

cout << "Invalid input: Cannot turn A/C on while it is already on" << endl;

}

if (Boeing747.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn air on if engine is off" << endl;

}

}

else {

Boeing747.turnAirOn();

cout << "Air is now on" << endl;

fout << "User turned on the air " << endl;

}

system("pause");

break;

case 11: //Turn air off

if (Boeing747.getAirOn() == false)

{

cout << "Invalid input: Cannot turn air off if it is already off" << endl;

}

else {

Boeing747.turnAirOff();

cout << "Air is now off" << endl;

fout << "User turned off the air" << endl;

}

system("pause");

break;

case 12: //Check altitude

cout << "Your plane's altitude is " << Boeing747.getAltitude() << " ft" << endl;

system("Pause");

break;

case 13: //Change altitude

if (Boeing747.getAltitude() >= 15000 && Boeing747.getAltitude() <= 35000) //If in air

{

if (Boeing747.getSpeed() > 140)

{

cout << "Invalid input: Cannot change altitude if speed is 141 - 500 mph" << endl;

}

else {

cout << "What would you like to set altitude to: ";

cin >> userAlt;

if (userAlt < 15000)

{

cout << "Invalid input: Cannot set the altitude less than 15000 without selecting to land plane" << endl;

}

else if (userAlt > 35000)

{

cout << "Invalid input: Cannot set the altitude more than 35000" << endl;

}

else {

Boeing747.setAltitude(userAlt);

cout << "Altitude is set successfully" << endl;

fout << "User changed altitude to " << userAlt << endl;

Boeing747.addMileage(200);

}

}

}

else { //If on land

cout << "Invalid input. Cannot change altitude if plane is on land. First take off" << endl;

}

system("pause");

break;

case 14: //Take off

if (Boeing747.getEngineOn() == false)

{

cout << "Invalid input. Cannot fly plane if engine is off" << endl;

}

else {

if (Boeing747.getAltitude() > 15000)

{

cout << "Invalid input. Cannot take off if already in air" << endl;

}

else if (Boeing747.getSpeed() < 100)

{

cout << "Invalid input. Cannot take off if speed is less than 100 mph" << endl;

}

else if (Boeing747.getSpeed() > 140)

{

cout << "Invalid input. Cannot take off if speed is more than 140 mph" << endl;

}

else {

cout << "Set altitude between 15,000 and 35,000" << endl;

cin >> userAlt;

if (userAlt < 15000)

{

cout << "Invalid input. Cannot change altitude to be less than 15,000" << endl;

}

else if (userAlt > 35000)

{

cout << "Invalid input. Cannot change altitude to be mroe than 35,000" << endl;

}

else {

Boeing747.setAltitude(userAlt);

cout << "You are now in air" << endl;

fout << "Plane has taken off " << endl;

Boeing747.addMileage(200);

}

}

}

system("pause");

break;

case 15: //Land

if (Boeing747.getAltitude() != 0)

{

if (Boeing747.getSpeed() > 140)

{

cout << "Invalid input: Cannot land if speed is more than 140 mph" << endl;

}

else {

Boeing747.setAltitude(0);

cout << "Landed successfully" << endl;

fout << "Plane has landed" << endl;

Boeing747.addMileage(200);

}

}

else {

cout << "Invalid input: Cannot land if plane is already on ground" << endl;

}

system("Pause");

break;

case 16: //Turn right

if (Boeing747.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (Boeing747.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else {

Boeing747.turnRight(turnsTripTwo);

cout << "You have turned right" << endl;

fout << "User turned right" << endl;

Boeing747.addMileage(200);

}

system("pause");

break;

case 17: //Turn left

if (Boeing747.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (Boeing747.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else {

Boeing747.turnLeft(turnsTripTwo);

cout << "You have turned left" << endl;

fout << "User turned left " << endl;

Boeing747.addMileage(200);

}

system("pause");

break;

case 18: //Accelerate

if (Boeing747.getEngineOn() == false)

{

cout << "Invalid input: Cannot accelerate if engine is off" << endl;

}

else if (Boeing747.getAltitude() == 0) {

if (Boeing747.getSpeed() == 140)

{

cout << "Invalid input: Cannot accelerate to more than 140 mph" << endl;

}

else {

Boeing747.accelerate();

cout << "Speed is now increased to " << Boeing747.getSpeed() << endl;

fout << "User increased speed to " << Boeing747.getSpeed() << endl;

Boeing747.addMileage(200);

}

}

else if (Boeing747.getSpeed() == 500)

{

cout << "Invalid input: Cannot accelerate to more than 500 mph while in air" << endl;

}

else {

Boeing747.accelerate();

cout << "Speed is now increased to " << Boeing747.getSpeed() << endl;

fout << "User increased speed to " << Boeing747.getSpeed() << endl;

Boeing747.addMileage(200);

}

system("pause");

break;

case 19: //Decelerate

if (Boeing747.getEngineOn() == false)

{

cout << "Invalid input: Cannot decelerate if engine is off" << endl;

}

else if (Boeing747.getSpeed() == 0)

{

cout << "Invalid input: Cannot decelerate to less than 0 mph" << endl;

}

else if (Boeing747.getAltitude() >= 15000 && Boeing747.getAltitude() <= 35000) {

if (Boeing747.getSpeed() == 120)

{

cout << "Invalid input: Cannot decelerate to less than 120 mph while in air" << endl;

}

else

{

Boeing747.decelerate();

cout << "Speed is now decreased to " << Boeing747.getSpeed() << endl;

fout << "User decreased speed to " << Boeing747.getSpeed() << endl;

Boeing747.addMileage(200);

}

}

else {

Boeing747.decelerate();

cout << "Speed is now decreased to " << Boeing747.getSpeed() << endl;

fout << "User decreased speed to " << Boeing747.getSpeed() << endl;

Boeing747.addMileage(200);

}

system("pause");

break;

case 20: //Return to last location

for (int i = 0; i < turnsTripTwo.size(); i++)

{

if (turnsTripTwo.at(i) == 1)

{

turnsTripTwo[i] = 3;

}

else {

turnsTripTwo[i] = 1;

}

}

reverse(turnsTripTwo.begin(), turnsTripTwo.end());

for (int i = 0; i < turnsTripTwo.size(); i++)

{

if (turnsTripTwo.at(i) == 1)

{

cout << "Turn left..." << endl;

fout << "User turned left " << endl;

Boeing747.addMileage(200);

}

else {

cout << "Turn right..." << endl;

fout << "User turned right " << endl;

Boeing747.addMileage(200);

}

}

location[location.size() - 1] = NULL; //Deletes last location

cout << "You are at home now" << endl;

fout << "User has returned to last location" << endl;

Boeing747.addMileage(200);

system("pause");

break;

case 21: //Reached destination

if (Boeing747.getEngineOn())

{

cout << "Invalid input: Cannot get out of vehicle if engine is on" << endl;

}

else {

location.push\_back(2);

fout << "User has ended this trip" << endl;

Boeing747.addMileage(200);

}

system("pause");

break;

case 22: //Quit

cout << "Sorry to see you go" << endl;

endOfMenu = true;

system("pause");

break;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

} while (userInput != 21);

}

else {

cout << "Invalid input. Neight Car, or truck, or plane was selected" << endl;

}

system("pause");

}

//................... . . . . . . . . . . . . . . . . . . . . . . . . .

else if (location.back() == 3) //If user is at Marina

{

cout << "You are at Marina right now" << endl;

cout << "1. Take the Car" << endl;

cout << "2. Take the Truck" << endl;

cout << "3. Take the Boat" << endl;

cin >> userMenu;

if (userMenu == 1) //If user selected Car

{

vehicle.push\_back(2);

do {

system("CLS");

cout << "You are driving Car" << endl;

printCarMenu();

cin >> userInput;

if (location.at(location.size() - 2) == 1) //Car

{

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

switch (userInput) {

case 1: //Set age of car

cout << "What is the age of car now?: ";

cin >> userSelection;

if (userSelection < Lambo.getAge())

{

cout << "Invalid input. Age cannot be less than last one" << endl;

system("pause");

}

else if (userSelection == Lambo.getAge())

{

cout << "The age you entered is the same age stored earlier" << endl;

system("pause");

}

else {

Lambo.setAge(userSelection);

cout << "Age stored successfully" << endl;

fout << "Age stored as " << userSelection << endl;

system("pause");

}

break;

case 2: //Set price of car

cout << "What is the price of your car now?: ";

cin >> userSelection;

if (userSelection > Lambo.getPrice())

{

cout << "Invalid input. Price cannot be more than last one" << endl;

system("pause");

}

else if (userSelection == Lambo.getPrice())

{

cout << "The price you enteres it the same price stored earlier" << endl;

system("pause");

}

else if (userSelection < 0)

{

cout << "Invalid input: Price cannot be negative" << endl;

system("Pause");

}

else {

Lambo.setPrice(userSelection);

cout << "Price stored successfully" << endl;

fout << "Price is stored as " << userSelection << endl;

system("pause");

}

break;

case 3: //Check balance of user

if (vehicle.at(0) == 1)

{

cout << "Your balance is " << Lambo.getBalance() << endl;

}

else if (vehicle.at(0) == 4)

{

cout << "Your balance is " << TankTruck.getBalance() << endl;

}

system("pause");

break;

case 4: //Turn engine on

if (Lambo.getEngineOn())

{

cout << "Invalid input: Cannot turn engine on while it is already on" << endl;

}

else {

Lambo.turnEngineOn();

cout << "Engine is now on" << endl;

fout << "User turned on the engine" << endl;

}

system("pause");

break;

case 5: //Turn engine off

if (Lambo.getEngineOn() == false || Lambo.getSpeed() != 0 || Lambo.getAirOn() == true || Lambo.getSunroofOpen() == true || Lambo.getLightsOn())

{

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn engine off while it is already off" << endl;

}

if (Lambo.getSpeed() != 0)

{

cout << "Invalid input: Cannot turn engine off while vehicle is moving" << endl;

}

if (Lambo.getAirOn() == true)

{

cout << "Invalid input: Cannot turn engine off while A/C is on" << endl;

}

if (Lambo.getSunroofOpen() == true)

{

cout << "Invalid input: Cannot turn engine off while sunroof is open" << endl;

}

if (Lambo.getLightsOn())

{

cout << "Invalid input: Cannot turn engine off while lights are on" << endl;

}

}

else {

Lambo.turnEngineOff();

cout << "Engine is now off" << endl;

fout << "User turned off the engine" << endl;

}

system("pause");

break;

case 6: //Turn lights on

if (Lambo.getLightsOn() || Lambo.getEngineOn() == false)

{

if (Lambo.getLightsOn())

{

cout << "Invalid input: Cannot turn lights on if they are already on" << endl;

}

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn lights on if engine is off" << endl;

}

}

else {

Lambo.turnLightsOn();

cout << "Lights are now on" << endl;

fout << "User turned on the lights" << endl;

}

system("pause");

break;

case 7: //Turn lights off

if (Lambo.getEngineOn() == false || Lambo.getLightsOn() == false)

{

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn turn lights off while engine is off" << endl;

}

if (Lambo.getLightsOn() == false)

{

cout << "Invalid input: Cannot turn lights off while they are already off" << endl;

}

}

else {

Lambo.turnLightsOff();

cout << "Lights are now off" << endl;

fout << "User turned off the lights" << endl;

}

system("pause");

break;

case 8: //Add Passengers

if (Lambo.getSpeed() != 0)

{

cout << "Invalid input: Cannot add passengers while vehicle is moving" << endl;

}

else {

cout << "How many passengers do you want to add? ";

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot add 0 passengers" << endl;

}

else if (Lambo.getNumPass() + numPass > 4)

{

cout << "Invlalid input: Cannot add passengers to have total of more than 4" << endl;

}

else {

Lambo.addPass(numPass);

cout << "Passengers added successfully" << endl;

fout << "User added " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 9: //Kick Passengers

if (Lambo.getSpeed() != 0)

{

cout << "Invalid input: Cannot kick passengers while vehicle is moving\nThey will die" << endl;

}

else {

cout << "How many passengers do you want to kick out" << endl;

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot kick 0 passengers" << endl;

}

else if (numPass > Lambo.getNumPass())

{

cout << "Invalid input: Cannot kick more passengers than there are" << endl;

}

else {

Lambo.kickPass(numPass);

cout << "Passengers kicked out successfully successfully" << endl;

fout << "User kicked out " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 10: //Check mileage

cout << "You car's mileage is: " << Lambo.getMileage() << " miles" << endl;

system("pause");

break;

case 11: //Set coefficient of friction

cout << "What is the current coefficient of friction" << endl;

cin >> coFriction;

if (coFriction < 0)

{

cout << "Invalid input: Cannot have negative coefficient of friction" << endl;

}

else if (coFriction > 1)

{

cout << "Invalid input: Cannot have coefficient of friction more than 1" << endl;

}

else {

Lambo.setCoFriction(coFriction);

cout << "Coefficient is updated successfully" << endl;

fout << "Coefficient is updated to " << coFriction << endl;

}

system("pause");

break;

case 12: //Check friction

cout << "Current friction on your car is: " << Lambo.getFriction();

system("pause");

break;

case 13: //Turn air on

if (Lambo.getAirOn() || Lambo.getEngineOn() == false || Lambo.getSunroofOpen())

{

if (Lambo.getAirOn())

{

cout << "Invalid input: Cannot turn A/C on while it is already on" << endl;

}

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn air on if engine is off" << endl;

}

if (Lambo.getSunroofOpen())

{

cout << "Invalid input: Cannot turn air on if sunroof is open" << endl;

}

}

else {

Lambo.turnAirOn();

cout << "Air is now on" << endl;

fout << "User turned on the air" << endl;

}

system("pause");

break;

case 14: //Turn air off

if (Lambo.getAirOn() == false)

{

cout << "Invalid input: Cannot turn air off if it is already off" << endl;

}

else {

Lambo.turnAirOff();

cout << "Air is now off" << endl;

fout << "User turned off the air" << endl;

}

system("pause");

break;

case 15: //Change race car status

cout << "Type 1 if it IS a race car and type 0 if it IS NOT a race car: ";

cin >> tempStat;

if (tempStat < 0 || tempStat > 1)

{

cout << "Invalid input: Only accepted values are 0 and 1" << endl;

}

else {

Lambo.setRaceCarStatus(tempStat);

cout << "Race Car status updated" << endl;

if (tempStat == true)

{

fout << "Race car status updated to false" << endl;

}

else {

fout << "Race car status updated to true" << endl;

}

}

system("pause");

break;

case 16: //Open sunroof

if (Lambo.getSunroofOpen() || Lambo.getEngineOn() == false || Lambo.getAirOn())

{

if (Lambo.getSunroofOpen())

{

cout << "Invalid input: Cannot open sunroof if it is already open" << endl;

}

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot open sunroof if engine is off" << endl;

}

if (Lambo.getAirOn())

{

cout << "Invalid input: Cannot open sunroof if A/C is on" << endl;

}

}

else {

Lambo.setSunRoofStatus(true);

cout << "Sunroof is now open" << endl;

fout << "User opened the sunroof" << endl;

}

system("pause");

break;

case 17: //Close sunroof

if (Lambo.getSunroofOpen() == false || Lambo.getEngineOn() == false)

{

if (Lambo.getSunroofOpen() == false)

{

cout << "Invalid input: Cannot close sunroof if it is already closed" << endl;

}

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot close sunroof if engine is off" << endl;

}

}

else {

Lambo.setSunRoofStatus(false);

cout << "Sunroof is now closed" << endl;

fout << "User closed the sunroof" << endl;

}

system("pause");

break;

case 18: //Turn right

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (Lambo.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else if (Lambo.getSpeed() > 10)

{

cout << "Invalid input: Cannot turn if speed is greater than 10" << endl;

}

else {

Lambo.turnRight(turnsTripOne);

cout << "You have turned right" << endl;

fout << "User turned right" << endl;

Lambo.addMileage(10);

}

system("pause");

break;

case 19: //Turn left

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (Lambo.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else if (Lambo.getSpeed() > 10)

{

cout << "Invalid input: Cannot turn if speed is greater than 10" << endl;

}

else {

Lambo.turnLeft(turnsTripOne);

cout << "You have turned left" << endl;

fout << "User turned left" << endl;

Lambo.addMileage(10);

}

system("pause");

break;

case 20: //Accelerate

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot accelerate if engine is off" << endl;

}

else if (Lambo.getSpeed() == 45)

{

cout << "Invalid input: Cannot accelerate to more than 45 mph" << endl;

}

else {

Lambo.accelerate();

cout << "Speed is now increased to " << Lambo.getSpeed() << endl;

fout << "User accelerated vehicle to " << Lambo.getSpeed() << endl;

Lambo.addMileage(10);

}

system("pause");

break;

case 21: //Decelerate

if (Lambo.getEngineOn() == false)

{

cout << "Invalid input: Cannot decelerate if engine is off" << endl;

}

else if (Lambo.getSpeed() == 0)

{

cout << "Invalid input: Cannot decelerate to less than 0 mph" << endl;

}

else {

Lambo.decelerate();

cout << "Speed is now decreased to " << Lambo.getSpeed() << endl;

fout << "User decelerated vehicle to " << Lambo.getSpeed() << endl;

Lambo.addMileage(10);

}

system("pause");

break;

case 22: //Return to last location

for (int i = 0; i < turnsTripOne.size(); i++)

{

if (turnsTripOne.at(i) == 1)

{

turnsTripOne[i] = 3;

}

else {

turnsTripOne[i] = 1;

}

}

reverse(turnsTripOne.begin(), turnsTripOne.end());

for (int i = 0; i < turnsTripOne.size(); i++)

{

if (turnsTripOne.at(i) == 1)

{

cout << "Turn left..." << endl;

fout << "User turned left" << endl;

}

else {

cout << "Turn right..." << endl;

fout << "User turned right" << endl;

}

}

location[location.size() - 1] = NULL; //Deletes last location

cout << "You are at home now" << endl;

fout << "User has gone back to last location" << endl;

system("pause");

break;

case 23: //Reached destination

if (Lambo.getEngineOn())

{

cout << "Invalid input: Cannot get out of vehicle if engine is on" << endl;

}

else {

cout << "What location is this? Type 2 for Airport. Type 3 for Marina: ";

cin >> userLocation;

if (userLocation == 2)

{

location.push\_back(2);

}

else if (userLocation == 3)

{

location.push\_back(3);

}

else {

cout << "Invalid input: Only values acceptable are 2 or 3" << endl;

}

fout << "User has ended this trip" << endl;

Lambo.addMileage(20);

}

system("pause");

break;

case 0: //Quit

cout << "Sorry to see you go" << endl;

endOfMenu = true;

system("pause");

break;

};

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

else { //Taxi

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

switch (userInput) {

case 1: //Set age of car

cout << "What is the age of car now?: ";

cin >> userSelection;

if (userSelection < Taxi.getAge())

{

cout << "Invalid input. Age cannot be less than last one" << endl;

system("pause");

}

else if (userSelection == Taxi.getAge())

{

cout << "The age you entered is the same age stored earlier" << endl;

system("pause");

}

else {

Taxi.setAge(userSelection);

cout << "Age stored successfully" << endl;

fout << "Age is stored as " << Taxi.getAge();

system("pause");

}

break;

case 2: //Set price of car

cout << "What is the price of your car now?: ";

cin >> userSelection;

if (userSelection > Taxi.getPrice())

{

cout << "Invalid input. Price cannot be more than last one" << endl;

system("pause");

}

else if (userSelection == Taxi.getPrice())

{

cout << "The price you enteres it the same price stored earlier" << endl;

system("pause");

}

else if (userSelection < 0)

{

cout << "Invalid input: Price cannot be negative" << endl;

system("Pause");

}

else {

Taxi.setPrice(userSelection);

cout << "Price stored successfully" << endl;

fout << "Price is stored as " << userSelection << endl;

system("pause");

}

break;

case 3: //Check balance of user

if (vehicle.at(0) == 1)

{

cout << "Your balance is " << Lambo.getBalance() << endl;

}

else if (vehicle.at(0) == 4)

{

cout << "Your balance is " << TankTruck.getBalance() << endl;

}

system("pause");

break;

case 4: //Turn engine on

if (Taxi.getEngineOn())

{

cout << "Invalid input: Cannot turn engine on while it is already on" << endl;

}

else {

Taxi.turnEngineOn();

cout << "Engine is now on" << endl;

fout << "User turned the engine on" << endl;

}

system("pause");

break;

case 5: //Turn engine off

if (Taxi.getEngineOn() == false || Taxi.getSpeed() != 0 || Taxi.getAirOn() == true || Taxi.getSunroofOpen() == true || Taxi.getLightsOn())

{

if (Taxi.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn engine off while it is already off" << endl;

}

if (Taxi.getSpeed() != 0)

{

cout << "Invalid input: Cannot turn engine off while vehicle is moving" << endl;

}

if (Taxi.getAirOn() == true)

{

cout << "Invalid input: Cannot turn engine off while A/C is on" << endl;

}

if (Taxi.getSunroofOpen() == true)

{

cout << "Invalid input: Cannot turn engine off while sunroof is open" << endl;

}

if (Taxi.getLightsOn())

{

cout << "Invalid input: Cannot turn engine off while lights are on" << endl;

}

}

else {

Taxi.turnEngineOff();

cout << "Engine is now off" << endl;

fout << "User turned the engine off" << endl;

}

system("pause");

break;

case 6: //Turn lights on

if (Taxi.getLightsOn() || Taxi.getEngineOn() == false)

{

if (Taxi.getLightsOn())

{

cout << "Invalid input: Cannot turn lights on if they are already on" << endl;

}

if (Taxi.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn lights on if engine is off" << endl;

}

}

else {

Taxi.turnLightsOn();

cout << "Lights are now on" << endl;

fout << "User turned the lights on" << endl;

}

system("pause");

break;

case 7: //Turn lights off

if (Taxi.getEngineOn() == false || Taxi.getLightsOn() == false)

{

if (Taxi.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn turn lights off while engine is off" << endl;

}

if (Taxi.getLightsOn() == false)

{

cout << "Invalid input: Cannot turn lights off while they are already off" << endl;

}

}

else {

Taxi.turnLightsOff();

cout << "Lights are now off" << endl;

fout << "User turned off the lights" << endl;

}

system("pause");

break;

case 8: //Add Passengers

if (Taxi.getSpeed() != 0)

{

cout << "Invalid input: Cannot add passengers while vehicle is moving" << endl;

}

else {

cout << "How many passengers do you want to add? ";

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot add 0 passengers" << endl;

}

else if (Taxi.getNumPass() + numPass > 4)

{

cout << "Invlalid input: Cannot add passengers to have total of more than 4" << endl;

}

else {

Taxi.addPass(numPass);

cout << "Passengers added successfully" << endl;

fout << "User added " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 9: //Kick Passengers

if (Taxi.getSpeed() != 0)

{

cout << "Invalid input: Cannot kick passengers while vehicle is moving\nThey will die" << endl;

}

else {

cout << "How many passengers do you want to kick out" << endl;

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot kick 0 passengers" << endl;

}

else if (numPass > Taxi.getNumPass())

{

cout << "Invalid input: Cannot kick more passengers than there are" << endl;

}

else {

Taxi.kickPass(numPass);

cout << "Passengers kicked out successfully" << endl;

fout << "User kicked out " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 10: //Check mileage

cout << "You car's mileage is: " << Taxi.getMileage() << " miles" << endl;

system("pause");

break;

case 11: //Set coefficient of friction

cout << "What is the current coefficient of friction" << endl;

cin >> coFriction;

if (coFriction < 0)

{

cout << "Invalid input: Cannot have negative coefficient of friction" << endl;

}

else if (coFriction > 1)

{

cout << "Invalid input: Cannot have coefficient of friction more than 1" << endl;

}

else {

Taxi.setCoFriction(coFriction);

cout << "Coefficient is updated successfully" << endl;

fout << "Coefficient is updated to " << coFriction << endl;

}

system("pause");

break;

case 12: //Check friction

cout << "Current friction on your car is: " << Taxi.getFriction();

system("pause");

break;

case 13: //Turn air on

if (Taxi.getAirOn() || Taxi.getEngineOn() == false || Taxi.getSunroofOpen())

{

if (Taxi.getAirOn())

{

cout << "Invalid input: Cannot turn A/C on while it is already on" << endl;

}

if (Taxi.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn air on if engine is off" << endl;

}

if (Taxi.getSunroofOpen())

{

cout << "Invalid input: Cannot turn air on if sunroof is open" << endl;

}

}

else {

Taxi.turnAirOn();

cout << "Air is now on" << endl;

fout << "User turned on the air" << endl;

}

system("pause");

break;

case 14: //Turn air off

if (Taxi.getAirOn() == false)

{

cout << "Invalid input: Cannot turn air off if it is already off" << endl;

}

else {

Taxi.turnAirOff();

cout << "Air is now off" << endl;

fout << "User turned off the air" << endl;

}

system("pause");

break;

case 15: //Change race car status

cout << "Type 1 if it IS a race car and type 0 if it IS NOT a race car: ";

cin >> tempStat;

if (tempStat < 0 || tempStat > 1)

{

cout << "Invalid input: Only accepted values are 0 and 1" << endl;

}

else {

Taxi.setRaceCarStatus(tempStat);

cout << "Race Car status updated" << endl;

if (tempStat)

{

fout << "Race car status updated to true" << endl;

}

else {

fout << "Race car status updated to false" << endl;

}

}

system("pause");

break;

case 16: //Open sunroof

if (Taxi.getSunroofOpen() || Taxi.getEngineOn() == false || Taxi.getAirOn())

{

if (Taxi.getSunroofOpen())

{

cout << "Invalid input: Cannot open sunroof if it is already open" << endl;

}

if (Taxi.getEngineOn() == false)

{

cout << "Invalid input: Cannot open sunroof if engine is off" << endl;

}

if (Taxi.getAirOn())

{

cout << "Invalid input: Cannot open sunroof if A/C is on" << endl;

}

}

else {

Taxi.setSunRoofStatus(true);

cout << "Sunroof is now open" << endl;

fout << "User opened sunroof" << endl;

}

system("pause");

break;

case 17: //Close sunroof

if (Taxi.getSunroofOpen() == false || Taxi.getEngineOn() == false)

{

if (Taxi.getSunroofOpen() == false)

{

cout << "Invalid input: Cannot close sunroof if it is already closed" << endl;

}

if (Taxi.getEngineOn() == false)

{

cout << "Invalid input: Cannot close sunroof if engine is off" << endl;

}

}

else {

Taxi.setSunRoofStatus(false);

cout << "Sunroof is now closed" << endl;

fout << "User closed sunroof" << endl;

}

system("pause");

break;

case 18: //Turn right

if (Taxi.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (Taxi.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else if (Taxi.getSpeed() > 10)

{

cout << "Invalid input: Cannot turn if speed is greater than 10" << endl;

}

else {

Taxi.turnRight(turnsTripTwo);

cout << "You have turned right" << endl;

fout << "User turned right" << endl;

Taxi.addMileage(10);

}

system("pause");

break;

case 19: //Turn left

if (Taxi.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (Taxi.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else if (Taxi.getSpeed() > 10)

{

cout << "Invalid input: Cannot turn if speed is greater than 10" << endl;

}

else {

Taxi.turnLeft(turnsTripTwo);

cout << "You have turned left" << endl;

fout << "User turned left" << endl;

Taxi.addMileage(10);

}

system("pause");

break;

case 20: //Accelerate

if (Taxi.getEngineOn() == false)

{

cout << "Invalid input: Cannot accelerate if engine is off" << endl;

}

else if (Taxi.getSpeed() == 45)

{

cout << "Invalid input: Cannot accelerate to more than 45 mph" << endl;

}

else {

Taxi.accelerate();

cout << "Speed is now increased to " << Taxi.getSpeed() << endl;

fout << "User accelerated vehicle to " << Taxi.getSpeed() << endl;

Taxi.addMileage(10);

}

system("pause");

break;

case 21: //Decelerate

if (Taxi.getEngineOn() == false)

{

cout << "Invalid input: Cannot decelerate if engine is off" << endl;

}

else if (Taxi.getSpeed() == 0)

{

cout << "Invalid input: Cannot decelerate to less than 0 mph" << endl;

}

else {

Taxi.decelerate();

cout << "Speed is now decreased to " << Taxi.getSpeed() << endl;

fout << "User decelerated vehicle to " << Taxi.getSpeed() << endl;

Taxi.addMileage(10);

}

system("pause");

break;

case 22: //Return to last location

for (int i = 0; i < turnsTripTwo.size(); i++)

{

if (turnsTripOne.at(i) == 1)

{

turnsTripOne[i] = 3;

}

else {

turnsTripOne[i] = 1;

}

}

reverse(turnsTripTwo.begin(), turnsTripTwo.end());

for (int i = 0; i < turnsTripTwo.size(); i++)

{

if (turnsTripTwo.at(i) == 1)

{

cout << "Turn left..." << endl;

fout << "User turned left" << endl;

Taxi.addMileage(10);

}

else {

cout << "Turn right..." << endl;

fout << "User turned right " << endl;

Taxi.addMileage(10);

}

}

location[location.size() - 1] = NULL; //Deletes last location

cout << "You are at home now" << endl;

fout << "User has returned to last location" << endl;

system("pause");

break;

case 23: //Reached destination

if (Taxi.getEngineOn())

{

cout << "Invalid input: Cannot get out of vehicle if engine is on" << endl;

}

else {

cout << "What location is this? Type 2 for Airport. Type 3 for Marina: ";

cin >> userLocation;

if (userLocation == 2)

{

location.push\_back(2);

}

else if (userLocation == 3)

{

location.push\_back(3);

}

else {

cout << "Invalid input: Only values acceptable are 2 or 3" << endl;

}

fout << "User has ended this trip" << endl;

Taxi.addMileage(20);

}

system("pause");

break;

case 0: //Quit

cout << "Sorry to see you go" << endl;

endOfMenu = true;

system("pause");

break;

};

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

if (userInput == 0)

{

break;

}

} while (userInput != 23);

}

else if (userMenu == 2) //If user selected Truck

{

vehicle.push\_back(3);

do {

system("CLS");

cout << "You are driving Truck" << endl;

printTruckMenu();

cin >> userInput;

if (location.at(location.size() - 2) == 1) //Tank Truck

{

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

switch (userInput) {

case 1: //Set age of Truck

cout << "What is the age of truck now?: ";

cin >> userSelection;

if (userSelection < TankTruck.getAge())

{

cout << "Invalid input. Age cannot be less than last one" << endl;

system("pause");

}

else if (userSelection == TankTruck.getAge())

{

cout << "The age you entered is the same age stored earlier" << endl;

system("pause");

}

else {

TankTruck.setAge(userSelection);

cout << "Age stored successfully" << endl;

fout << "Age is stored as " << userSelection << endl;

system("pause");

}

break;

case 2: //Set price of Truck

cout << "What is the price of your truck now?: ";

cin >> userSelection;

if (userSelection > TankTruck.getPrice())

{

cout << "Invalid input. Price cannot be more than last one" << endl;

system("pause");

}

else if (userSelection == TankTruck.getPrice())

{

cout << "The price you enteres it the same price stored earlier" << endl;

system("pause");

}

else if (userSelection < 0)

{

cout << "Invalid input: Price cannot be negative" << endl;

system("Pause");

}

else {

TankTruck.setPrice(userSelection);

cout << "Price stored successfully" << endl;

fout << "Price is stored as " << userSelection << endl;

system("pause");

}

break;

case 3: //Check balance of user

if (vehicle.at(0) == 1)

{

cout << "Your balance is " << Lambo.getBalance() << endl;

}

else if (vehicle.at(0) == 4)

{

cout << "Your balance is " << TankTruck.getBalance() << endl;

}

system("pause");

break;

case 4: //Turn engine on

if (TankTruck.getEngineOn())

{

cout << "Invalid input: Cannot turn engine on while it is already on" << endl;

}

else {

TankTruck.turnEngineOn();

cout << "Engine is now on" << endl;

fout << "User turned on the engine" << endl;

}

system("pause");

break;

case 5: //Turn engine off

if (TankTruck.getEngineOn() == false || TankTruck.getSpeed() != 0 || TankTruck.getAirOn() == true || TankTruck.getLightsOn())

{

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn engine off while it is already off" << endl;

}

if (TankTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot turn engine off while vehicle is moving" << endl;

}

if (TankTruck.getAirOn() == true)

{

cout << "Invalid input: Cannot turn engine off while A/C is on" << endl;

}

if (TankTruck.getLightsOn())

{

cout << "Invalid input: Cannot turn engine off while lights are on" << endl;

}

}

else {

TankTruck.turnEngineOff();

cout << "Engine is now off" << endl;

fout << "User turned off the engine" << endl;

}

system("pause");

break;

case 6: //Turn lights on

if (TankTruck.getLightsOn() || TankTruck.getEngineOn() == false)

{

if (TankTruck.getLightsOn())

{

cout << "Invalid input: Cannot turn lights on if they are already on" << endl;

}

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn lights on if engine is off" << endl;

}

}

else {

TankTruck.turnLightsOn();

cout << "Lights are now on" << endl;

fout << "User turned on the lights" << endl;

}

system("pause");

break;

case 7: //Turn lights off

if (TankTruck.getEngineOn() == false || TankTruck.getLightsOn() == false)

{

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn turn lights off while engine is off" << endl;

}

if (TankTruck.getLightsOn() == false)

{

cout << "Invalid input: Cannot turn lights off while they are already off" << endl;

}

}

else {

TankTruck.turnLightsOff();

cout << "Lights are now off" << endl;

fout << "User turned ooff the ligths" << endl;

}

system("pause");

break;

case 8: //Add Passengers

if (TankTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot add passengers while vehicle is moving" << endl;

}

else {

cout << "How many passengers do you want to add? ";

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot add 0 passengers" << endl;

}

else if (TankTruck.getNumPass() + numPass > 1)

{

cout << "Invlalid input: Cannot add passengers to have total of more than 1" << endl;

}

else {

TankTruck.addPass(numPass);

cout << "Passengers added successfully" << endl;

fout << "User added " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 9: //Kick Passengers

if (TankTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot kick passengers while vehicle is moving\nThey will die" << endl;

}

else {

cout << "How many passengers do you want to kick out" << endl;

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot kick 0 passengers" << endl;

}

else if (numPass > TankTruck.getNumPass())

{

cout << "Invalid input: Cannot kick more passengers than there are" << endl;

}

else {

TankTruck.kickPass(numPass);

cout << "Passengers kicked out successfully" << endl;

fout << "User kicked out " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 10: //Check mileage

cout << "You car's mileage is: " << TankTruck.getMileage() << " miles" << endl;

system("pause");

break;

case 11: //Set coefficient of friction

cout << "What is the current coefficient of friction" << endl;

cin >> coFriction;

if (coFriction < 0)

{

cout << "Invalid input: Cannot have negative coefficient of friction" << endl;

}

else if (coFriction > 1)

{

cout << "Invalid input: Cannot have coefficient of friction more than 1" << endl;

}

else {

TankTruck.setCoFriction(coFriction);

cout << "Coefficient is updated successfully" << endl;

fout << "Coefficient is updated to " << coFriction << endl;

}

system("pause");

break;

case 12: //Check friction

cout << "Current friction on your car is: " << TankTruck.getFriction();

system("pause");

break;

case 13: //Turn air on

if (TankTruck.getAirOn() || TankTruck.getEngineOn() == false)

{

if (TankTruck.getAirOn())

{

cout << "Invalid input: Cannot turn A/C on while it is already on" << endl;

}

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn air on if engine is off" << endl;

}

}

else {

TankTruck.turnAirOn();

cout << "Air is now on" << endl;

fout << "User turned on the air" << endl;

}

system("pause");

break;

case 14: //Turn air off

if (TankTruck.getAirOn() == false)

{

cout << "Invalid input: Cannot turn air off if it is already off" << endl;

}

else {

TankTruck.turnAirOff();

cout << "Air is now off" << endl;

fout << "User turned off the air" << endl;

}

system("pause");

break;

case 15: //Change diesel type status

cout << "Type 1 if it IS a diesel type and type 0 if it IS NOT a diesel type: ";

cin >> tempStat;

if (tempStat < 0 || tempStat > 1)

{

cout << "Invalid input: Only accepted values are 0 and 1" << endl;

}

else {

TankTruck.setDieselTypeStatus(tempStat);

cout << "Diesel type status updated" << endl;

if (tempStat)

{

fout << "Diesel type status updated to true" << endl;

}

else {

fout << "Diesel type status updated to false" << endl;

}

}

system("pause");

break;

case 16: //Load cargo

cout << "How much do you want to load?: " << endl;

cin >> userCapacity;

if (userCapacity < 0 || userCapacity == 0 || userCapacity > 100 || TankTruck.getCapacity() + userCapacity > 100 || TankTruck.getSpeed() != 0)

{

if (100 - TankTruck.getCapacity() < userCapacity)

{

cout << "Invalid input: Cannot add to end up with more than 100 cargo" << endl;

}

if (userCapacity < 0)

{

cout << "Invalid input: Cannot add 0 cargo" << endl;

}

if (userCapacity == 0)

{

cout << "Invalid input: Cannot add 0 cargo" << endl;

}

if (TankTruck.getCapacity() > 100)

{

cout << "Invalid input: Cannot add more than 100 cargo" << endl;

}

if (TankTruck.getCapacity() + userCapacity > 100)

{

cout << "Invalid input: Cannot add cargo to have total of more than 100 cargo" << endl;

}

if (TankTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot add cargo while vehicle is moving" << endl;

}

}

else {

TankTruck.loadCargo(userCapacity);

cout << "Cargo loaded successfully" << endl;

fout << "User added " << userCapacity << " cargo" << endl;

}

system("pause");

break;

case 17: //Unload Cargo

cout << "How much do you want to unload?: " << endl;

cin >> userCapacity;

if (userCapacity > TankTruck.getCapacity() || userCapacity < 0 || userCapacity == 0 || userCapacity > 100 || TankTruck.getSpeed() != 0)

{

if (userCapacity > TankTruck.getCapacity())

{

cout << "Invalid input: Cannot unload to end up with negative cargo" << endl;

}

if (userCapacity < 0)

{

cout << "Invalid input: Cannot have a negative cargo" << endl;

}

if (userCapacity == 0)

{

cout << "Invalid input: Cannot unload 0 cargo" << endl;

}

if (userCapacity > 100)

{

cout << "Invalid input: Cannot unload more than 100 cargo" << endl;

}

if (TankTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot unload cargo while vehicle is moving" << endl;

}

}

else {

TankTruck.loadCargo(userCapacity);

cout << "Cargo loaded successfully" << endl;

fout << "User took out " << userCapacity << " cargo" << endl;

}

system("pause");

break;

case 18: //Check cargo

cout << "Current cargo is " << TankTruck.getCapacity() << endl;

system("pause");

break;

case 19: //Turn right

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (TankTruck.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else if (TankTruck.getSpeed() > 10)

{

cout << "Invalid input: Cannot turn if speed is greater than 10" << endl;

}

else {

TankTruck.turnRight(turnsTripOne);

cout << "You have turned right" << endl;

fout << "User turned right " << endl;

TankTruck.addMileage(10);

}

system("pause");

break;

case 20: //Turn left

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (TankTruck.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else if (TankTruck.getSpeed() > 10)

{

cout << "Invalid input: Cannot turn if speed is greater than 10" << endl;

}

else {

TankTruck.turnLeft(turnsTripOne);

cout << "You have turned left" << endl;

fout << "User turned left" << endl;

TankTruck.addMileage(10);

}

system("pause");

break;

case 21: //Accelerate

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot accelerate if engine is off" << endl;

}

else if (TankTruck.getSpeed() == 45)

{

cout << "Invalid input: Cannot accelerate to more than 45 mph" << endl;

}

else {

TankTruck.accelerate();

cout << "Speed is now increased to " << TankTruck.getSpeed() << endl;

fout << "User accelerated vehicle to " << TankTruck.getSpeed() << endl;

TankTruck.addMileage(10);

}

system("pause");

break;

case 22: //Decelerate

if (TankTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot decelerate if engine is off" << endl;

}

else if (TankTruck.getSpeed() == 0)

{

cout << "Invalid input: Cannot decelerate to less than 0 mph" << endl;

}

else {

TankTruck.decelerate();

cout << "Speed is now decreased to " << TankTruck.getSpeed() << endl;

fout << "User decelerated vehicle to " << TankTruck.getSpeed() << endl;

}

system("pause");

break;

case 23: //Return to last location

for (int i = 0; i < turnsTripTwo.size(); i++)

{

if (turnsTripTwo.at(i) == 1)

{

turnsTripTwo[i] = 3;

}

else {

turnsTripTwo[i] = 1;

}

}

reverse(turnsTripTwo.begin(), turnsTripTwo.end());

for (int i = 0; i < turnsTripTwo.size(); i++)

{

if (turnsTripTwo.at(i) == 1)

{

cout << "Turn left..." << endl;

fout << "User turned left" << endl;

TankTruck.addMileage(10);

}

else {

cout << "Turn right..." << endl;

fout << "User turned right" << endl;

TankTruck.addMileage(10);

}

}

location[location.size() - 1] = NULL; //Deletes last location

cout << "You are at home now" << endl;

fout << "User has returned to last location" << endl;

TankTruck.addMileage(20);

system("pause");

break;

case 24: //Reached destination

if (TankTruck.getEngineOn())

{

cout << "Invalid input: Cannot get out of vehicle if engine is on" << endl;

}

else {

cout << "What location is this? Type 2 for Airport. Type 3 for Marina: ";

cin >> userLocation;

if (userLocation == 2)

{

location.push\_back(2);

}

else if (userLocation == 3)

{

location.push\_back(3);

}

else {

cout << "Invalid input: Only values acceptable are 2 or 3" << endl;

}

fout << "User has reached destionation" << endl;

TankTruck.addMileage(20);

}

system("pause");

break;

case 0: //Quit

cout << "Sorry to see you go" << endl;

endOfMenu = true;

system("pause");

break;

};

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

else //HaulTruck

{

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

switch (userInput) {

case 1: //Set age of Truck

cout << "What is the age of truck now?: ";

cin >> userSelection;

if (userSelection < HaulTruck.getAge())

{

cout << "Invalid input. Age cannot be less than last one" << endl;

system("pause");

}

else if (userSelection == HaulTruck.getAge())

{

cout << "The age you entered is the same age stored earlier" << endl;

system("pause");

}

else {

HaulTruck.setAge(userSelection);

cout << "Age stored successfully" << endl;

fout << "Age is stored as " << userSelection << endl;

system("pause");

}

break;

case 2: //Set price of Truck

cout << "What is the price of your truck now?: ";

cin >> userSelection;

if (userSelection > HaulTruck.getPrice())

{

cout << "Invalid input. Price cannot be more than last one" << endl;

system("pause");

}

else if (userSelection == HaulTruck.getPrice())

{

cout << "The price you enteres it the same price stored earlier" << endl;

system("pause");

}

else if (userSelection < 0)

{

cout << "Invalid input: Price cannot be negative" << endl;

system("Pause");

}

else {

HaulTruck.setPrice(userSelection);

cout << "Price stored successfully" << endl;

fout << "Price is stored as " << userSelection << endl;

system("pause");

}

break;

case 3: //Check balance of user

if (vehicle.at(0) == 1)

{

cout << "Your balance is " << Lambo.getBalance() << endl;

}

else if (vehicle.at(0) == 4)

{

cout << "Your balance is " << TankTruck.getBalance() << endl;

}

system("pause");

break;

case 4: //Turn engine on

if (HaulTruck.getEngineOn())

{

cout << "Invalid input: Cannot turn engine on while it is already on" << endl;

}

else {

HaulTruck.turnEngineOn();

cout << "Engine is now on" << endl;

fout << "User turned on the engine" << endl;

}

system("pause");

break;

case 5: //Turn engine off

if (HaulTruck.getEngineOn() == false || HaulTruck.getSpeed() != 0 || HaulTruck.getAirOn() == true || HaulTruck.getLightsOn())

{

if (HaulTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn engine off while it is already off" << endl;

}

if (HaulTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot turn engine off while vehicle is moving" << endl;

}

if (HaulTruck.getAirOn() == true)

{

cout << "Invalid input: Cannot turn engine off while A/C is on" << endl;

}

if (HaulTruck.getLightsOn())

{

cout << "Invalid input: Cannot turn engine off while lights are on" << endl;

}

}

else {

HaulTruck.turnEngineOff();

cout << "Engine is now off" << endl;

fout << "User turned off the engine" << endl;

}

system("pause");

break;

case 6: //Turn lights on

if (HaulTruck.getLightsOn() || HaulTruck.getEngineOn() == false)

{

if (HaulTruck.getLightsOn())

{

cout << "Invalid input: Cannot turn lights on if they are already on" << endl;

}

if (HaulTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn lights on if engine is off" << endl;

}

}

else {

HaulTruck.turnLightsOn();

cout << "Lights are now on" << endl;

fout << "User turned on the lights" << endl;

}

system("pause");

break;

case 7: //Turn lights off

if (HaulTruck.getEngineOn() == false || HaulTruck.getLightsOn() == false)

{

if (HaulTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn turn lights off while engine is off" << endl;

}

if (HaulTruck.getLightsOn() == false)

{

cout << "Invalid input: Cannot turn lights off while they are already off" << endl;

}

}

else {

HaulTruck.turnLightsOff();

cout << "Lights are now off" << endl;

fout << "User turned off the lights" << endl;

}

system("pause");

break;

case 8: //Add Passengers

if (HaulTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot add passengers while vehicle is moving" << endl;

}

else {

cout << "How many passengers do you want to add? ";

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot add 0 passengers" << endl;

}

else if (HaulTruck.getNumPass() + numPass > 8)

{

cout << "Invlalid input: Cannot add passengers to have total of more than 1" << endl;

}

else {

HaulTruck.addPass(numPass);

cout << "Passengers added successfully" << endl;

fout << "User added " << numPass << " passengers " << endl;

}

}

system("pause");

break;

case 9: //Kick Passengers

if (HaulTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot kick passengers while vehicle is moving\nThey will die" << endl;

}

else {

cout << "How many passengers do you want to kick out" << endl;

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot kick 0 passengers" << endl;

}

else if (numPass > HaulTruck.getNumPass())

{

cout << "Invalid input: Cannot kick more passengers than there are" << endl;

}

else {

HaulTruck.kickPass(numPass);

cout << "Passengers kicked out successfully" << endl;

fout << "User kicked out " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 10: //Check mileage

cout << "You car's mileage is: " << HaulTruck.getMileage() << " miles" << endl;

system("pause");

break;

case 11: //Set coefficient of friction

cout << "What is the current coefficient of friction" << endl;

cin >> coFriction;

if (coFriction < 0)

{

cout << "Invalid input: Cannot have negative coefficient of friction" << endl;

}

else if (coFriction > 1)

{

cout << "Invalid input: Cannot have coefficient of friction more than 1" << endl;

}

else {

HaulTruck.setCoFriction(coFriction);

cout << "Coefficient is updated successfully" << endl;

fout << "Coefficeient is updated to " << coFriction << endl;

}

system("pause");

break;

case 12: //Check friction

cout << "Current friction on your car is: " << HaulTruck.getFriction();

system("pause");

break;

case 13: //Turn air on

if (HaulTruck.getAirOn() || HaulTruck.getEngineOn() == false)

{

if (HaulTruck.getAirOn())

{

cout << "Invalid input: Cannot turn A/C on while it is already on" << endl;

}

if (HaulTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn air on if engine is off" << endl;

}

}

else {

HaulTruck.turnAirOn();

cout << "Air is now on" << endl;

fout << "User turned on the air" << endl;

}

system("pause");

break;

case 14: //Turn air off

if (HaulTruck.getAirOn() == false)

{

cout << "Invalid input: Cannot turn air off if it is already off" << endl;

}

else {

HaulTruck.turnAirOff();

cout << "Air is now off" << endl;

fout << "User turned off the air" << endl;

}

system("pause");

break;

case 15: //Change diesel type status

cout << "Type 1 if it IS a diesel type and type 0 if it IS NOT a diesel type: ";

cin >> tempStat;

if (tempStat < 0 || tempStat > 1)

{

cout << "Invalid input: Only accepted values are 0 and 1" << endl;

}

else {

HaulTruck.setDieselTypeStatus(tempStat);

cout << "Diesel type status updated" << endl;

if (tempStat)

{

fout << "Diesel type status updated to true" << endl;

}

else {

fout << "Diesel type status updated to false " << endl;

}

}

system("pause");

break;

case 16: //Load cargo

cout << "How much do you want to load?: " << endl;

cin >> userCapacity;

if (userCapacity < 0 || userCapacity == 0 || userCapacity > 100 || HaulTruck.getCapacity() + userCapacity > 100 || HaulTruck.getSpeed() != 0)

{

if (100 - HaulTruck.getCapacity() < userCapacity)

{

cout << "Invalid input: Cannot add to end up with more than 100 cargo" << endl;

}

if (userCapacity < 0)

{

cout << "Invalid input: Cannot add 0 cargo" << endl;

}

if (userCapacity == 0)

{

cout << "Invalid input: Cannot add 0 cargo" << endl;

}

if (HaulTruck.getCapacity() > 100)

{

cout << "Invalid input: Cannot add more than 100 cargo" << endl;

}

if (HaulTruck.getCapacity() + userCapacity > 100)

{

cout << "Invalid input: Cannot add cargo to have total of more than 100 cargo" << endl;

}

if (HaulTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot add cargo while vehicle is moving" << endl;

}

}

else {

HaulTruck.loadCargo(userCapacity);

cout << "Cargo loaded successfully" << endl;

fout << "User added " << userCapacity << " cargo" << endl;

}

system("pause");

break;

case 17: //Unload Cargo

cout << "How much do you want to unload?: " << endl;

cin >> userCapacity;

if (userCapacity > HaulTruck.getCapacity() || userCapacity < 0 || userCapacity == 0 || userCapacity > 100 || HaulTruck.getSpeed() != 0)

{

if (userCapacity > HaulTruck.getCapacity())

{

cout << "Invalid input: Cannot unload to end up with negative cargo" << endl;

}

if (userCapacity < 0)

{

cout << "Invalid input: Cannot have a negative cargo" << endl;

}

if (userCapacity == 0)

{

cout << "Invalid input: Cannot unload 0 cargo" << endl;

}

if (userCapacity > 100)

{

cout << "Invalid input: Cannot unload more than 100 cargo" << endl;

}

if (HaulTruck.getSpeed() != 0)

{

cout << "Invalid input: Cannot unload cargo while vehicle is moving" << endl;

}

}

else {

HaulTruck.loadCargo(userCapacity);

cout << "Cargo unloaded successfully" << endl;

fout << "User took out " << userCapacity << " cargo" << endl;

}

system("pause");

break;

case 18: //Check cargo

cout << "Current cargo is " << HaulTruck.getCapacity() << endl;

system("pause");

break;

case 19: //Turn right

if (HaulTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (HaulTruck.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else if (HaulTruck.getSpeed() > 10)

{

cout << "Invalid input: Cannot turn if speed is greater than 10" << endl;

}

else {

HaulTruck.turnRight(turnsTripOne);

cout << "You have turned right" << endl;

fout << "User turned right " << endl;

HaulTruck.addMileage(10);

}

system("pause");

break;

case 20: //Turn left

if (HaulTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (HaulTruck.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else if (HaulTruck.getSpeed() > 10)

{

cout << "Invalid input: Cannot turn if speed is greater than 10" << endl;

}

else {

HaulTruck.turnLeft(turnsTripOne);

cout << "You have turned left" << endl;

fout << "User turned left " << endl;

HaulTruck.addMileage(10);

}

system("pause");

break;

case 21: //Accelerate

if (HaulTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot accelerate if engine is off" << endl;

}

else if (HaulTruck.getSpeed() == 45)

{

cout << "Invalid input: Cannot accelerate to more than 45 mph" << endl;

}

else {

HaulTruck.accelerate();

cout << "Speed is now increased to " << HaulTruck.getSpeed() << endl;

fout << "User increased speed to " << HaulTruck.getSpeed() << endl;

HaulTruck.addMileage(10);

}

system("pause");

break;

case 22: //Decelerate

if (HaulTruck.getEngineOn() == false)

{

cout << "Invalid input: Cannot decelerate if engine is off" << endl;

}

else if (HaulTruck.getSpeed() == 0)

{

cout << "Invalid input: Cannot decelerate to less than 0 mph" << endl;

}

else {

HaulTruck.decelerate();

cout << "Speed is now decreased to " << HaulTruck.getSpeed() << endl;

fout << "Speed is decreased to " << HaulTruck.getSpeed() << endl;

HaulTruck.addMileage(10);

}

system("pause");

break;

case 23: //Return to last location

for (int i = 0; i < turnsTripTwo.size(); i++)

{

if (turnsTripTwo.at(i) == 1)

{

turnsTripTwo[i] = 3;

}

else {

turnsTripTwo[i] = 1;

}

}

reverse(turnsTripTwo.begin(), turnsTripTwo.end());

for (int i = 0; i < turnsTripTwo.size(); i++)

{

if (turnsTripTwo.at(i) == 1)

{

cout << "Turn left..." << endl;

fout << "User turned left " << endl;

HaulTruck.addMileage(10);

}

else {

cout << "Turn right..." << endl;

fout << "User turned right " << endl;

HaulTruck.addMileage(10);

}

}

location[location.size() - 1] = NULL; //Deletes last location

cout << "You are at home now" << endl;

fout << "User has returned to last lcoation" << endl;

HaulTruck.addMileage(20);

system("pause");

break;

case 24: //Reached destination

if (HaulTruck.getEngineOn())

{

cout << "Invalid input: Cannot get out of vehicle if engine is on" << endl;

}

else {

cout << "What location is this? Type 2 for Airport. Type 3 for Marina: ";

cin >> userLocation;

if (userLocation == 2)

{

location.push\_back(2);

}

else if (userLocation == 3)

{

location.push\_back(3);

}

else {

cout << "Invalid input: Only values acceptable are 2 or 3" << endl;

}

fout << "User has ended this trip" << endl;

HaulTruck.addMileage(20);

}

system("pause");

break;

case 0: //Quit

cout << "Sorry to see you go" << endl;

endOfMenu = true;

system("pause");

break;

};

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

if (userInput == 0)

{

break;

}

} while (userInput != 24);

}

else if (userMenu == 3) //If user selected Boat

{

vehicle.push\_back(3);

do {

system("CLS");

cout << "You are driving Boat" << endl;

printBoatMenu();

cin >> userInput;

if (location.at(location.size() - 2) == 1) //Hovercraft

{

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

switch (userInput)

{

case 1: //Set age

cout << "What is the age of plane now?: ";

cin >> userSelection;

if (userSelection < Hovercraft.getAge())

{

cout << "Invalid input. Age cannot be less than last one" << endl;

system("pause");

}

else if (userSelection == Hovercraft.getAge())

{

cout << "The age you entered is the same age stored earlier" << endl;

system("pause");

}

else {

Hovercraft.setAge(userSelection);

cout << "Age stored successfully" << endl;

fout << "Age stored as " << userSelection << endl;

system("pause");

}

break;

case 2: //Set price

cout << "What is the price of your plane now?: ";

cin >> userSelection;

if (userSelection > Hovercraft.getPrice())

{

cout << "Invalid input. Price cannot be more than last one" << endl;

system("pause");

}

else if (userSelection == Hovercraft.getPrice())

{

cout << "The price you enteres it the same price stored earlier" << endl;

system("pause");

}

else if (userSelection < 0)

{

cout << "Invalid input: Price cannot be negative" << endl;

system("Pause");

}

else {

Hovercraft.setPrice(userSelection);

cout << "Price stored successfully" << endl;

fout << "Price stored as " << userSelection << endl;

system("pause");

}

break;

case 3: //Check balance

if (vehicle.at(0) == 1)

{

cout << "Your balance is " << Lambo.getBalance() << endl; //Don't change

}

else if (vehicle.at(0) == 4)

{

cout << "Your balance is " << TankTruck.getBalance() << endl; //Don't change

}

system("pause");

break;

case 4: //Turn engine on

if (Hovercraft.getEngineOn())

{

cout << "Invalid input: Cannot turn engine on while it is already on" << endl;

}

else {

Hovercraft.turnEngineOn();

cout << "Engine is now on" << endl;

fout << "User turns on the engine" << endl;

}

system("pause");

break;

case 5: //Turn engine off

if (Hovercraft.getEngineOn() == false || Hovercraft.getSpeed() != 0 || Hovercraft.getLightsOn() || Hovercraft.getLaunch())

{

if (Hovercraft.getLaunch())

{

cout << "Invalid input: Cannot turn engine off while boat is not docked" << endl;

}

if (Hovercraft.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn engine off while it is already off" << endl;

}

if (Hovercraft.getSpeed() != 0)

{

cout << "Invalid input: Cannot turn engine off while vehicle is moving" << endl;

}

if (Hovercraft.getLightsOn())

{

cout << "Invalid input: Cannot turn engine off while lights are on" << endl;

}

}

else {

Hovercraft.turnEngineOff();

cout << "Engine is now off" << endl;

fout << "User turned off the engine " << endl;

}

system("pause");

break;

case 6: //Turn lights on

if (Hovercraft.getLightsOn() || Hovercraft.getEngineOn() == false)

{

if (Hovercraft.getLightsOn())

{

cout << "Invalid input: Cannot turn lights on if they are already on" << endl;

}

if (Hovercraft.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn lights on if engine is off" << endl;

}

}

else {

Hovercraft.turnLightsOn();

cout << "Lights are now on" << endl;

fout << "User turned on the lights " << endl;

}

system("pause");

break;

case 7: //Turn lights off

if (Hovercraft.getEngineOn() == false || Hovercraft.getLightsOn() == false)

{

if (Hovercraft.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn turn lights off while engine is off" << endl;

}

if (Hovercraft.getLightsOn() == false)

{

cout << "Invalid input: Cannot turn lights off while they are already off" << endl;

}

}

else {

Hovercraft.turnLightsOff();

cout << "Lights are now off" << endl;

fout << "User turned off the ligths " << endl;

}

system("pause");

break;

case 8: //Add passengers

if (Hovercraft.getSpeed() != 0)

{

cout << "Invalid input: Cannot add passengers while vehicle is moving" << endl;

}

else {

cout << "How many passengers do you want to add? ";

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot add 0 passengers" << endl;

}

else if (Hovercraft.getNumPass() + numPass > 8)

{

cout << "Invlalid input: Cannot add passengers to have total of more than 8" << endl;

}

else {

Hovercraft.addPass(numPass);

cout << "Passengers added successfully" << endl;

fout << "User adds " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 9: //Kick passengers

if (Hovercraft.getSpeed() != 0)

{

cout << "Invalid input: Cannot kick passengers while vehicle is moving\nThey will drown" << endl;

}

else {

cout << "How many passengers do you want to kick out" << endl;

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot kick 0 passengers" << endl;

}

else if (numPass > Hovercraft.getNumPass())

{

cout << "Invalid input: Cannot kick more passengers than there are" << endl;

}

else {

Hovercraft.kickPass(numPass);

cout << "Passengers kicked out successfully" << endl;

fout << "User kicked out " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 10: //Check propeller level

cout << "Your propeller level is " << Hovercraft.getPropellerLevel() << endl;

system("pause");

break;

case 11: //Change propeller level

if (Hovercraft.getEngineOn() == false)

{

cout << "Invalid input. Cannot change propeller level is engine is off" << endl;

}

else

{

cout << "What level would you like to set the propeller to?: ";

cin >> userPropeller;

if (userPropeller == Hovercraft.getPropellerLevel())

{

cout << "Invalid input. Cannot set propeller level to be the same as before" << endl;

}

else if (userPropeller == 0)

{

if (Hovercraft.getDock())

{

Hovercraft.setPropellerLevel(0);

cout << "Propeller level is set successfully to 0. Boat will no longer move" << endl;

fout << "User changed propeller level to " << userPropeller << endl;

}

else {

cout << "Invalid input: Cannot set propeller level to 0 while boat is not docked" << endl;

}

}

else if (userPropeller > 5)

{

cout << "Invalid input: Cannot set propeller level to be more than 5. That would be too deep" << endl;

}

else if (userPropeller < 0)

{

cout << "Invalid input: Cannot set propeller level to a negative value" << endl;

}

else {

Hovercraft.setPropellerLevel(userPropeller);

cout << "Propeller level set successfully" << endl;

fout << "User changed propeller level to " << userPropeller << endl;

}

}

system("pause");

break;

case 12: //Dock the boat

if (Hovercraft.getDock())

{

cout << "Invalid input: Cannot dock the boat if it is already docked" << endl;

}

else if (Hovercraft.getSpeed() > 10)

{

cout << "Invalid input: Cannot dock the boat if speed is more than 10 mph" << endl;

}

else {

Hovercraft.setDock(true);

Hovercraft.setLaunch(false);

cout << "Boat is now docked" << endl;

fout << "User docked the boat" << endl;

}

system("pause");

break;

case 13: //Launch the boat

if (Hovercraft.getLaunch())

{

cout << "Invalid input: Cannot launch the boat if it is already launched" << endl;

}

else if (Hovercraft.getSpeed() == 0)

{

cout << "Invalid input: Cannot launch the boat if speed is 0 mph" << endl;

}

else {

Hovercraft.setLaunch(true);

Hovercraft.setDock(false);

cout << "Boat is now launched" << endl;

fout << "User launched the boat" << endl;

}

system("pause");

break;

case 14: //Turn right

if (Hovercraft.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (Hovercraft.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else {

Hovercraft.turnRight(turnsTripThree);

cout << "You have turned right" << endl;

fout << "User turned right" << endl;

}

system("pause");

break;

case 15: //Turn left

if (Hovercraft.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (Hovercraft.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else {

Hovercraft.turnLeft(turnsTripThree);

cout << "You have turned left" << endl;

fout << "User turned left" << endl;

}

system("pause");

break;

case 16: //Accelerate

if (Hovercraft.getEngineOn() == false)

{

cout << "Invalid input: Cannot accelerate if engine is off" << endl;

}

else if (Hovercraft.getPropellerLevel() == 0)

{

cout << "Invalid input: Cannot accelerate if propeller level is 0" << endl;

}

else

{

if (Hovercraft.getDock() && Hovercraft.getSpeed() == 10)

{

cout << "Invalid input: Cannot accelerate to more than 10 mph while boat is docked" << endl;

}

else if (Hovercraft.getSpeed() == 45)

{

cout << "Invalid input: Cannot go above 45 mph" << endl;

}

else {

Hovercraft.accelerate();

cout << "Speed is now increased to " << Hovercraft.getSpeed() << endl;

fout << "User increased speed to " << Hovercraft.getSpeed() << endl;

}

}

system("pause");

break;

case 17: //Decelerate

if (Hovercraft.getEngineOn() == false)

{

cout << "Invalid input: Cannot decelerate if engine is off" << endl;

}

else if (Hovercraft.getPropellerLevel() == 0)

{

cout << "Invalid input: Cannot decelerate if propeller level is 0" << endl;

}

else if (Hovercraft.getSpeed() == 0)

{

cout << "Invalid input: Cannot decrease speed to a negative value" << endl;

}

else {

Hovercraft.decelerate();

cout << "Speed is now decreased to " << Hovercraft.getSpeed() << endl;

fout << "User decreased speed to " << Hovercraft.getSpeed() << endl;

}

system("pause");

break;

case 18: //Return to last location

for (int i = 0; i < turnsTripThree.size(); i++)

{

if (turnsTripThree.at(i) == 1)

{

turnsTripThree[i] = 3;

}

else {

turnsTripThree[i] = 1;

}

}

reverse(turnsTripThree.begin(), turnsTripThree.end());

for (int i = 0; i < turnsTripThree.size(); i++)

{

if (turnsTripThree.at(i) == 1)

{

cout << "Turn left..." << endl;

fout << "User turned left" << endl;

}

else {

cout << "Turn right..." << endl;

fout << "User turned right" << endl;

}

}

location[location.size() - 1] = NULL; //Deletes last location

fout << "User has returend to last location" << endl;

system("pause");

break;

case 19: //Reached destination

if (Hovercraft.getEngineOn())

{

cout << "Invalid input: Cannot get out of vehicle if engine is on" << endl;

}

else {

location.push\_back(3);

fout << "User has ended this trip" << endl;

}

system("pause");

break;

case 0: //Quit

cout << "Sorry to see you go" << endl;

endOfMenu = true;

system("pause");

break;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

else //Yacht

{

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

switch (userInput)

{

case 1: //Set age

cout << "What is the age of plane now?: ";

cin >> userSelection;

if (userSelection < Yacht.getAge())

{

cout << "Invalid input. Age cannot be less than last one" << endl;

system("pause");

}

else if (userSelection == Yacht.getAge())

{

cout << "The age you entered is the same age stored earlier" << endl;

system("pause");

}

else {

Yacht.setAge(userSelection);

cout << "Age stored successfully" << endl;

fout << "Age stored as " << userSelection << endl;

system("pause");

}

break;

case 2: //Set price

cout << "What is the price of your plane now?: ";

cin >> userSelection;

if (userSelection > Yacht.getPrice())

{

cout << "Invalid input. Price cannot be more than last one" << endl;

system("pause");

}

else if (userSelection == Yacht.getPrice())

{

cout << "The price you enteres it the same price stored earlier" << endl;

system("pause");

}

else if (userSelection < 0)

{

cout << "Invalid input: Price cannot be negative" << endl;

system("Pause");

}

else {

Yacht.setPrice(userSelection);

cout << "Price stored successfully" << endl;

fout << "Price stored as " << userSelection << endl;

system("pause");

}

break;

case 3: //Check balance

if (vehicle.at(0) == 1)

{

cout << "Your balance is " << Lambo.getBalance() << endl; //Don't change

}

else if (vehicle.at(0) == 4)

{

cout << "Your balance is " << TankTruck.getBalance() << endl; //Don't change

}

system("pause");

break;

case 4: //Turn engine on

if (Yacht.getEngineOn())

{

cout << "Invalid input: Cannot turn engine on while it is already on" << endl;

}

else {

Yacht.turnEngineOn();

cout << "Engine is now on" << endl;

fout << "User turned on the engine" << endl;

}

system("pause");

break;

case 5: //Turn engine off

if (Yacht.getEngineOn() == false || Yacht.getSpeed() != 0 || Yacht.getLightsOn() || Yacht.getLaunch())

{

if (Yacht.getLaunch())

{

cout << "Invalid input: Cannot turn engine off while boat is not docked" << endl;

}

if (Yacht.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn engine off while it is already off" << endl;

}

if (Yacht.getSpeed() != 0)

{

cout << "Invalid input: Cannot turn engine off while vehicle is moving" << endl;

}

if (Yacht.getLightsOn())

{

cout << "Invalid input: Cannot turn engine off while lights are on" << endl;

}

}

else {

Yacht.turnEngineOff();

cout << "Engine is now off" << endl;

fout << "User turned off the engine" << endl;

}

system("pause");

break;

case 6: //Turn lights on

if (Yacht.getLightsOn() || Yacht.getEngineOn() == false)

{

if (Yacht.getLightsOn())

{

cout << "Invalid input: Cannot turn lights on if they are already on" << endl;

}

if (Yacht.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn lights on if engine is off" << endl;

}

}

else {

Yacht.turnLightsOn();

cout << "Lights are now on" << endl;

fout << "User turned on the lights" << endl;

}

system("pause");

break;

case 7: //Turn lights off

if (Yacht.getEngineOn() == false || Yacht.getLightsOn() == false)

{

if (Yacht.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn turn lights off while engine is off" << endl;

}

if (Yacht.getLightsOn() == false)

{

cout << "Invalid input: Cannot turn lights off while they are already off" << endl;

}

}

else {

Yacht.turnLightsOff();

cout << "Lights are now off" << endl;

fout << "User turned off the lights" << endl;

}

system("pause");

break;

case 8: //Add passengers

if (Yacht.getSpeed() != 0)

{

cout << "Invalid input: Cannot add passengers while vehicle is moving" << endl;

}

else {

cout << "How many passengers do you want to add? ";

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot add 0 passengers" << endl;

}

else if (Yacht.getNumPass() + numPass > 8)

{

cout << "Invlalid input: Cannot add passengers to have total of more than 8" << endl;

}

else {

Yacht.addPass(numPass);

cout << "Passengers added successfully" << endl;

fout << "User adds " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 9: //Kick passengers

if (Yacht.getSpeed() != 0)

{

cout << "Invalid input: Cannot kick passengers while vehicle is moving\nThey will drown" << endl;

}

else {

cout << "How many passengers do you want to kick out" << endl;

cin >> numPass;

if (numPass == 0)

{

cout << "Invalid input: Cannot kick 0 passengers" << endl;

}

else if (numPass > Yacht.getNumPass())

{

cout << "Invalid input: Cannot kick more passengers than there are" << endl;

}

else {

Yacht.kickPass(numPass);

cout << "Passengers kicked out successfully" << endl;

fout << "User kicks out " << numPass << " passengers" << endl;

}

}

system("pause");

break;

case 10: //Check propeller level

cout << "Your propeller level is " << Yacht.getPropellerLevel() << endl;

system("pause");

break;

case 11: //Change propeller level

if (Yacht.getEngineOn() == false)

{

cout << "Invalid input. Cannot change propeller level is engine is off" << endl;

}

else

{

cout << "What level would you like to set the propeller to?: ";

cin >> userPropeller;

if (userPropeller == Yacht.getPropellerLevel())

{

cout << "Invalid input. Cannot set propeller level to be the same as before" << endl;

}

else if (userPropeller == 0)

{

if (Yacht.getDock())

{

Yacht.setPropellerLevel(0);

cout << "Propeller level is set successfully to 0. Boat will no longer move" << endl;

fout << "User changed propeller level to " << userPropeller << endl;

}

else {

cout << "Invalid input: Cannot set propeller level to 0 while boat is not docked" << endl;

}

}

else if (userPropeller < 0)

{

cout << "Invalid input: Cannot set propeller level to a negative value" << endl;

}

else if (userPropeller > 5)

{

cout << "Invalid input: Cannot set propeller level to be more than 5. That would be too deep" << endl;

}

else {

Yacht.setPropellerLevel(userPropeller);

cout << "Propeller level set successfully" << endl;

fout << "User changed propeller level to " << userPropeller << endl;

}

}

system("pause");

break;

case 12: //Dock the boat

if (Yacht.getDock())

{

cout << "Invalid input: Cannot dock the boat if it is already docked" << endl;

}

else if (Yacht.getSpeed() > 10)

{

cout << "Invalid input: Cannot dock the boat if speed is more than 10 mph" << endl;

}

else {

Yacht.setDock(true);

Yacht.setLaunch(false);

cout << "Boat is now docked" << endl;

fout << "User docked the boad" << endl;

}

system("pause");

break;

case 13: //Launch the boat

if (Yacht.getLaunch())

{

cout << "Invalid input: Cannot launch the boat if it is already launched" << endl;

}

else if (Yacht.getSpeed() == 0)

{

cout << "Invalid input: Cannot launch the boat if speed is 0 mph" << endl;

}

else {

Yacht.setLaunch(true);

Yacht.setDock(false);

cout << "Boat is now launched" << endl;

fout << "User launcehd the boat" << endl;

}

system("pause");

break;

case 14: //Turn right

if (Yacht.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (Yacht.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else {

Yacht.turnRight(turnsTripThree);

cout << "You have turned right" << endl;

fout << "User turned right" << endl;

}

system("pause");

break;

case 15: //Turn left

if (Yacht.getEngineOn() == false)

{

cout << "Invalid input: Cannot turn if engine is off" << endl;

}

else if (Yacht.getSpeed() == 0)

{

cout << "Invalid input: Cannot turn if speed is 0" << endl;

}

else {

Yacht.turnLeft(turnsTripThree);

cout << "You have turned left" << endl;

fout << "User turned left" << endl;

}

system("pause");

break;

case 16: //Accelerate

if (Yacht.getEngineOn() == false)

{

cout << "Invalid input: Cannot accelerate if engine is off" << endl;

}

else if (Hovercraft.getPropellerLevel() == 0)

{

cout << "Invalid input: Cannot accelerate if propeller level is 0" << endl;

}

else

{

if (Yacht.getDock() && Yacht.getSpeed() == 10)

{

cout << "Invalid input: Cannot accelerate to more than 10 mph while boat is docked" << endl;

}

else if (Yacht.getSpeed() == 45)

{

cout << "Invalid input: Cannot go above 45 mph" << endl;

}

else {

Yacht.accelerate();

cout << "Speed is now increased to " << Yacht.getSpeed() << endl;

fout << "User increases speed to " << Yacht.getSpeed() << endl;

}

}

system("pause");

break;

case 17: //Decelerate

if (Yacht.getEngineOn() == false)

{

cout << "Invalid input: Cannot decelerate if engine is off" << endl;

}

else if (Hovercraft.getPropellerLevel() == 0)

{

cout << "Invalid input: Cannot decelerate if propeller level is 0" << endl;

}

else if (Yacht.getSpeed() == 0)

{

cout << "Invalid input: Cannot decrease speed to a negative value" << endl;

}

else {

Yacht.decelerate();

cout << "Speed is now decreased to " << Yacht.getSpeed() << endl;

fout << "User decreases speed to " << Yacht.getSpeed() << endl;

}

system("pause");

break;

case 18: //Return to last location

for (int i = 0; i < turnsTripThree.size(); i++)

{

if (turnsTripThree.at(i) == 1)

{

turnsTripThree[i] = 3;

}

else {

turnsTripThree[i] = 1;

}

}

reverse(turnsTripThree.begin(), turnsTripThree.end());

for (int i = 0; i < turnsTripThree.size(); i++)

{

if (turnsTripThree.at(i) == 1)

{

cout << "Turn left..." << endl;

fout << "User turned left" << endl;

}

else {

cout << "Turn right..." << endl;

fout << "User turned right" << endl;

}

}

location[location.size() - 1] = NULL; //Deletes last location

fout << "User has returned to last location" << endl;

system("pause");

break;

case 19: //Reached destination

if (Yacht.getEngineOn())

{

cout << "Invalid input: Cannot get out of vehicle if engine is on" << endl;

}

else {

location.push\_back(3);

fout << "User has ended this trip" << endl;

}

system("pause");

break;

case 0: //Quit

cout << "Sorry to see you go" << endl;

endOfMenu = true;

system("pause");

break;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Menu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

} while (userInput != 19);

}

system("pause");

}

//................... . . . . . . . . . . . . . . . . . . . . . . . . .

else {

cout << "Error in location vector..." << endl;

system("pause");

}

} while (endOfMenu == false);

//Thank you message

cout << "Bye, human. Will see you next time.....hopefully.....if I don't take over the humanity!" << endl;

fout.close();

return 0;

}

void printCarMenu()

{

cout << "1. Set age" << endl;

cout << "2. Set price" << endl;

cout << "3. Check balance" << endl;

cout << "4. Turn engine on" << endl;

cout << "5. Turn engine off" << endl;

cout << "6. Turn lights on" << endl;

cout << "7. Turn lights off" << endl;

cout << "8. Add passengers" << endl;

cout << "9. Kick passengers" << endl;

cout << "10. Check mileage" << endl;

cout << "11. Set coefficient of friction" << endl;

cout << "12. Check friction" << endl;

cout << "13. Turn air on" << endl;

cout << "14. Turn air off" << endl;

cout << "15. Change race car status" << endl;

cout << "16. Open sunroof" << endl;

cout << "17. Close sunroof" << endl;

cout << "18. Turn right" << endl;

cout << "19. Turn left" << endl;

cout << "20. Accelerate" << endl;

cout << "21. Decelerate" << endl;

cout << "22. Return to last location" << endl;

cout << "23. Reached destination" << endl;

cout << "0. Quit" << endl;

}

void printTruckMenu()

{

cout << "1. Set age" << endl;

cout << "2. Set price" << endl;

cout << "3. Check balance" << endl;

cout << "4. Turn engine on" << endl;

cout << "5. Turn engine off" << endl;

cout << "6. Turn lights on" << endl;

cout << "7. Turn lights off" << endl;

cout << "8. Add passengers" << endl;

cout << "9. Kick passengers" << endl;

cout << "10. Check mileage" << endl;

cout << "11. Set coefficient of friction" << endl;

cout << "12. Check friction" << endl;

cout << "13. Turn air on" << endl;

cout << "14. Turn air off" << endl;

cout << "15. Change diesel type status" << endl;

cout << "16. Load cargo" << endl;

cout << "17. Unload cargo" << endl;

cout << "18. Check cargo" << endl;

cout << "19. Turn right" << endl;

cout << "20. Turn left" << endl;

cout << "21. Accelerate" << endl;

cout << "22. Decelerate" << endl;

cout << "23. Return to last location" << endl;

cout << "24. Reached destination" << endl;

cout << "0. Quit" << endl;

}

void printBoatMenu()

{

cout << "1. Set age" << endl;

cout << "2. Set price" << endl;

cout << "3. Check balance" << endl;

cout << "4. Turn engine on" << endl;

cout << "5. Turn engine off" << endl;

cout << "6. Turn lights on" << endl;

cout << "7. Turn lights off" << endl;

cout << "8. Add passengers" << endl;

cout << "9. Kick passengers" << endl;

cout << "10. Check propeller level" << endl;

cout << "11. Change propeller level" << endl;

cout << "12. Dock the boad" << endl;

cout << "13. Launch the boat" << endl;

cout << "14. Turn right" << endl;

cout << "15. Turn left" << endl;

cout << "16. Accelerate" << endl;

cout << "17. Decelerate" << endl;

cout << "18. Return to last location" << endl;

cout << "19. Reached destination" << endl;

cout << "0. Quit" << endl;

}

void printPlaneMenu()

{

cout << "1. Set age" << endl;

cout << "2. Set price" << endl;

cout << "3. Check balance" << endl;

cout << "4. Turn engine on" << endl;

cout << "5. Turn engine off" << endl;

cout << "6. Turn lights on" << endl;

cout << "7. Turn lights off" << endl;

cout << "8. Add passengers" << endl;

cout << "9. Kick passengers" << endl;

cout << "10. Turn air on" << endl;

cout << "11. Turn air off" << endl;

cout << "12. Check altitude" << endl;

cout << "13. Change altitude" << endl;

cout << "14. Take off" << endl;

cout << "15. Land" << endl;

cout << "16. Turn right" << endl;

cout << "17. Turn left" << endl;

cout << "18. Accelerate" << endl;

cout << "19. Deccelerate" << endl;

cout << "20. Return to last location" << endl;

cout << "21. Reached destination" << endl;

cout << "22. Quit" << endl;

}

**Vehicle.h**

#pragma once

#ifndef VECHICLE\_H

#define VEHICLE\_H

#include <string>

#include <vector>

#include <iostream>

using namespace std;

class Vehicle

{

public:

Vehicle();

void setAge(int thisAge);

void setPrice(float thisPrice);

void subtractBalance(float thisBalance);

void accelerate();

void decelerate();

void turnEngineOn();

void turnEngineOff();

void turnRight(vector<int> &turns);

void turnLeft(vector<int> &turns);

void turnLightsOn();

void turnLightsOff();

void addPass(int thisPass);

void kickPass(int thisPass);

float getBalance();

float getSpeed();

int getAge();

int getNumPass();

float getPrice();

bool getEngineOn();

bool getLightsOn();

private:

int age;

float price;

float balance;

int speed;

bool engineOn;

bool lightsOn;

int numPass;

};

#endif

**Vehicle.cpp**

#include "Vehicle.h"

Vehicle::Vehicle()

{

age = 0;

price = 0.0;

balance = 40000;

speed = 0;

engineOn = false;

lightsOn = false;

numPass = 0;

}

void Vehicle::setAge(int thisAge)

{

age = thisAge;

}

void Vehicle::setPrice(float thisPrice)

{

price = thisPrice;

}

void Vehicle::subtractBalance(float thisBalance)

{

balance -= thisBalance;

}

void Vehicle::accelerate()

{

speed += 5;

}

void Vehicle::decelerate()

{

speed -= 5;

}

void Vehicle::turnEngineOn()

{

engineOn = true;

}

void Vehicle::turnEngineOff()

{

engineOn = false;

}

void Vehicle::turnRight(vector<int> &turns)

{

turns.push\_back(3);

}

void Vehicle::turnLeft(vector<int> &turns)

{

turns.push\_back(1);

}

void Vehicle::turnLightsOn()

{

lightsOn = true;

}

void Vehicle::turnLightsOff()

{

lightsOn = false;

}

void Vehicle::addPass(int thisPass)

{

numPass += thisPass;

}

void Vehicle::kickPass(int thisPass)

{

numPass -= thisPass;

}

float Vehicle::getBalance()

{

return balance;

}

float Vehicle::getSpeed()

{

return speed;

}

int Vehicle::getAge()

{

return age;

}

int Vehicle::getNumPass()

{

return numPass;

}

float Vehicle::getPrice()

{

return price;

}

bool Vehicle::getEngineOn()

{

return engineOn;

}

bool Vehicle::getLightsOn()

{

return lightsOn;

}

**LandVehicle.h**

#ifndef LANDVEHICLE\_H

#define LANDVEHICLE\_H

#include "Vehicle.h"

using namespace std;

class LandVehicle : public Vehicle

{

public:

LandVehicle();

int getMileage();

float getCoFriction();

float getFriction();

bool getAirOn();

void turnAirOn();

void turnAirOff();

void setMileage(int thisMileage);

void addMileage(int thisMileage);

void setCoFriction(float thisCoFriction);

void applyFriction(int thisSpeed);

private:

int mileage;

float coFriction;

float friction;

bool airOn;

};

#endif // !LANDVEHICLE\_H

**LandVehicle.cpp**

#include "LandVehicle.h"

LandVehicle::LandVehicle()

{

mileage = 1000;

coFriction = 0.5;

applyFriction(getSpeed());

airOn = false;

}

int LandVehicle::getMileage()

{

return mileage;

}

float LandVehicle::getCoFriction()

{

return coFriction;

}

float LandVehicle::getFriction()

{

applyFriction(getSpeed());

return friction;

}

bool LandVehicle::getAirOn()

{

return airOn;

}

void LandVehicle::turnAirOn()

{

airOn = true;

}

void LandVehicle::turnAirOff()

{

airOn = false;

}

void LandVehicle::setMileage(int thisMileage)

{

mileage = thisMileage;

}

void LandVehicle::addMileage(int thisMileage)

{

mileage += thisMileage;

}

void LandVehicle::setCoFriction(float thisCoFriction)

{

coFriction = thisCoFriction;

applyFriction(coFriction);

}

void LandVehicle::applyFriction(int thisSpeed)

{

friction = coFriction \* thisSpeed;

}

**Car.h**

#pragma once

#ifndef CAR\_H

#define CAR\_H

#include "LandVehicle.h"

#include <string>

#include <vector>

#include <iostream>

class Car : public LandVehicle

{

public:

Car();

void setRaceCarStatus(bool thisStatus);

void setSunRoofStatus(bool thisOpen);

bool getRaceCarStatus();

bool getSunroofOpen();

private:

bool raceCarStatus;

bool sunRoofOpen;

};

#endif // !CAR\_H

**Car.cpp**

#include "Car.h"

Car::Car()

{

raceCarStatus = false;

sunRoofOpen = false;

setPrice(15000);

}

void Car::setRaceCarStatus(bool thisStatus)

{

raceCarStatus = thisStatus;

}

void Car::setSunRoofStatus(bool thisOpen)

{

sunRoofOpen = thisOpen;

}

bool Car::getRaceCarStatus()

{

return raceCarStatus;

}

bool Car::getSunroofOpen()

{

return sunRoofOpen;

}

**Truck.h**

#pragma once

#ifndef TRUCK\_H

#define TRUCK\_H

#include "Vehicle.h"

#include "LandVehicle.h"

class Truck : public LandVehicle

{

public:

Truck();

void setDieselTypeStatus(bool thisStatus);

void loadCargo(int thisCargo);

void unloadCargo(int thisCargo);

bool getDieselTypeStatus();

int getCapacity();

int maxCapacity;

private:

bool dieselTypeStatus;

int cargoCapacity;

};

#endif // !TRUCK\_H

**Truck.cpp**

#include "Truck.h"

Truck::Truck()

{

dieselTypeStatus = false;

cargoCapacity = 0;

maxCapacity = 100;

}

void Truck::setDieselTypeStatus(bool thisStatus)

{

dieselTypeStatus = thisStatus;

}

void Truck::loadCargo(int thisCargo)

{

cargoCapacity += thisCargo;

}

void Truck::unloadCargo(int thisCargo)

{

cargoCapacity -= thisCargo;

}

bool Truck::getDieselTypeStatus()

{

return dieselTypeStatus;

}

int Truck::getCapacity()

{

return cargoCapacity;

}

**Boat.h**

#pragma once

#ifndef BOAT\_H

#define BOAT\_H

#include "Vehicle.h"

#include <string>

#include <vector>

#include <iostream>

class Boat : public Vehicle

{

public:

Boat();

int getPropellerLevel();

bool getLaunch();

bool getDock();

void setPropellerLevel(int thisLevel);

void setLaunch(bool thisLaunch);

void setDock(bool thisDock);

private:

int propellerLevel;

bool launch;

bool dock;

};

#endif // !BOAT\_H

**Boat.cpp**

#include "Boat.h"

Boat::Boat()

{

propellerLevel = 0;

launch = false;

dock = true;

}

int Boat::getPropellerLevel()

{

return propellerLevel;

}

bool Boat::getLaunch()

{

return launch;

}

bool Boat::getDock()

{

return dock;

}

void Boat::setPropellerLevel(int thisLevel)

{

propellerLevel = thisLevel;

}

void Boat::setLaunch(bool thisLaunch)

{

launch = thisLaunch;

}

void Boat::setDock(bool thisDock)

{

dock = thisDock;

}

**Plane.h**

#pragma once

#ifndef PLANE\_H

#define PLANE\_H

#include "LandVehicle.h"

#include <string>

#include <vector>

#include <iostream>

class Plane : public LandVehicle

{

public:

Plane();

int getAltitude();

void setAltitude(int thisAltitude);

bool getAirOn();

void turnAirOn();

void turnAirOff();

private:

int altitude;

bool takeOff;

bool land;

bool airOn;

};

#endif // !PLANE\_H

**Plane.cpp**

#include "Plane.h"

Plane::Plane()

{

altitude = 0;

takeOff = false;

land = true;

}

int Plane::getAltitude()

{

return altitude;

}

void Plane::setAltitude(int thisAltitude)

{

altitude = thisAltitude;

}

bool Plane::getAirOn()

{

return airOn;

}

void Plane::turnAirOn()

{

airOn = true;

}

void Plane::turnAirOff()

{

airOn = false;

}

1. **Updated Algorithm**
   1. In class Vehicle
      1. Make private variables
         1. Age (int)
         2. Price (float)
         3. Balance (float)
         4. Speed (int)
         5. engineOn (Boolean)
         6. lightsOn (Boolean)
         7. numPass (int)
      2. Make a vector called turns to store turns that users take
      3. Constructor
         1. Set age to 0
         2. Set price to 0.0
         3. Set balance to $4,000
         4. Set speed = 0
         5. Set engineOn to false
         6. Set lightsOn to false
         7. Set numPass to 0
      4. In *setAge()* function
         1. Set vehicle’s age to be the value passed in
            1. Validate the vehicle’s age passed in is greater than previously stored value if not setting the age for 1st time
            2. If setting the age for 1st time, validate that value passed in is positive
      5. In *setPrice()* function
         1. Set vehicle’s price to be the value passed in
            1. Validate the vehicle’s price passed in is smaller than previously stored value if not setting the price for 1st time
            2. If setting the price for 1st time, validate that value passed in is positive
      6. In *getAge()* function
         1. Return vehicle’s age
      7. In *getPrice()* function
         1. Return vehicle’s price
      8. In *subtractBalance()* function
         1. Subtract passed in value from current balance
      9. In *accelerate()* function
         1. Add 5 to speed
      10. In *decelerate()* function
          1. Subtract 5 from speed
      11. In *getBalance()* function
          1. Return balance
      12. In *getSpeed()* function
          1. Return speed
      13. In *turnEngineOn()* function
          1. Set engineOn variable to true
      14. In *turnEngineOff()* function
          1. Set entingOn variable o false
      15. In *turnRight()* function
          1. Store number 3 in first empty space in the vector
      16. In *turnLeft()* function
          1. Store number 1 in the first empty space in the vector
      17. In *turnLightsOn()* function
          1. Set lightsOn variable to true
      18. In *turnLightsOff()* function
          1. Set lightsOn variable to false
      19. In *addPass()* function
          1. Add passed in value to current number of passengers
      20. In *kickPass()* function
          1. Subtract passed in value from current number of passengers
   2. In class LandVehicle (inherits class Vehicle)
      1. Private variables
         1. Mileage (int) to add mileage of each vehicle
         2. Coefficient of friction (int) to calculate friction
         3. Friction (float) to be calculated by multiplying coefficient of friction by speed
         4. Air on (Boolean) to hold status of A/C on or off
      2. Constructor
         1. Set mileage to 1000
         2. Set coefficient of friction to 0.5
         3. Call function to calculate friction
         4. Set air on status to false
      3. In *getNumPass()* function
         1. Return number of passengers currently in the car / truck
      4. In *getMileage* function
         1. Return current mileage of car / truck
      5. In *getCoFriction* function
         1. Return coefficient of friction
      6. In *getFriction()* function
         1. Return calculated friction
      7. In *getAirOn()* function
         1. Return air on status
      8. In *turnAirOn()* function
         1. Set air on status to true
      9. In *turnAirOff()* function
         1. Set air on status to false
      10. In *addMileage()* function
          1. Add passed in value to current mileage of vehicle
      11. In *setCoFriction()* function
          1. Set the passed in value to the variable of coefficient of friction
      12. In *applyFriction()* function
          1. Friction = coefficient of friction \* speed
   3. In class Car (inherits class LandVehicle)
      1. Private variable for race car status (bool)
      2. Private variable for sun roof open (bool)
      3. Constructor
         1. Set race car status to false
         2. Sets sun roof open status to false
      4. In *setRaceCarStatus()* function
         1. Set race car status to be the value passed in
      5. In *getRaceCarStatus()* function
         1. Return race car status
      6. In *getSunroofOpen()* function
         1. Return sunroof open status
      7. In *setSunroofOpen()* function
         1. Set sunroof status to be the value passed in
   4. In class Truck (inherits class LandVehicle)
      1. Private variable for diesel type status (bool)
      2. Private variable for cargo capacity (int)
      3. Public variable for maximum cargo capacity (int)
      4. Constructor
         1. Set diesel type status to false
         2. Set cargo capacity to 0
         3. Set maximum cargo capacity to 100
      5. In *setDieselTypeStatus()* function
         1. Set diesel type status to be the value passed in
      6. In *getDieselTypeStatus()* function
         1. Return diesel type status
      7. In *getCapacity()* function
         1. Return the current number of cargo in the truck
      8. In *loadCargo()* function
         1. Add the value passed in to the current number of cargo stored
      9. In *unloadCargo()* function
         1. Subtract the value passed in from the current number of cargo stored
   5. In class Boat (inherits class Vehicle)
      1. Private variables
         1. Propeller level (int)
            1. 0 = Outside of water
            2. 1 to 5 = under water
         2. Launch (Boolean)
         3. Dock (Boolean)
      2. Constructor
         1. Set propeller level to 0
         2. Set launch to false
         3. Set dock to true
      3. In *getPropellerLevel()* function
         1. Return current propeller level
      4. In *getLaunch()* function
         1. Return current launch status
      5. In *getDock()* function
         1. Return current dock status
      6. In *setPropellerLevel()* function
         1. Set the current propeller level to be the passed in value
      7. In *setLaunch()* function
         1. Set the current launch status to be the status passed in
      8. In *setDock()* function
         1. Set the current dock status to be the status passed in
   6. In class Plane (inherits class Vehicle)
      1. Private variables
         1. Altitude (int)
         2. Take off status (bool) to store that plane is in air
         3. Land status (bool) to store that plane is in air
      2. Constructor
         1. Set altitude to 0
         2. Set take off status to false
         3. Set land status to true
      3. In *getAltitude()* function
         1. Return current altitude of plane
      4. In *setAltitude()* function
         1. Set current altitude to the value passed in
      5. In *getAirOn()* function
         1. Return air on status
      6. In *turnAirOn()* function
         1. Set air on status to true
      7. In *turnAirOff()* function
         1. Set air on status to false
   7. In class Main
      1. Make variables to be used later
      2. Make vector for turns a car, truck or boat makes.
      3. Make vector for storing locations user goes
      4. Make vector for storing which vehicle user uses each time
      5. Do following, then loop while user chooses to quit program
         1. If user is at home, print a menu with options Car, Truck, Quit
            1. If user selects Car and if last location was home, then user a particular object else use a different object like Taxi

Using switch case, print and call functions for all the different things user can do and loop until user says they have reached destination

Set age if user gives an age more than or equal to previous age

Set price if user gives a price less than or equal to previous price

Return balance

Turn engine on after performing checks

Turn engine off after performing checks

Turn lights on / off after performing checks

Add / kick passengers after performing checks

Return mileage

Set coefficient of friction after performing checks

Return friction

Turn air on / off after performing checks

Set race car status after performing checks

Open / close sunroof after performing checks

Turn right – insert 3 in the first empty box in vector

Turn left – insert 1 in the first empty box in vector

Reached destination – ask if user wants to return to last location. If yes, quit loop and print returning messages

* + - * 1. If user selects Truck

Using switch case, print and call functions for all the different things user can do and loop until user says they have reached destination

Set age if user gives an age more than or equal to previous age

Set price if user gives a price less than or equal to previous price

Return balance

Turn engine on after performing checks

Turn engine off after performing checks

Turn lights on / off after performing checks

Add / kick passengers after performing checks

Return cargo capacity

Add / Subtract cargo after performing checks

Return mileage

Set coefficient of friction after performing checks

Return friction

Turn air on / off after performing checks

Set diesel type status after performing checks

Turn right – insert 3 in the first empty box in vector

Turn left – insert 1 in the first empty box in vector

Reached destination – ask if user wants to return to last location. If yes, quit loop and print returning messages

* + - * 1. If user selects quit, then break out the loop
      1. If user is at airport, print a menu with options Plane, Car, Truck, Quit
         1. If user selects Plane

Using switch case, print and call functions for all the different things user can do and loop until user says they have reached destination

Set age if user gives an age more than or equal to previous age

Set price if user gives a price less than or equal to previous price

Return balance

Turn engine on after performing checks

Turn engine off after performing checks

Turn lights on / off after performing checks

Add / kick passengers after performing checks

Turn air on / off after performing checks

Reached destination – ask if user wants to return to last location. If yes, quit loop and print returning messages

Return altitude

Set altitude after performing checks

Return air on status

* + - * 1. If user selects Car, using switch case, print and call functions for all the different things user can do and loop until user says they have reached destination
        2. If user selects Truck, using switch case, print and call functions for all the different things user can do and loop until user says they have reached destination
        3. If user selects Quit, break out of the loop and ask which location is the user at and store it in the vector
      1. If user is at marina, print a menu with options Boat, Car, Truck, Quit
         1. If user selects Boat

Using switch case, print and call functions for all the different things user can do and loop until user says they have reached destination

Set age if user gives an age more than or equal to previous age

Set price if user gives a price less than or equal to previous price

Return balance

Turn engine on after performing checks

Turn engine off after performing checks

Turn lights on / off after performing checks

Add / kick passengers after performing checks

Set / change launch / dock status after performing checks

Set propeller level after performing checks

Reached destination – ask if user wants to return to last location. If yes, quit loop and print returning messages

* + - * 1. If user selects Car, using switch case, print and call functions for all the different things user can do and loop until user says they have reached destination
        2. If user selects Truck, using switch case, print and call functions for all the different things user can do and loop until user says they have reached destination
        3. If user selects Quit, break out of the loop and ask which location is the user at and store it in the vector
    1. Along each action, update vectors of location and vehicles used, so the program runs correctly

1. **Test Plan Version 3**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass / Fail |
| Valid | 1 | Age of vehicle is greater than previously stored value | Previous: “5”  New: “10” | Age stored successfully | Age stored successfully | Pass |
| Valid | 2 | Price of vehicle is less than previously stored value unless storing it 1st time | Previous: “15000”  New: “10000” | Price stored successfully | Price stored successfully | Pass |
| Valid | 3 | Price value is always positive | “10000” | “Price stored successfully” | “Price stored successfully” | Pass |
| Valid | 4 | User enters corresponding number for choosing either true or false for race car status | “1” for true | “Race car status updated” | “Race car status updated” | Pass |
| Valid | 5 | User enters corresponding number for choosing either true or false for diesel type status | “1” for true | “Diesel type status updated” | “Diesel type status updated” | Pass |
| Valid | 6 | User enters corresponding number for choosing from all vehicles on list | “1” for Car when 1: Car  2: Truck  3: Quit | “You are driving car now” | “You are driving car now” | Pass |
| Valid | 7 | Speed is between 0 mph and 45 mph | “40” | “Speed is now 40” | “Speed is now 40” | Pass |
| Valid | 8.1 | Speed of car / truck is changed while engine is on, speed is not the same as previous | Engine: On  Speed previous: 40  Speed new: “45” | “Speed is now 45” | “Speed is now 45” | Pass |
| Valid | 8.2 | Speed of boat is changed while engine is on, speed is not the same as previous, propeller is under water for boat, and boat is not docked | Engine: On  Speed previous: 40  Speed new: “45”  Propeller level: 2  Boat: launched | “Speed is now 45” | “Speed is now 45” | Pass |
| Valid | 9 | User turns on engine while it was previously off | Previous: engine off  New: engine on | “Engine is now on” | “Engine is now on” | Pass |
| Valid | 10.1 | User turns off engine while speed is 0 mph, engine is previously on, air is off, sun roof is closed | Speed: 0  Previous: engine on  New: engine off  Air: off  Sunroof: closed | “Engine is now off” | “Engine is now off” | Pass |
| Valid | 10.2 | User turns off engine while speed is 0 mph, engine is previously on, propeller level for boat is 0, boat is docked | Speed: 0  Previous: engine on  New: engine off  Propeller level: 0  Boat: docked | “Engine is now off” | “Engine is now off” | Pass |
| Valid | 10.3 | User turns off engine while speed is 0 mph, engine is previously on, air is off, and plane altitude is 0 ft | Speed: 0  Previous: engine on  New: engine off  Air: off  Altitude: 0 | “Engine is now off” | “Engine is now off” | Pass |
| Valid | 11 | User turns on the lights while they are previously off, and while engine is on | Previous: Lights off  New: lights on  Engine: on | “Lights are now on” | “Lights are now on” | Pass |
| Valid | 12 | User turns off the lights while they are previously on, and while engine is on | Previous: Lights on  New: lights off  Engine: on | “Lights are now off” | “Lights are now off” | Pass |
| Valid | 13.1 | Passengers are added to have total of between 1 and 200 for plane, while speed = 0 | Previous: 50  Add: 100  Total: 150  Speed: 0 | “Passengers added successfully” | “Passengers added successfully” | Pass |
| Valid | 13.2 | Passengers are added to have total of between 1 and 4 for Car, while speed = 0 | Previous: 2  Add: 2  Total: 4  Speed: 0 | “Passengers added successfully” | “Passengers added successfully” | Pass |
| Valid | 13.3 | Passengers are added to have total of 1 for Truck, while speed = 0 | Previous: 0  Add: 1  Total: 1  Speed: 0 | “Passengers added successfully” | “Passengers added successfully” | Pass |
| Valid | 13.4 | Passengers are added to have total of 8 for Boat, while speed = 0 | Previous: 4  Add: 4  Total: 8  Speed: 0 | “Passengers added successfully” | “Passengers added successfully” | Pass |
| Valid | 14 | Passengers are kicked out to have total of between 0 and 200 for plane, while speed = 0 | Previous: 150  Kick: 50  Total:100  Speed = 0 | “Passengers kicked out successfully” | “Passengers kicked out successfully” | Pass |
| Valid | 15 | User enters original mileage 1000 | “1000” | “Mileage see successfully” | “Mileage see successfully” | Pass |
| Valid | 16 | Coefficient of friction is between 0 and 1 | “0.5” | “Coefficient is updated successfully” | “Coefficient is updated successfully” | Pass |
| Valid | 17 | User turns air off while it is previously on and while engine is on | Previous: air on  New: air off  Engine: on | “Air is not off” | “Air is not off” | Pass |
| Valid | 18 | User turns air on while it is previously off, while engine is on, and while sunroof is closed | Previous: air off  New: air on  Engine: on  Sunroof: closed | “Air is now on” | “Air is now on” | Pass |
| Valid | 19 | User opens sunroof while it is previously closed, while air is off, and while engine is on | Previous: sunroof closed  New: sunroof open  Air: off  Engine on | “Sunroof is now open” | “Sunroof is now open” | Pass |
| Valid | 20 | User closes sunroof while it is previously open and while engine is on | Previous: sunroof open  New: sunroof closed  Engine: on | “Sunroof is now closed” | “Sunroof is now closed” | Pass |
| Valid | 21 | User sets cargo capacity in beginning to be between 1 and 100 | “50” | “Current cargo capacity set” | “Current cargo capacity set” | Pass |
| Valid | 22 | User adds cargo to have total of 100 or less cargo if number of cargo added is not 0 while truck speed = 0 | Previous: 50  Add: 25  Total: 75  Speed: 0 | “Cargo added successfully” | “Cargo added successfully” | Pass |
| Valid | 23 | User enters propeller level to be between 1 and 5 for boat while boat engine is on | Level: 2  Engine: on | “Propeller level updated successfully” | “Propeller level updated successfully” | Pass |
| Valid | 24 | User selects to launch boat if it is not previously launched | Select “launch”  Previous: docked | “Boat is now launched” | “Boat is now launched” | Pass |
| Valid | 25 | User selects to dock boat if it is not previously docked | Select “dock”  Previous: launched | “Boat is now docked” | “Boat is now docked” | Pass |
| Valid | 26 | User sets altitude of plane to be between 15,000 ft to 35,000 ft while speed 100 – 140 mph and while altitude is previously 0 | Speed: 130  Set new altitude: “30000”  Previous altitude: 0 | “Altitude of plane updated successfully” | “Altitude of plane updated successfully” | Pass |
| Valid | 27 | User sets altitude of plane to be 0 while speed 100 – 140 mph and while altitude is previously 15,000 ft to 35,000 | Speed: 110  Set new altitude: “0”  Previous altitude: 18000 | “Altitude of plane updated successfully” | “Altitude of plane updated successfully” | Pass |
| Invalid | 1 | Age of vehicle is less than previously stored value | Previous: 15  New: 10 | “Invalid input: Age cannot be less than last one” | “Invalid input: Age cannot be less than last one” | Pass |
| Invalid | 2 | Price of vehicle is more than previously stored value unless storing it 1st time | Previous: 15000  New: 20000 | “Invalid input: Price cannot be more than last one” | “Invalid input: Price cannot be more than last one” | Pass |
| Invalid | 3 | Price value is negative | “-25000” | “Invalid input: Price cannot be negative” | “Invalid input: Price cannot be negative” | Pass |
| Invalid | 4 | User enters number not corresponding to choosing either true or false for race car status | “3” | “Invalid input: Try again” | “Invalid input: Try again” | Pass |
| Invalid | 5 | User enters number not corresponding to choosing either true or false for race diesel type status | “100” | “Invalid input: Try again” | “Invalid input: Try again” | Pass |
| Invalid | 6 | User enters number not corresponding to choosing either Car or Truck | “-4” | “Invalid input: Try again” | “Invalid input: Try again” | Pass |
| Invalid | 7 | Speed of car / truck is less than 0 | “-15” | “Invalid input: Speed cannot be less than 0” | “Invalid input: Speed cannot be less than 0” | Pass |
| Invalid | 8 | Speed of car / truck is more than 45 | “55” | “Invalid input: Speed cannot be more than 45” | “Invalid input: Speed cannot be more than 45” | Pass |
| Invalid | 9 | User selects to change speed while engine is off | Engine: off  Previous speed: 0  New: 45 | “Invalid input: Cannot change speed while engine is off” | “Invalid input: Cannot change speed while engine is off” | Pass |
| Invalid | 10 | User selects to change speed to the previously stored speed value | Previous: 40  New: 40 | “Invalid input: Cannot change speed to the same value as before” | “Invalid input: Cannot change speed to the same value as before” | Pass |
| Invalid | 11 | User selects to change speed while boat propeller level is 0 | Previous: 0  New: 30  Propeller level: 0 | “Invalid input: Cannot change speed while boat propeller level is not in water” | “Invalid input: Cannot change speed while boat propeller level is not in water” | Pass |
| Invalid | 12 | User selects to change speed while boat is not launched | Previous: 0  New: 20  Boat: docked | “Invalid input: Cannot change speed while boat is docked” | “Invalid input: Cannot change speed while boat is docked” | Pass |
| Invalid | 13 | User selects to turn engine on while it’s on | Previous:  Engine on  New:  Engine on | “Invalid input: Cannot turn engine on while it is already on” | “Invalid input: Cannot turn engine on while it is already on” | Pass |
| Invalid | 14 | User selects to turn engine off while it’s off | Previous: Engine off  New: Engine off | “Invalid input: Cannot turn engine off while is already off” | “Invalid input: Cannot turn engine off while is already off” | Pass |
| Invalid | 15 | User selects to turn engine off while speed is more than 0 | Previous: engine on  New: engine off  Speed: 14 | “Invalid input: Cannot turn engine off while vehicle is moving” | “Invalid input: Cannot turn engine off while vehicle is moving” | Pass |
| Invalid | 16 | User selects to turn engine off while air is on | Air: on  Previous: engine on  New: engine off | “Invalid input: Cannot turn engine off while A/C is on” | “Invalid input: Cannot turn engine off while A/C is on” | Pass |
| Invalid | 17 | User selects to turn engine off while sun roof is open | Sunroof: open  Previous: engine on  New: engine off | “Invalid input: Cannot turn engine of while sunroof is open” | “Invalid input: Cannot turn engine of while sunroof is open” | Pass |
| Invalid | 18 | User selects to turn engine off while boat propeller level is more than 0 | Propeller level: 1  Previous: engine on  New: engine off | “Invalid input: Cannot turn engine off while boat propeller is inside water” | “Invalid input: Cannot turn engine off while boat propeller is inside water” | Pass |
| Invalid | 19 | User selects to turn engine off while boat is not docked | Boat: launched  Previous: engine on  New: engine off | “Invalid input: Cannot turn engine off while boat is launched” | “Invalid input: Cannot turn engine off while boat is launched” | Pass |
| Invalid | 20 | User selects to turn engine off while plane altitude is more than 0 | Altitude: 20,000  Previous: engine on  New: engine off | “Invalid input: Cannot turn engine off while plane is in air” | “Invalid input: Cannot turn engine off while plane is in air” | Pass |
| Invalid | 21 | User selects to turn lights on while they are on | Previous: lights on  New: lights on | “Invalid input: Cannot turn lights on if they already on” | “Invalid input: Cannot turn lights on if they already on” | Pass |
| Invalid | 22 | User selects to turn lights on while engine is off | Engine: off  Previous: lights off  New: lights on | “Invalid input: Cannot turn lights on if engine is off” | “Invalid input: Cannot turn lights on if engine is off” | Pass |
| Invalid | 23 | User selects to turn lights off while are off | Previous: lights off  New: lights off | “Invalid input: Cannot turn lights off while they are already off” | “Invalid input: Cannot turn lights off while they are already off” | Pass |
| Invalid | 24 | User selects to add passengers to have total of 0 | Previous passengers: 0  Add: “0”  Total: 0 | “Invalid input: Cannot add 0 passengers” | “Invalid input: Cannot add 0 passengers” | Pass |
| Invalid | 25 | User selects to add passengers to have total of more than 4 passengers in car | Previous passengers: 2  Add: 3  Total: 5 | “Invalid input: Cannot add passengers to have total of more than 4” | “Invalid input: Cannot add passengers to have total of more than 4” | Pass |
| Invalid | 26 | User selects to add passengers to have total of more than 1 passengers in truck | Previous passengers: 0  Add: 2  Total: 2 | “Invalid input: Cannot add passengers to have total of more than 1” | “Invalid input: Cannot add passengers to have total of more than 1” | Pass |
| Invalid | 27 | User selects to add passengers to have total of more than 8 passengers in boat | Previous passengers: 3  Add: 10  Total: 13 | “Invalid input: Cannot add passengers to have total of more than 8” | “Invalid input: Cannot add passengers to have total of more than 8” | Pass |
| Invalid | 28 | User selects to add passengers to have total of more than 200 passengers in plane | Previous passengers: 100  Add: 150  Total: 250 | “Invalid input: Cannot add passengers to have total of more than 200” | “Invalid input: Cannot add passengers to have total of more than 200” | Pass |
| Invalid | 29 | User selects to kick 0 passengers out | Kick “0” passengers out | “Invalid input: Cannot kick 0 passengers out” | “Invalid input: Cannot kick 0 passengers out” | Pass |
| Invalid | 30 | User selects to kick passengers out to have total of negative passengers | Previous passengers: 2  Kick: 3  Total: -1 | “Invalid input: Cannot kick out more passengers than there are” | “Invalid input: Cannot kick out more passengers than there are” | Pass |
| Invalid | 31 | User enters a negative number for mileage | “-1000” | “Invalid input: Cannot have negative mileage” | “Invalid input: Cannot have negative mileage” | Pass |
| Invalid | 32 | Coefficient of friction is less than 0 | “-1” | “Invalid input: Cannot have negative coefficient of friction” | “Invalid input: Cannot have negative coefficient of friction” | Pass |
| Invalid | 33 | Coefficient of friction is more than 1 | “2” | “Invalid input: Cannot have coefficient of friction more than 2” | “Invalid input: Cannot have coefficient of friction more than 2” | Pass |
| Invalid | 34 | User selects to turn air off while it is off | Previous air: off  New air: off | “Invalid input: Cannot turn engine off if it is already off” | “Invalid input: Cannot turn engine off if it is already off” | Pass |
| Invalid | 35 | User selects to turn air on while it is on | Previous air: on  New air: on | “Invalid input: Cannot turn A/C on while it is already on” | “Invalid input: Cannot turn A/C on while it is already on” | Pass |
| Invalid | 36 | User selects to turn air on while engine is off | Engine: off  Previous air: off  New air: on | “Invalid input: Cannot turn air on if engine is off” | “Invalid input: Cannot turn air on if engine is off” | Pass |
| Invalid | 37 | User selects to turn air on while sunroof is open | Sunroof: on  Previous air: off  New air: on | “Invalid input: Cannot turn air on if sunroof is open” | “Invalid input: Cannot turn air on if sunroof is open” | Pass |
| Invalid | 38 | User selects to open sunroof while it is on | Previous sunroof: open  New sunroof: open | “Invalid input: Cannot open sunroof if it is already open” | “Invalid input: Cannot open sunroof if it is already open” | Pass |
| Invalid | 39 | User selects to open sunroof while engine is off | Engine: off  Previous sunroof: closed  New sunroof: open | “Invalid input: Cannot open sunroof if engine is off” | “Invalid input: Cannot open sunroof if engine is off” | Pass |
| Invalid | 40 | User selects to open sunroof while air is on | Air: on  Previous sunroof: closed  New sunroof: open | “Invalid input: Cannot open sunroof if A/C is on” | “Invalid input: Cannot open sunroof if A/C is on” | Pass |
| Invalid | 41 | User selects to close sunroof while it is closed | Previous sunroof: closed  New sunroof: closed | “Invalid input: Cannot close sunroof if it is already closed” | “Invalid input: Cannot close sunroof if it is already closed” | Pass |
| Invalid | 42 | User sets cargo capacity in the beginning to be negative | “-50” | “Invalid input: Cannot have a negative cargo” | “Invalid input: Cannot have a negative cargo” | Pass |
| Invalid | 43 | User sets cargo capacity in the beginning to be greater than 100 | “150” | “Invalid input: Cannot have more than 100 cargo” | “Invalid input: Cannot have more than 100 cargo” | Pass |
| Invalid | 44 | User adds cargo to have total of more than 100 cargo | Previous cargo: 10  Add: “100”  Total: 110 | “Invalid input: Cannot add cargo to have total of more than 100 cargo” | “Invalid input: Cannot add cargo to have total of more than 100 cargo” | Pass |
| Invalid | 45 | User adds cargo to have total of the same amount as before | Previous cargo: 75  Add: 0  Total: 75 | “Invalid input: Cannot add 0 cargo” | “Invalid input: Cannot add 0 cargo” | Pass |
| Invalid | 46 | User selects to add cargo while speed of vehicle is greater than 0 | Speed: 40  Select add cargo | “Invalid input: Cannot add cargo if vehicle is moving” | “Invalid input: Cannot add cargo if vehicle is moving” | Pass |
| Invalid | 47 | User enters propeller level to be the same level as before | Previous level: 1  New: “1” | “Invalid input: Cannot set propeller level to be same as before” | “Invalid input: Cannot set propeller level to be same as before” | Pass |
| Invalid | 48 | User enters propeller level to be negative | “-4” | “Invalid input: Propeller level cannot be negative” | “Invalid input: Propeller level cannot be negative” | Pass |
| Invalid | 49 | User enters propeller level to be more than 5 | “10” | “Invalid input: Propeller level cannot be more than 5” | “Invalid input: Propeller level cannot be more than 5” | Pass |
| Invalid | 50 | User selects to change propeller level while engine is off | Engine: off  Previous level: 4  New: 5 | “Invalid input: Cannot change propeller level if engine is off” | “Invalid input: Cannot change propeller level if engine is off” | Pass |
| Invalid | 51 | User selects to launch boat while it is already launched | Previous boat: launched  New: Launch | “Invalid input: Cannot launch boat if it is already launched” | “Invalid input: Cannot launch boat if it is already launched” | Pass |
| Invalid | 52 | User selects to dock boat while it is already docked | Pervious boat: docked  New: docked | “Invalid input: Cannot dock boat if it is already docked” | “Invalid input: Cannot dock boat if it is already docked” | Pass |
| Invalid | 53 | User selects to set altitude of plane to be less than 15,000 ft while speed is 141-500 mph while landing / taking off | Speed: 135  Previous altitude: 15000  Set altitude: “1” | “Invalid input: Cannot change altitude to be less than 15,000 if speed is 141 – 500 mph ” | “Invalid input: Cannot change altitude to be less than 15,000 if speed is 141 – 500 mph ” | Pass |
| Invalid | 54 | User selects to set altitude of plane to be more than 35,000 ft while speed is 100-140 mph while landing / taking off | Speed: 135  Previous altitude: 0  Set altitude: “99000” | “Invalid input: Plane cannot go above altitude of 35,000” | “Invalid input: Plane cannot go above altitude of 35,000” | Pass |
| Invalid | 55 | User selects to set altitude of plane to be between 15,000 ft to 35,000 ft while speed is less than 100 while landing / taking off | Speed: 99  Previous altitude: 0  Set altitude: “30150” | “Invalid input: Cannot change altitude if speed is less than 100 mph” | “Invalid input: Cannot change altitude if speed is less than 100 mph” | Pass |
| Invalid | 56 | User selects to set altitude of plane to be between 15,000 ft to 35,000 ft while speed is more than 140 while landing / taking off | Speed: 141  Previous altitude: 0  Set altitude: “30150” | “Invalid input: Cannot change altitude if speed is more than 140 mph” | “Invalid input: Cannot change altitude if speed is more than 140 mph” | Pass |
| Invalid | 57 | User selects to increase speed of plane while it is previously 500 mph (max) | Previous speed: 500  Add: 1  Total: 501 | “Invalid input: Plane cannot go more than 500 mph” | “Invalid input: Plane cannot go more than 500 mph” | Pass |
| Invalid | 58 | User selects to decrease speed of plane while it is previously 120 mph (min) | Previous speed: 120  Subtract: 1  Total: 119 | “Invalid input: Plane cannot go less than 120 mph to maintain altitude” | “Invalid input: Plane cannot go less than 120 mph to maintain altitude” | Pass |
| Invalid | 59 | User selects to set altitude of plane to be 0 while speed is more than 140 mph | Speed: 145  Previous altitude: 15001  Set altitude: 0 | “Invalid input: Cannot land plane if speed is more than 140 mph” | “Invalid input: Cannot land plane if speed is more than 140 mph” | Pass |
| Invalid | 60 | User selects to set altitude of plane to be 0 while plane is already on land | Previous altitude: 0  Speed 120  Set altitude: 0 | “Invalid input: Cannot land plane if it is already on ground” | “Invalid input: Cannot land plane if it is already on ground” | Pass |
| Invalid | 61 | User selects to change altitude while engine is off | Engine: off  Set altitude: “15001” | “Invalid input: Cannot change altitude if engine is off” | “Invalid input: Cannot change altitude if engine is off” | Pass |

1. **Screenshots**

Valid Test Case 1:





Valid Test Case 2:



This proves that previously stored value is 15000



Valid Test Case 3:



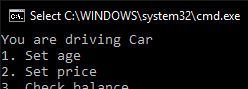
Valid Test Case 4:



Valid Test Case 5:



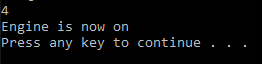
Valid Test Case 6:



Valid Test Case 7:



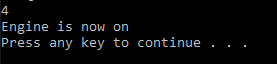
Valid Test Case 8.1:

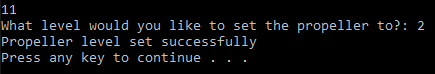






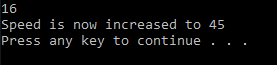
Valid Test Case 8.2:











Valid Test Case 9:



This proves that engine is previously off



Valid Test Case 10.1:



This proves speed is 0 mph









Valid Test Case 10.2:



This proves that speed = 0







This proves boat is docked



Valid Test Case 10.3:









Valid Test Case 11:





This proves that lights are off



Valid Test Case 12:

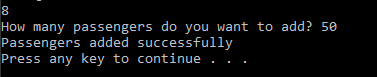


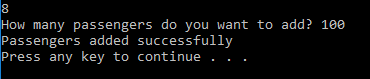
This proves that lights are on



Valid Test Case 13.1:

  
This proves that speed is 0

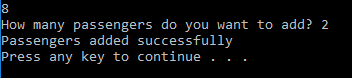


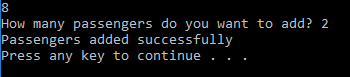


Valid Test Case 13.2:

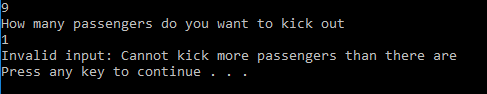


This proves that speed is 0





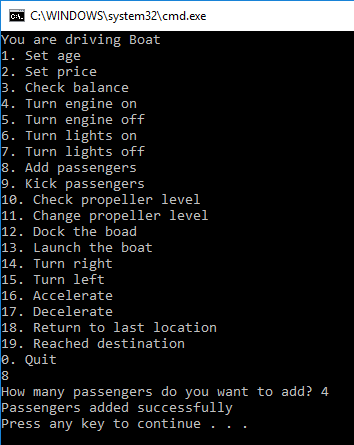
Valid Test Case 13.3:

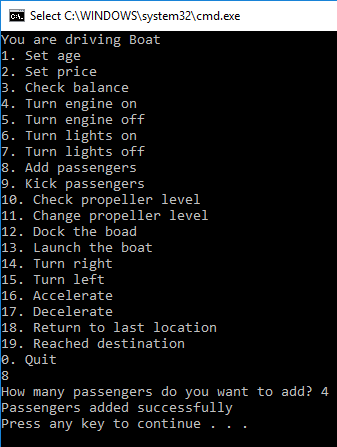


This proves there are 0 passengers previously

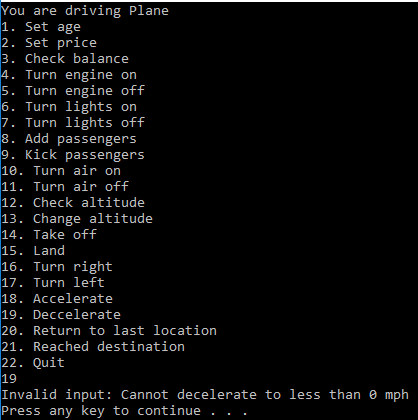


Valid Test Case 13.4:

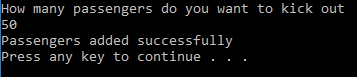
 Previous: 4 passengers

New: 4

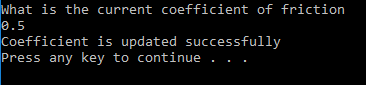
Valid Test Case 14:

Proves that speed is 0





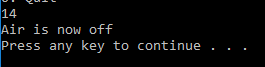
Valid Test Case 16:



Valid Test Case 17:



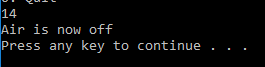




Valid Test Case 18:



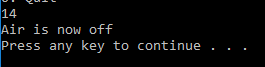






Valid test Case 19:









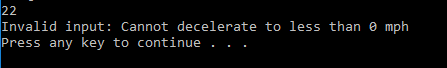
Valid Test Case 20:



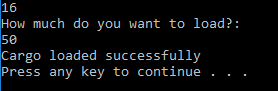


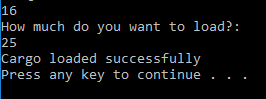


Valid Test Case 22:

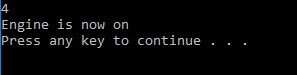


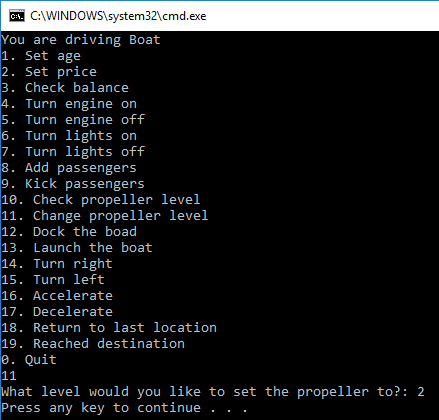
Speed = 0





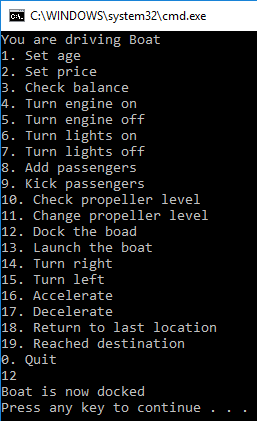
Valid Test Case 23:

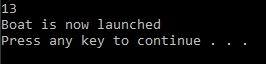




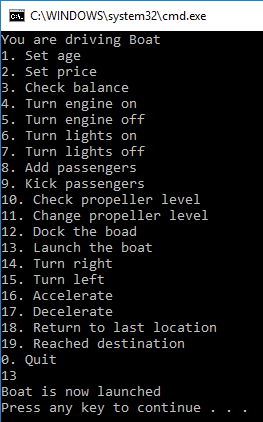


Valid Test Case 24:



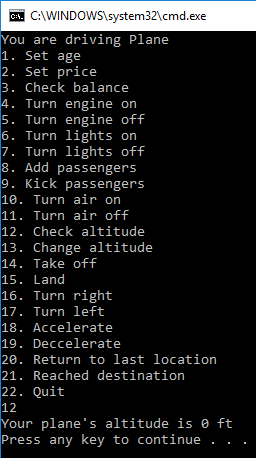


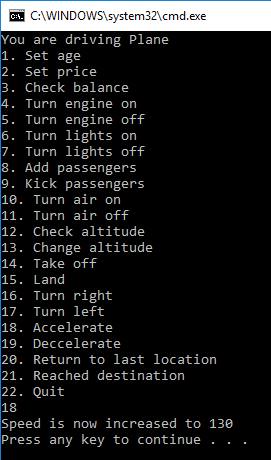
Valid Test Case 25:

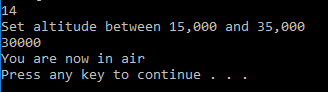




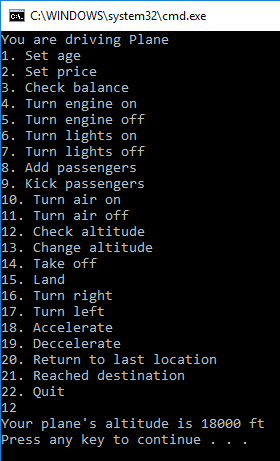
Valid Test Case 26:







Valid Test Case 27:

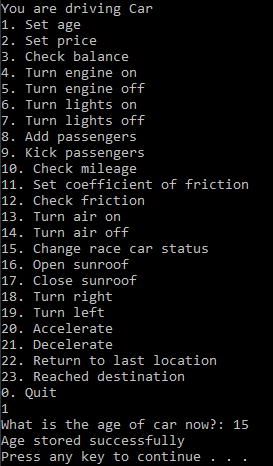




This proves that speed is 120

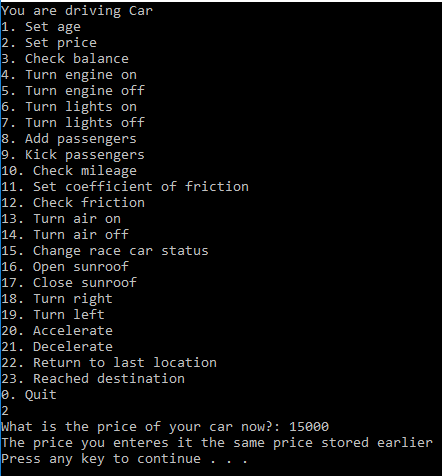


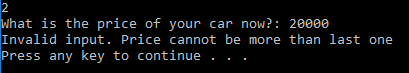
Invalid Test Case 1:



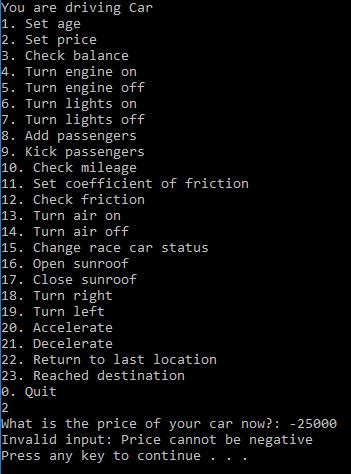


Invalid Test Case 2:

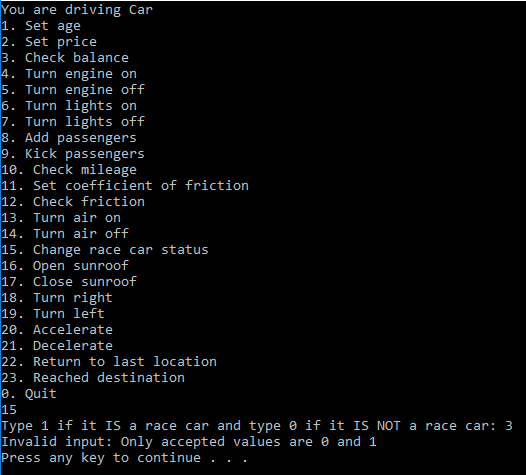




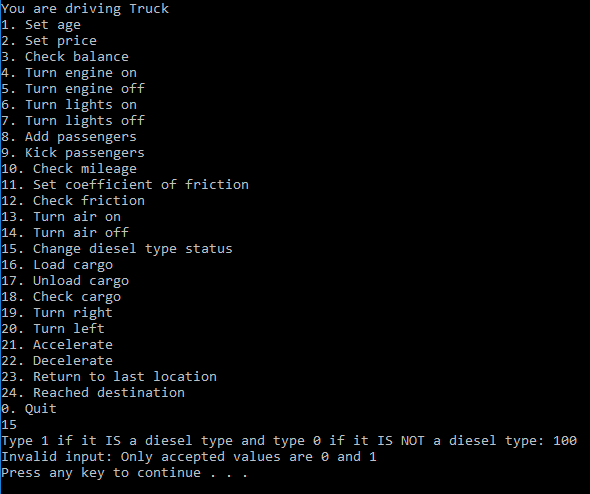
Invalid Test Case 3:



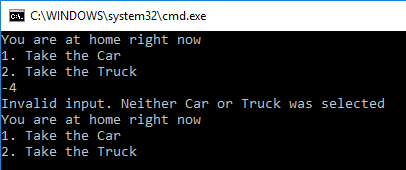
Invalid Test Case 4:



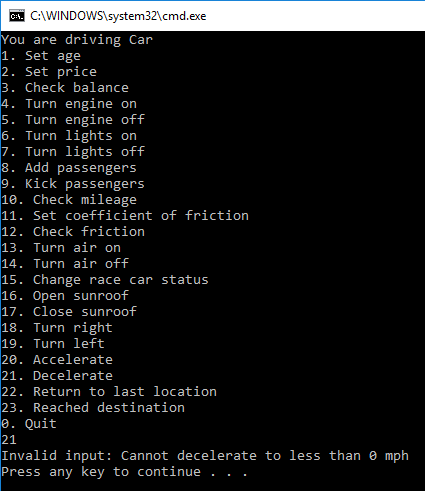
Invalid Test Case 5:



Invalid test Case 6:

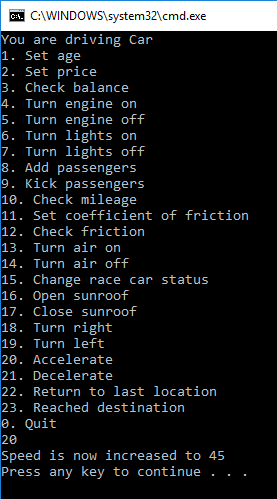


Invalid Test Case 7:



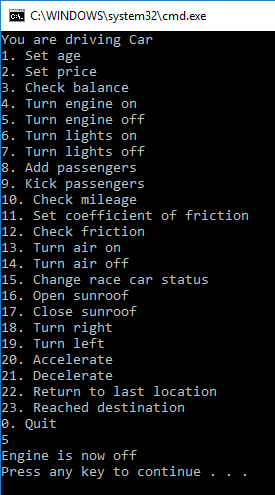
This proves that it cannot go below 0 mph

Invalid test Case 8:



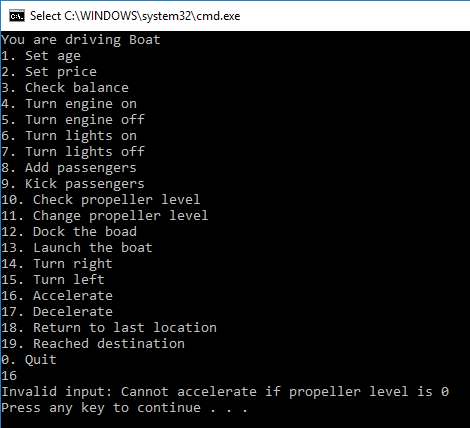


Invalid Test Case 9:

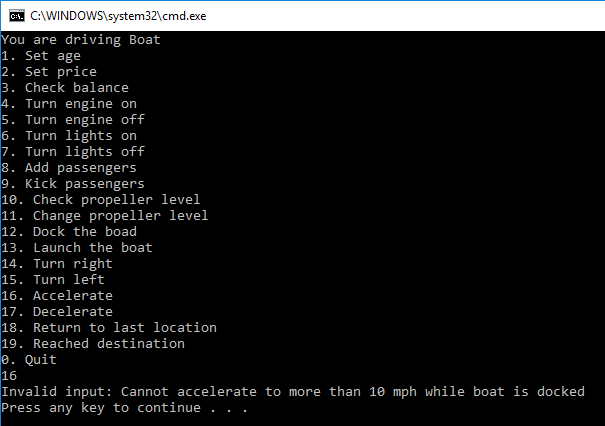




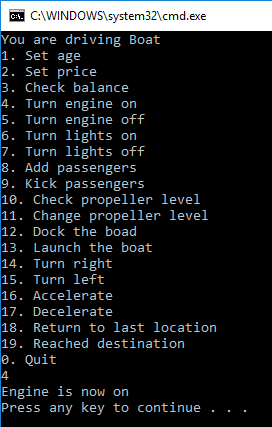
Invalid Test Case 10:



Invalid Test Case 12:

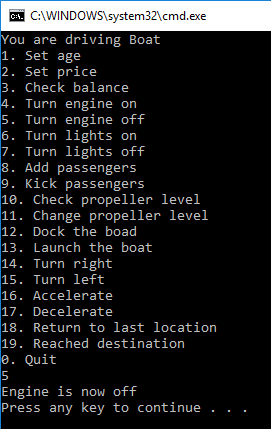


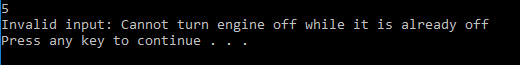
Invalid Test Case 13:



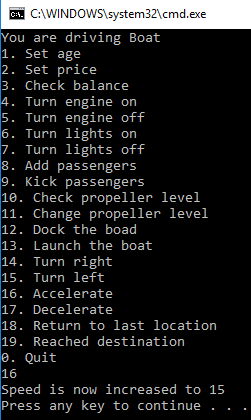


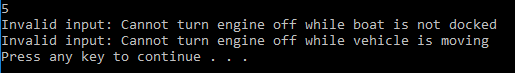
Invalid Test Case 14:



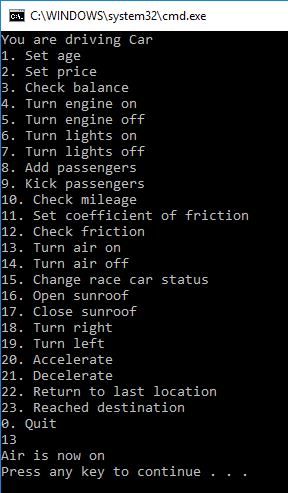


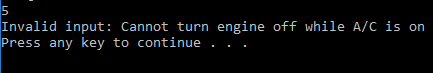
Invalid test Case 15:



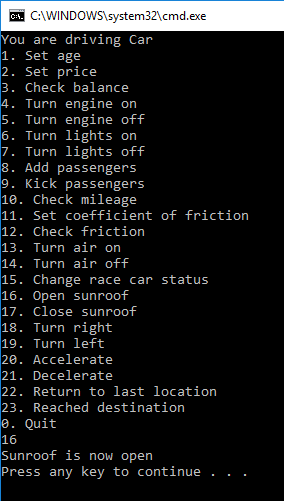


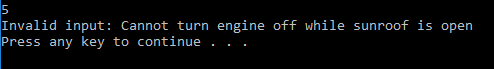
Invalid test Case 16:



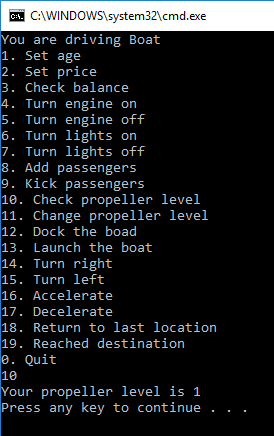


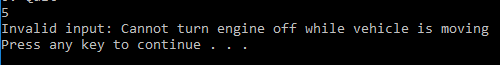
Invalid Test Case 17:



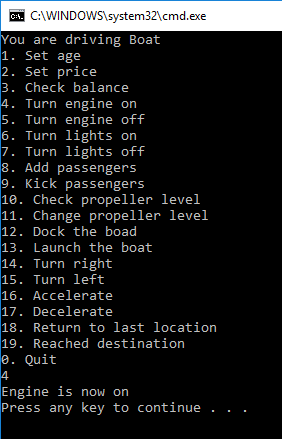


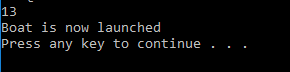
Invalid Test Case 18:

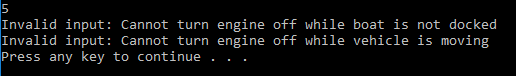




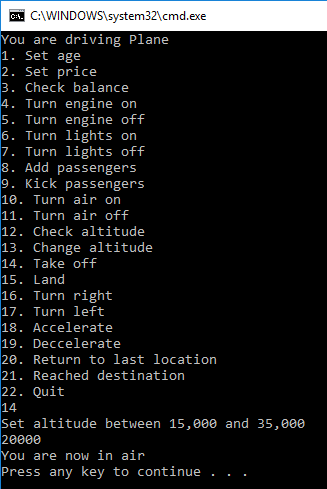
Invalid Test Case 19:





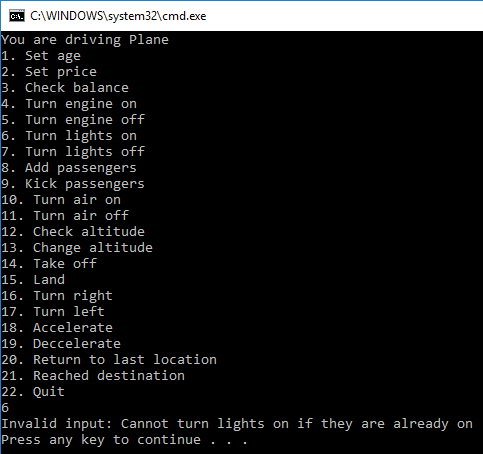


Invalid Test Case 20:

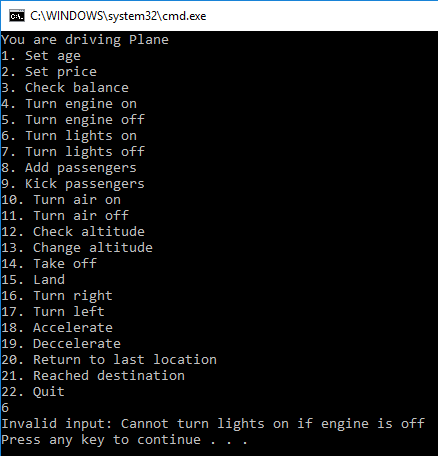




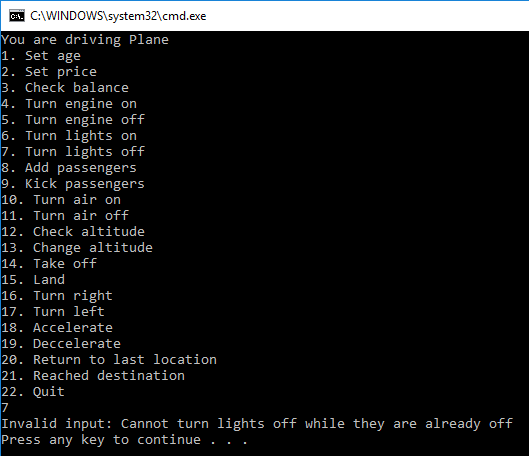
Invalid Test Case 21:



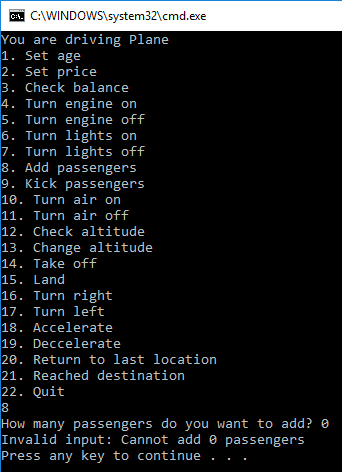
Invalid Test Case 22:



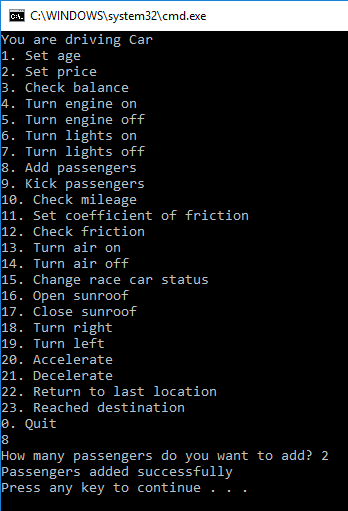
Invalid Test Case 23:

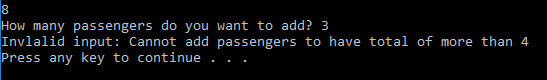


Invalid Test Case 24:

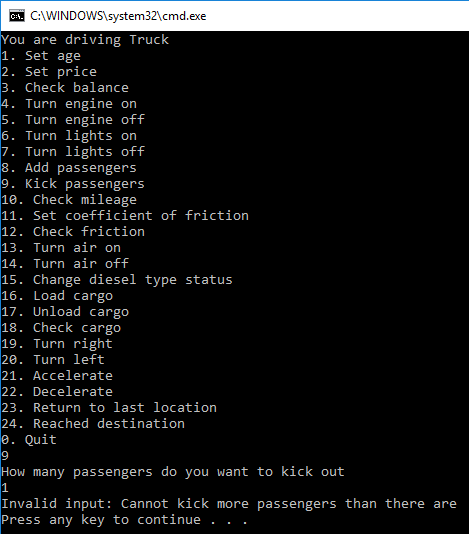


Invalid Test Case 25:

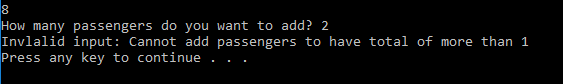




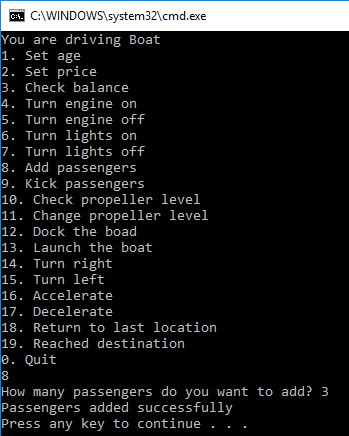
Invalid Test Case 26:

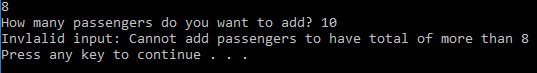


Proves there are currently 0 passengers

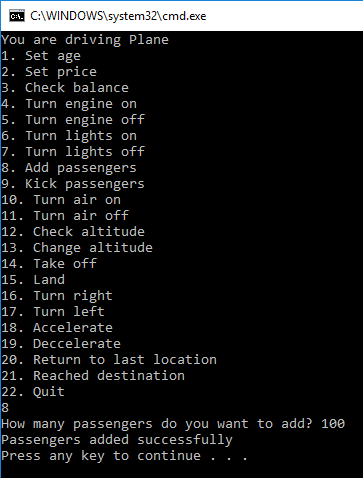


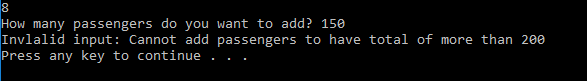
Invalid Test Case 27:



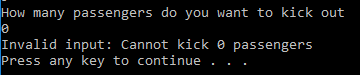


Invalid Test Case 28:

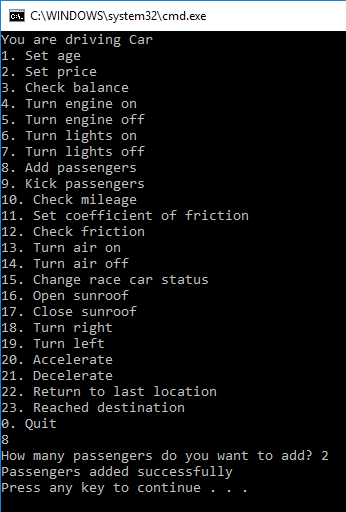


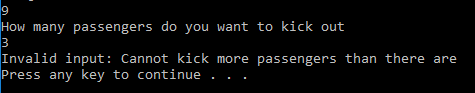


Invalid Test Case 29:

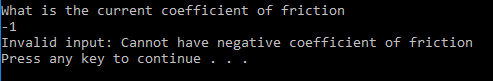


Invalid Test Case 30:

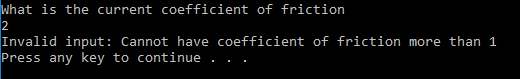




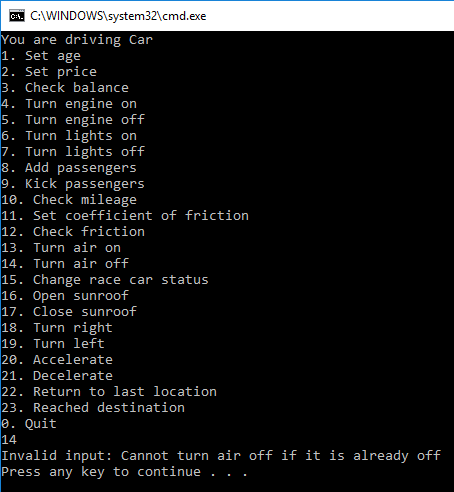
Invalid Test Case 32:



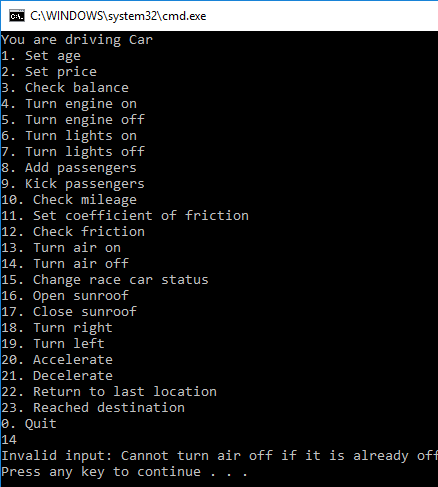
Invalid Test Case 33:



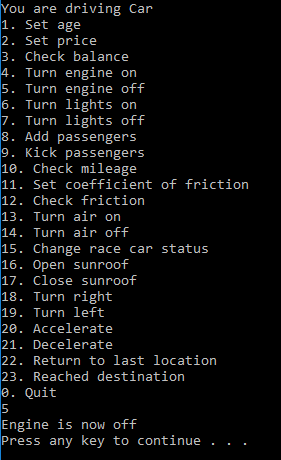
Invalid Test Case 34:

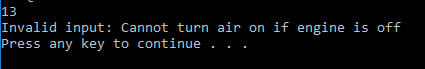


Invalid Test Case 35:

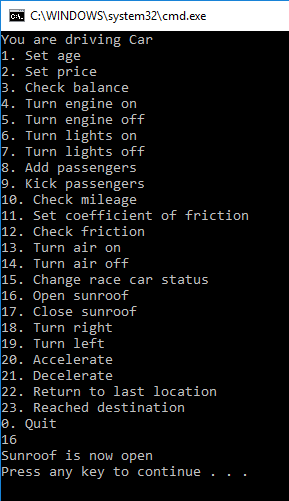


Invalid Test Case 36:



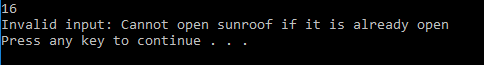


Invalid Test Case 37:

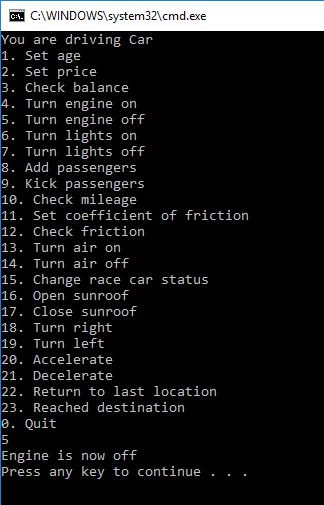


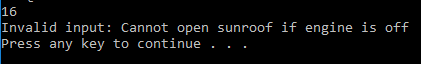


Invalid Test Case 38:

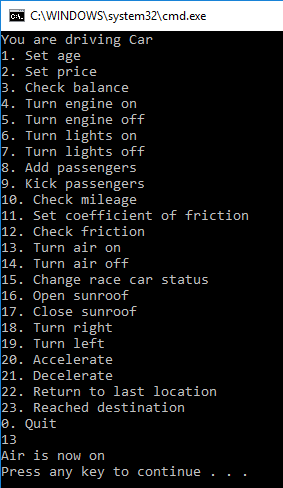


Invalid Test Case 39:





Invalid Test Case 40:

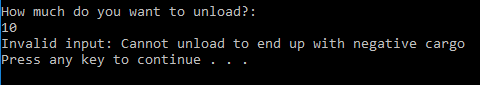




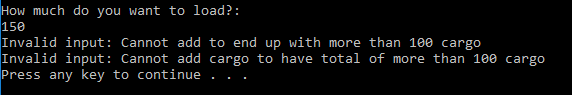
Invalid Test Case 41:



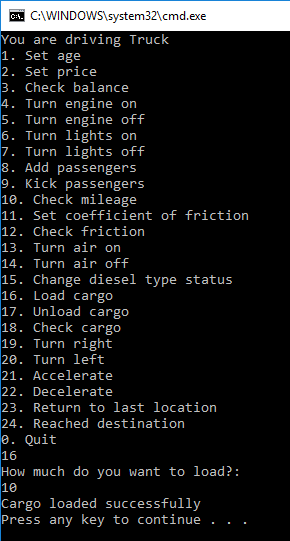
Invalid Test Case 42:

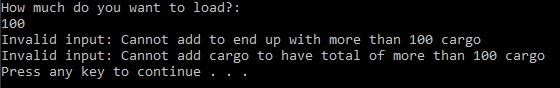


Invalid Test Case 43:

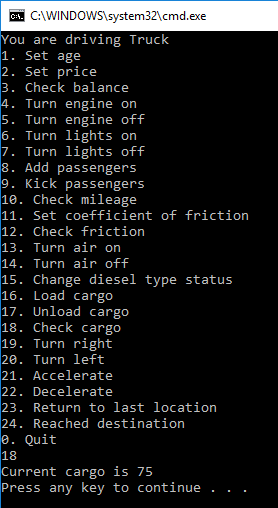


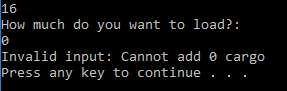
Invalid Test Case 44:



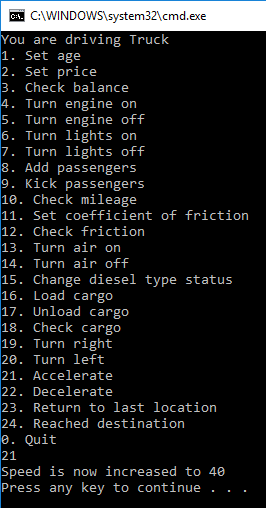


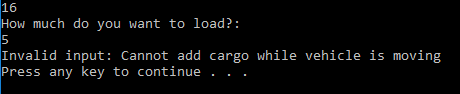
Invalid Test Case 45:



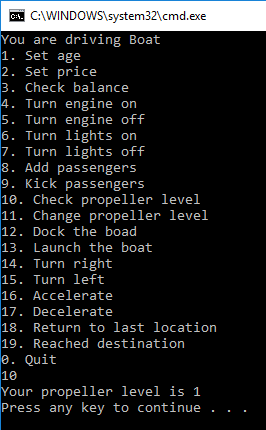


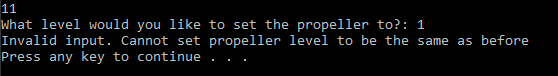
Invalid Test Case 46:



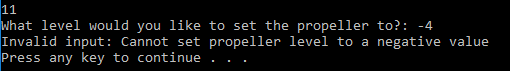


Invalid Test Case 47:

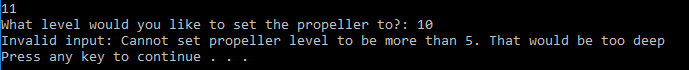




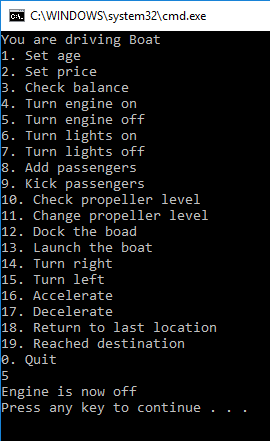
Invalid Test Case 48:



Invalid Test Case 49:

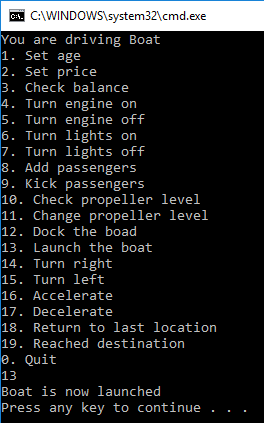


Invalid Test Case 50:



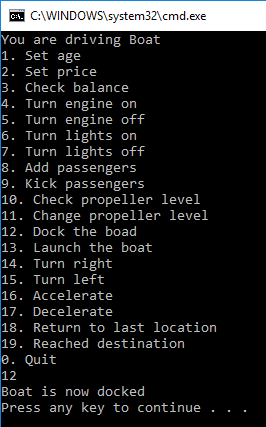


Invalid Test Case 51:



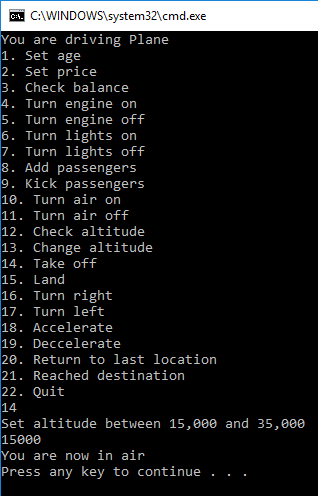


Invalid Test Case 52:





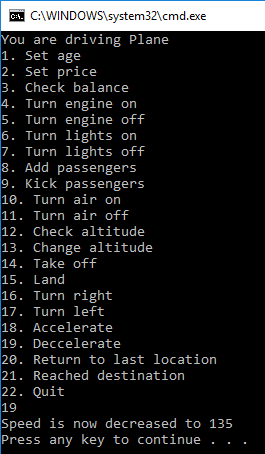
Invalid Test Case 53:



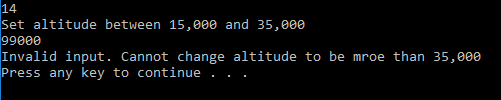




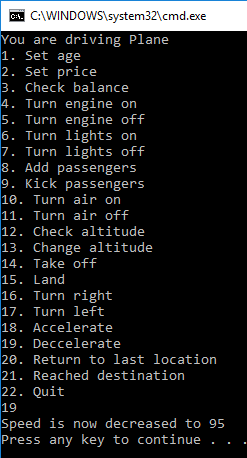
Invalid Test Case 54:

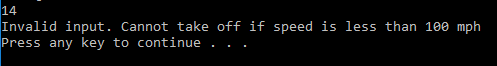




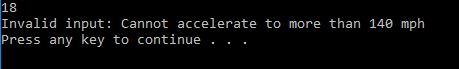


Invalid Test Case 55:

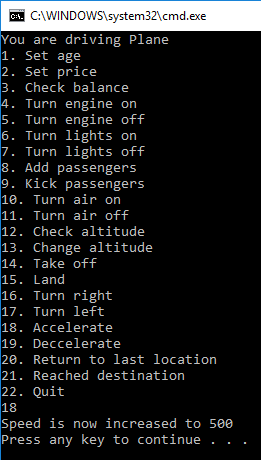




Invalid Test Case 56:



Invalid Test Case 57:

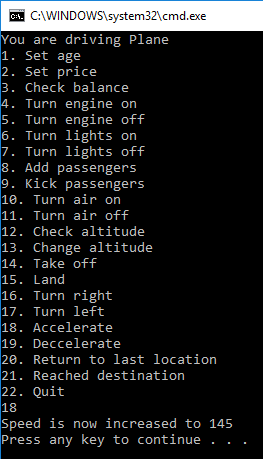




Invalid Test Case 58:

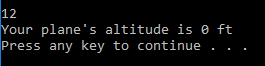


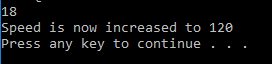
Invalid Test Case 59:





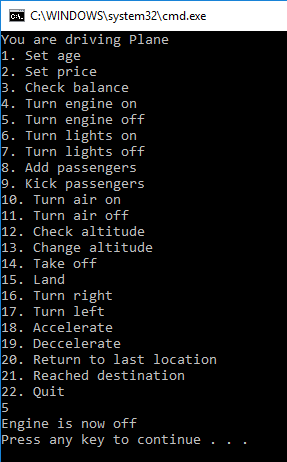
Invalid Test Case 60:







Invalid Test Case 61:





1. **Status**

Program works nearly perfect with assumptions and exceptions and test cases in mind. When returning to last location, program does not let user return to a location before the last location.