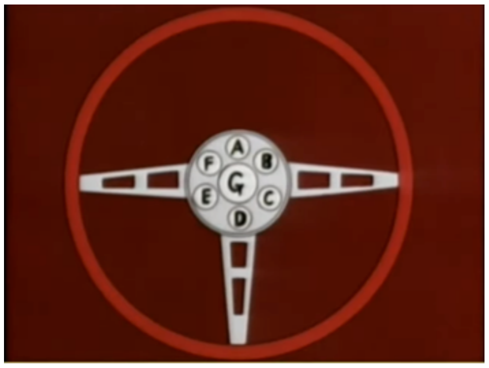
Midterm Programming Project: Automotive Electronic Control Unit (ECU)
Topics: Boolean Expressions, Switch Statement, If/Else statements
CPSC-298-6 Programming in C++
jbonang@chapman.edu

Introduction

An electronic control unit (ECU) is an embedded system in automotive electronics that controls one or more of the electrical systems or subsystems in a vehicle. Some modern motor vehicles have up to 150 ECUs linked together via a Controller Area Network, or CAN, bus. The software running on the ECU is often coded in C/C++ (though portions of the software may be coded using a model-based development environment such as Mathwork's MATLAB/Simulink environment).

Custom Electronic Control Units are needed for the unusual features of specialty vehicles. In this project, you'll develop the framework of an Electronic Control Unit for the special functions of one of the world's most famous vehicles.

The special features are activated by the driver using an array of buttons labeled A through G in the center of the steering wheel. Instead of writing the embedded code to interact with actuators, you'll instead output text to simulate the interaction. Further, you'll only implement functions A, B and E (that's enough to give you the idea).



Function A deploys a built-in "auto-jack", a hydraulic jack feature that raises the vehicle for servicing by pit crew members (it has other uses too). Function B deploys special "grip" surfaces on top of the treads of the tires for additional traction. Function E actives infra-red driving lights. Functions A and B interact with each other. The auto-jack hydraulic jack feature must not be deployed when the semi-adhesive "grip tires" are in use. (The grip tiers may be deployed if the auto-jack is already activated and the vehicle is raised, though.)

You'll implement a C++ class named EngineControlUnit that implements the logic to control these features. In addition to a constructor and destructor, the class will include a handleCommand function that uses a switch statement to handle the button presses A, B or E.

```
class ElectronicControlUnit
{
public:
    ElectronicControlUnit();
    ~ElectronicControlUnit();
    void handleCommand(char cCommand);
```

The handleCommand should accept a character that corresponds to the button press.

The class will maintain the state of each of the controls using Boolean variables - one for each feature. If a feature is activated and the button for the feature is again pressed, the feature is deactivated. For example, if the button to deploy the auto-jacks, A, is pressed, the jacks deploy. When pressed again, the jacks retract.

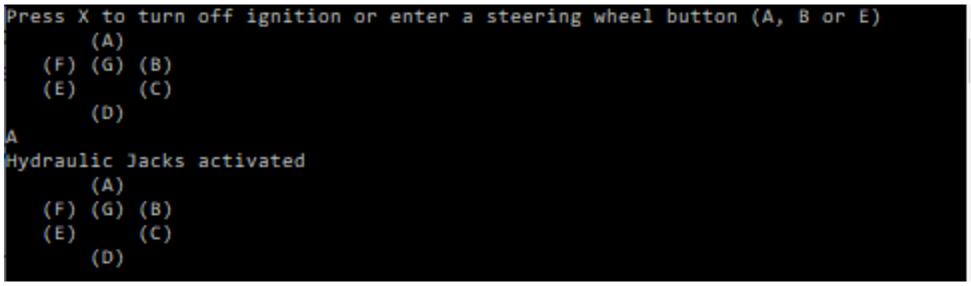
```
private:
    bool m_bActivatedHydraulicJacks;    // Button A
    bool m_bActivatedBeltTires;         // Button B
    bool m_bActivatedIRIllumination;    // Button E
```

A private member function can be used to handle each feature. For example, you might define a function actuateHydraulicJacks to handle the deployment (and retraction) of the hydraulic jacks.

```
 void actuateHydraulicJacks();
```

The private member functions will emit text strings such as "Hydraulic Jacks activated" to indicate the change in state of the controls.

You'll implement a main function that exercises your ElectronicControlUnit class. The main function will display the steering wheel and allow the user to press any of the buttons (but entering them at the keyboard).



```
Press X to turn off ignition or enter a steering wheel button (A, B or E)
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
A
Hydraulic Jacks activated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
```

After a key is pressed, a std::cout statement reports the change in state of the ECU. If the user presses X, the ignition is turned off and the program terminates.

The next section provides background information on the assignment that you can skip over. The Assignment section following the Background section details exactly what you need to do.

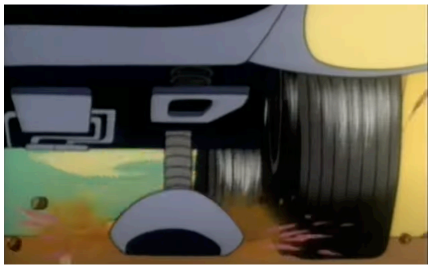
Background

Perhaps the most famous fictional vehicle is the 1966 Batmobile. Nearly as famous is the James Bond Aston Martin DB5 originally appearing in the 1964 film *Goldfinger*.

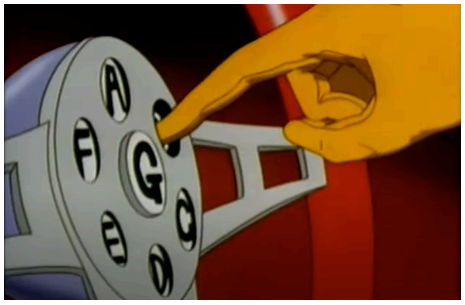
Both the Batmobile and the DB5 included a plethora of specialty features: drag chutes, mortars, machine guns, bullet proof shields, and so on. Even more famous, at least among the junior high school set at the time, is the Mach 5, driven by none other than Speed Racer.

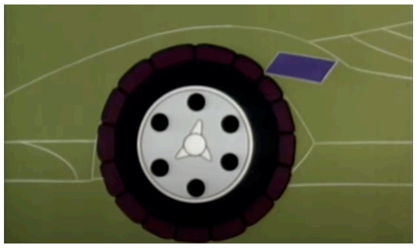
General Motors didn't introduce the first electronics system into automobiles until 1978, so the Batmobile and DB5 wouldn't have incorporated ECUs. However, the animated Mach 5 was re-created for the 2008 movie *Speed Racer*, and even more recently, by automotive aficionados. These realizations of the Mach 5 must have specialty ECUs to handle the special gadgets.

The gadgets are all activated by buttons on the steering wheel, though for this project you'll only implement buttons A, B and E. Button A deploys automatic hydraulic jacks used to raise the vehicle for servicing (or to jump over obstacles).

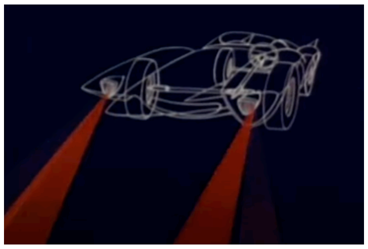


Button B activates special "grip tire treads" that provide extra traction.

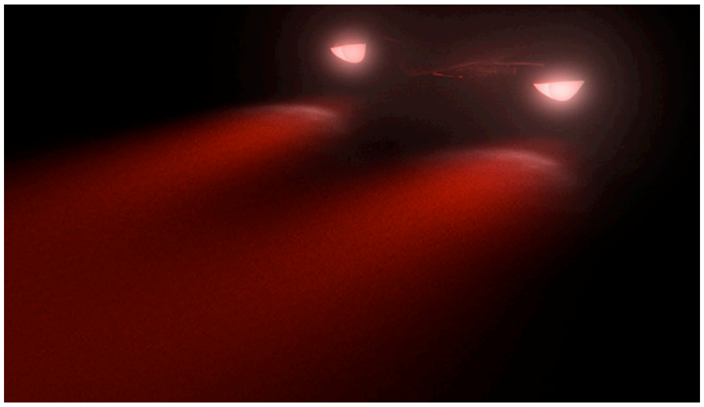




Button E activates infrared driving lights.



The realized Mach 5 includes pseudo-IR driving lights.



Pressing the button for a feature when the feature is already activated deactivates the feature.

References

For additional details on the Mach 5 features, the following Youtube video may help: (Speed Racer - Mach 5's Advanced Features - YouTube) https://www.youtube.com/watch?v=43rxThjYWsE

Additionally, the Mach 5 Wikipedia article provides extensive information on the (animated) car's capabilities.

https://en.wikipedia.org/wiki/Mach_Five

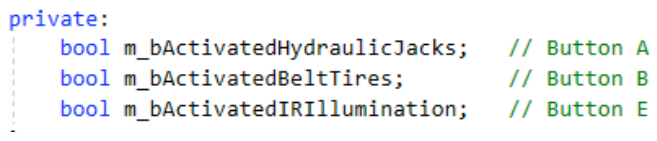
For information on real world electronic Control Systems, begin with the Wikipedia article Electronic control unit - Wikipedia (https://en.wikipedia.org/wiki/Electronic_control_unit).

In this assignment, you'll create a simple C++ class that simulates the ECU of the Mach 5.

The Assignment

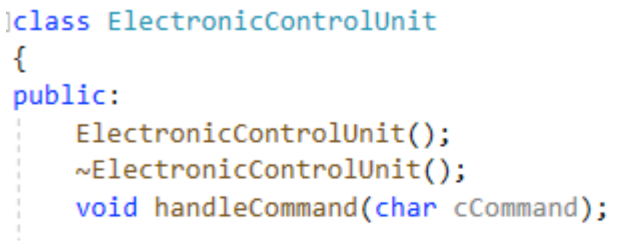
Write a C++ class to simulate the Electronic Control Unit (ECU) of the realized Mach V race car.

The class must use (at least) three Boolean variables to track the state of each feature; for example:



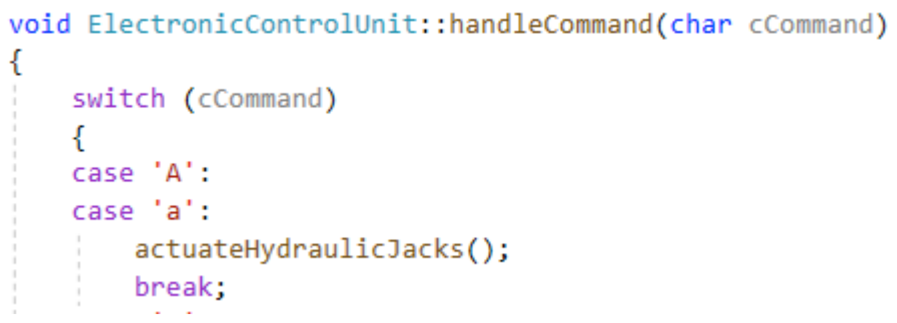
```
private:
    bool m_bActivatedHydraulicJacks;   // Button A
    bool m_bActivatedBeltTires;        // Button B
    bool m_bActivatedIRIllumination;   // Button E
```

In addition to a constructor and a destructor, the class must implement a public member function that accepts a single character as a parameter, for example, handleCommand as shown below.



```
class ElectronicControlUnit
{
public:
    ElectronicControlUnit();
    ~ElectronicControlUnit();
    void handleCommand(char cCommand);
```

The handleCommand member function must utilize a switch statement. Each case within the switch statement must invoke a private member function to handle the activation/deactivation of the feature.

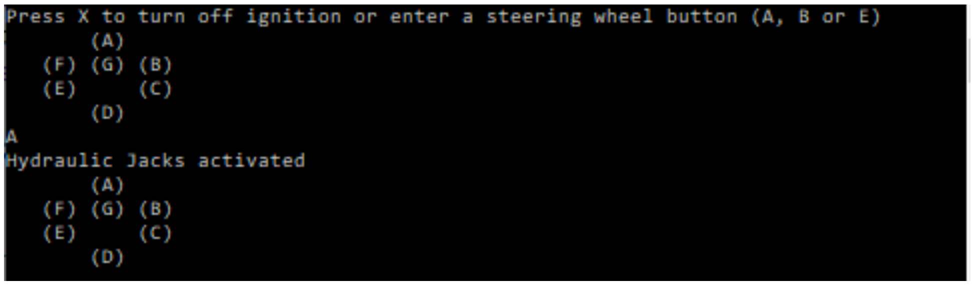


```
void ElectronicControlUnit::handleCommand(char cCommand)
{
    switch (cCommand)
    {
    case 'A':
    case 'a':
        actuateHydraulicJacks();
        break;
```

The private feature-handling functions should toggle the Boolean variable that maintains the state for that feature. It must also use std::cout to report the current state of the feature.

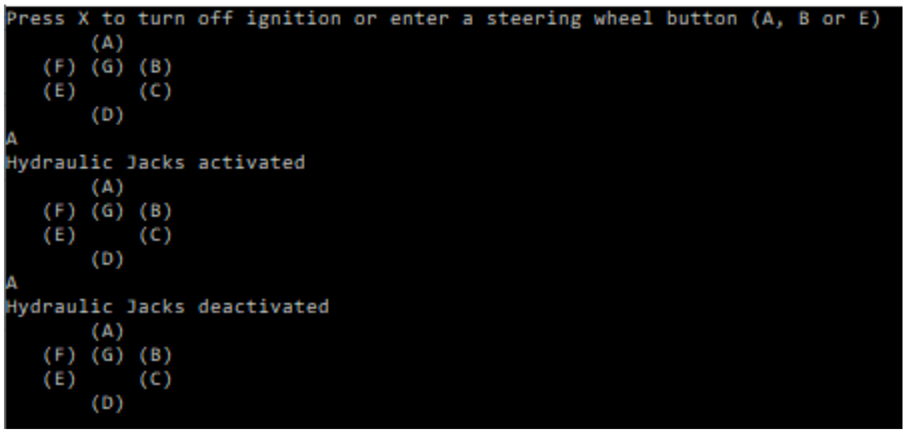
The program will include a main function that displays the steering wheel using std::cout. User input must be implemented using std::cin, where the user enters a single character (A, B or E).

The screen capture below shows the steering wheel display and the user entering the A character. The private member function that handles feature A reports that the feature, Hydraulic Jacks in this case, has been activated.



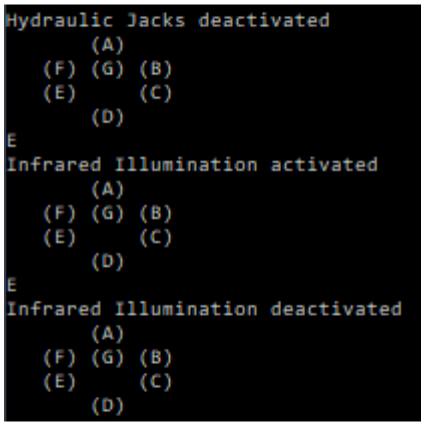
```
Press X to turn off ignition or enter a steering wheel button (A, B or E)
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
A
Hydraulic Jacks activated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
```

If the user selects the A button again, the state of the Hydraulic Jacks is toggled; that is, they are retracted and the state is "deactivated."



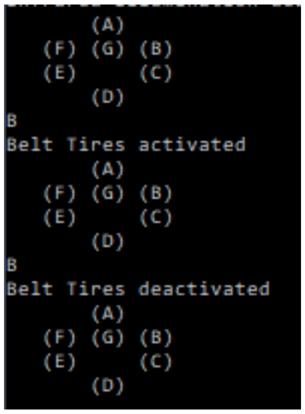
```
Press X to turn off ignition or enter a steering wheel button (A, B or E)
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
A
Hydraulic Jacks activated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
A
Hydraulic Jacks deactivated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
```

The same functionality is supported for the Infrared driving lights.



```
Hydraulic Jacks deactivated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
E
Infrared Illumination activated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
E
Infrared Illumination deactivated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
```

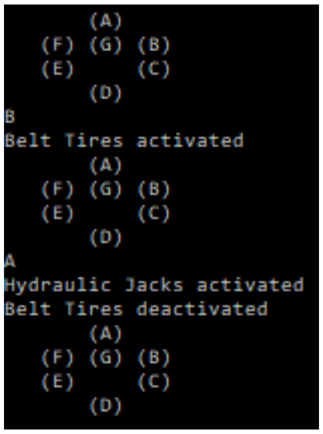
The Special Grip Tire treads for extra traction behave the same way.



```
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
B
Belt Tires activated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
B
Belt Tires deactivated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
```

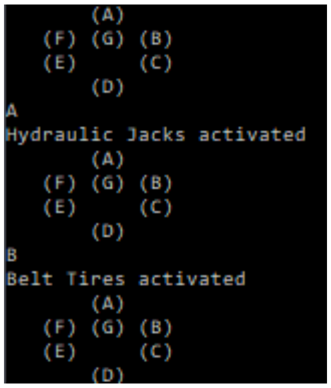
The Auto-Jack feature and the Special Grip Tire treads have an interaction which you must handle.

Activating the Auto-Jack automatically deactivates the Special Grip (or "Belt") Tire feature. Activating the Jacks when the grip tires are deployed may damage one or the other (the tires try to grip while the jacks try to break the grip).



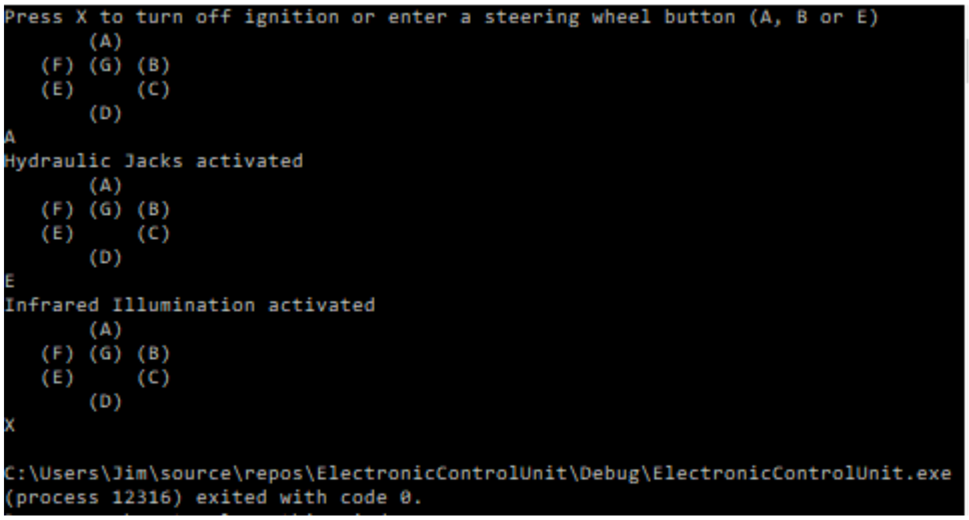
```
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
B
Belt Tires activated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
A
Hydraulic Jacks activated
Belt Tires deactivated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
```

In contrast, if the Auto-Jack is already activated, the car is raised on its jacks and the grip ("Belt") tires may be activated. (No harm is done as the wheels are off the ground; there's nothing to grip.)



```
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
A
Hydraulic Jacks activated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
B
Belt Tires activated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
```

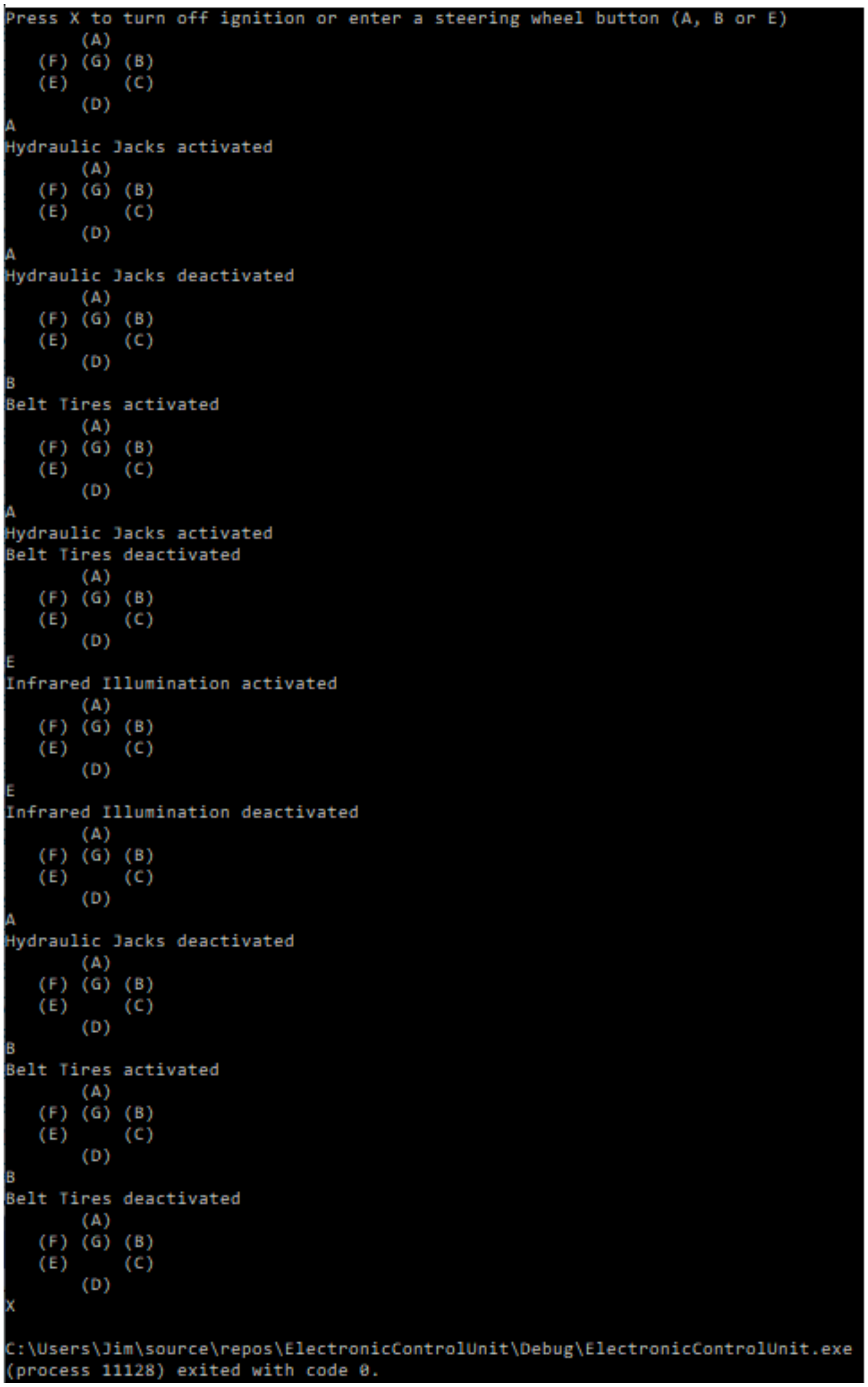
The main function should also allow the user to enter the 'X' character to turn off the ignition and end the program. Otherwise, the main program should continue to display the wheel controls after each press (and the reporting of the status).



```
Press X to turn off ignition or enter a steering wheel button (A, B or E)
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
A
Hydraulic Jacks activated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
E
Infrared Illumination activated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
X

C:\Users\Jim\source\repos\ElectronicControlUnit\Debug\ElectronicControlUnit.exe
(process 12316) exited with code 0.
```

To test your program, enter the sequence of commands shown in the following screen capture.



```
Press X to turn off ignition or enter a steering wheel button (A, B or E)
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
A
Hydraulic Jacks activated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
A
Hydraulic Jacks deactivated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
B
Belt Tires activated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
A
Hydraulic Jacks activated
Belt Tires deactivated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
E
Infrared Illumination activated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
E
Infrared Illumination deactivated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
A
Hydraulic Jacks deactivated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
B
Belt Tires activated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
B
Belt Tires deactivated
       (A)
   (F) (G) (B)
   (E)     (C)
       (D)
X

C:\Users\Jim\source\repos\ElectronicControlUnit\Debug\ElectronicControlUnit.exe
(process 11128) exited with code 0.
```

Submit this Midterm programming project as you would any programming project.