23COA202 Coursework

F312575

Semester 1

1 FSMs

- My implementation of the parking management system features a finite state machine with three states, Sync, Main and Select.
- The system begins in the Sync state. In this state the lcd screen is set to purple and the character "Q" is sent over the serial monitor at a frequency of one per second. The system is waiting for the input "X" to be sent over the serial monitor with a no new line or carriage return character, only then will the system transition into the main state.
- While in the main state, the system is able to carry out its main functionality as this is where all the processing of data and output onto the lcd screen is carried out. Upon holding down the SELECT button though, the system will transition into the SELECT state.
- While in the SELECT state, the system sets the lcd screen to purple and displays my student ID. If the SELECT button is not being held down, then the system will transition back into the Main state.

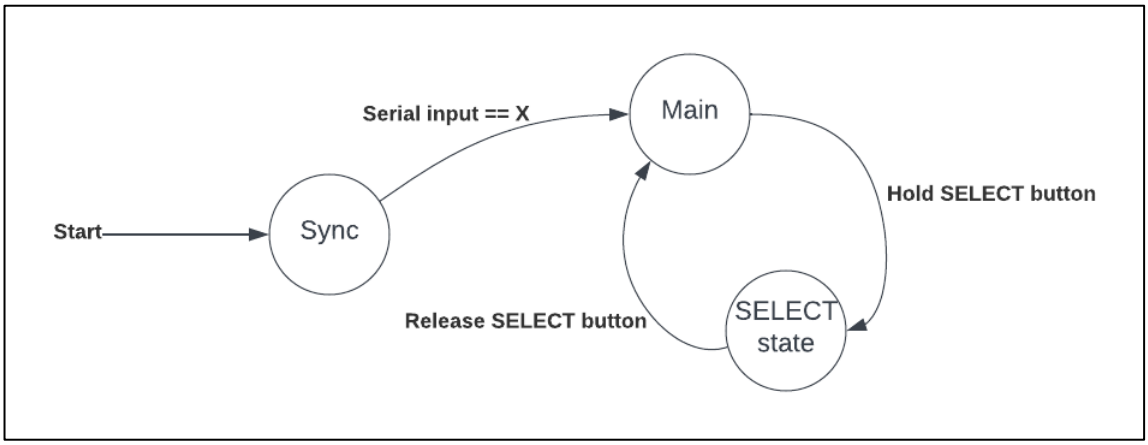

Figure 1:Finite state machine

2 Data structures

- For this project, I decided to use a struct array to handle my data. I defined a structure “Cars” which had the following fields:
  - CarReg- A string field to store the alphanumeric car registration numbers
  - CarType-A character field to store the car type
  - CarPark-A string field to store the name of the car park each car is situated in
  - Paid-A Boolean field to store the payment status of each car (“PD” or “NPD”)
  - EntryTime- A string field to store the arrival time of each car
  - ExitTime-A string field to store the exit time of each car
- I defined an array “CarLog” which is an instance of the “Cars” structure. All valid inputs over the serial monitor are parsed into the CarLog arrayand it can store up to the pre-determined “MaxCars” number of car records. “MaxCars” is the global variable that determines the size of the array and is set to 7 in this current system.
- The following functions all update or alter the information stored within the CarLog array:
  - **ParseInput** -Function to parse a serial input and populate the structure
  - **insertIntoCarLog**-Function to insert the parsed input into the struct array at the current index (current index auto increments with each insertion)
  - **UpdatePayment**-Function to update the payment status of a car already in the CarLogs array
  - **UpdateCarType**-Function to update the car type of a car already in the CarLogs array and with a payment status of “PD”
  - **UpdateCarPark**-Function to update the car park of a car already in the CarLogs array and with a payment status of “PD”
  - **DeleteCar**-Function to delete a car from the CarLogs array with a payment status of “PD”

3 Debugging

To debug my code I used a combination of manual testing and automated testing using the provided python script. I used manual testing while still developing parts of the system, making sure each new function works with my own hand typed data and inputs directly sent over the serial monitor. If a problem was found, I would go back and ensure and tweak the particular function I was working on until it worked. Once the system was complete, I made use of the automated python script, with some of my own data inserted into the script, to test the whole system as a whole to ensure the functionality was working. I would use the "DEBUG" messages sent over the console to identify what parts of the system were working as intended and those that needed adjustment in order to attain the desired results.

4 Reflection

All in all, I would say I am very happy with my implementation of the system. As a complete beginner to the world of coding and programming, being able to implement the base functionality as well as some extension features is something I am very pleased about. However my code is not perfect and I will be the first to acknowledge this. If there is one thing I would like to massively improve on, it is the amount of cars that can be stored in my system. My implementation can only hold a maximum of seven cars within the array. In theory I could increase the size of the array to hold more cars while not using up all the available memory, but through testing I have learnt the optimal number is seven as I need to leave enough overhead memory in order for the program is run smoothly without issues. There are two possible solutions to allow for more cars to be stored, the first being better memory management through the use of pointers. Using pointers would have allowed to save on SRAM which would have allowed me to increase the size of my array, thus storing more cars. The second option would have been utilizing eeprom memory. This would not only have allowed me to store more cars but would have also made my system non-volatile which would result in data being kept even after closing the program. I feel improving this one aspect of my implementation would have drastically elevated the program as a whole.

5 UDCHARS

I implemented the UDCHARS extension using the “arrows” function that I defined in my code. My UDCHARS extension consists of two unique characters that I defined as the “arrows” function, “Up_arrow” and “Down_arrow” respectively. Logic was also implemented in the function to ensure that each respective arrow only shows up on the screen with respect to the information currently being displayed on the screen and the amount to be displayed, i.e. the “Up_arrow” is only displayed if there is a record to be displayed above and the “Down_arrow” is displayed only if there is a record to be displayed below. All this is done by keeping track of the global variable “i” which keeps track of what index of the CarLogs array is being displayed on the lcd screen.

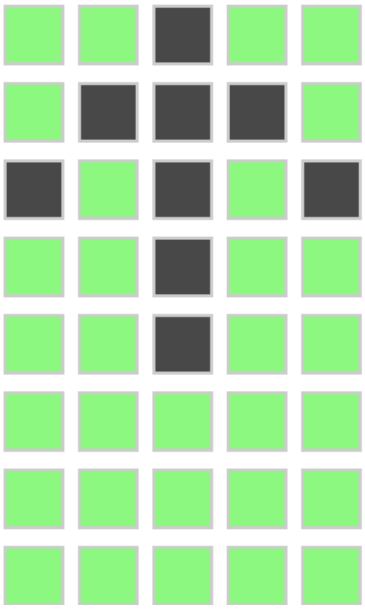

Figure 2: Up_arrow

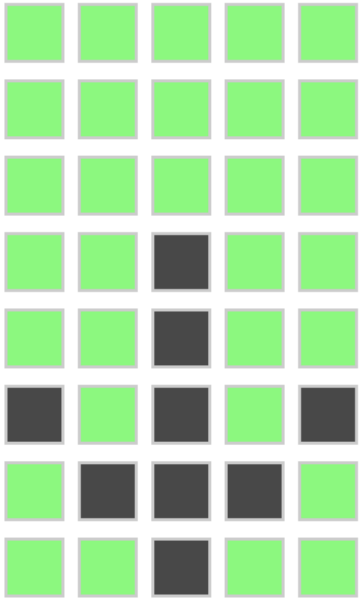

Figure 3:Down_arrow

6 FREERAM

For the FREERAM extension I made use of the “MemoryFree” library. Using the “freeMemory” function from the library, I was able to print the free amount of SRAM to the lcd screen when the program is in the “SELECT” state. The available amount of SRAM is displayed right below my student ID when the SELECT button is held down for more than one second.

Figure 4: SRAM displayed on lcd screen