**Explanation of the Code for the Emulator:**

The program for the Emulator consists of two major threads:

1. **Main Thread:** Responsiblefor initializing bit values of the sensors based on the initialized encoder values and communication between the TCP Client and the TCP Server.
2. **Emulator Thread:** Responsible for changing the values of different bits based on the inputs received from the TCP Client.
3. Global static variables for sensor1,sensor2,encI,encD ,encS , dwellTime , totalTime ,errorCode are created so that they can be accessed throughout the class “Program”.encI is initialized to 1000 while other variables are by default initialized to 0.
4. A global static constant array named “indexerVals” to store the values of all the different slots of the indexer.
5. **Main Thread**
6. **Socket Creation**:
   * 1. An endpoint is created combining the port no. and IP Address of the TCP Server Machine.
     2. A TcpListener is created to enable the TCP Server to listen to any incoming requests from potential TCP Clients.
     3. The TcpListener is then binds with the endpoint to enable communication.
     4. Once a TCP Client’s request is encountered ,the TCP Server accepts the same and a handler is created in order to handle the communication process with the TCP Client.
     5. The handler is used to get the common stream through which the TCP Server and TCP Client would communicate with each other.
     6. A buffer is created in order to store the data (that would be sent as bytes) sent by the TCP Client on the common stream.
     7. The data stored in the buffer would then converted to the string datatype for interpretation.
7. **Communication:**
   1. During the process of communication, the TCP Server reads data from the buffer sent by the TCP Client.
   2. First calls the HB function.

**HB** -Sends current values of the sensors and encoders.

* 1. Then the Emulator thread is created for the interpretation of the instructions sent by the TCP Client on the common stream and then subsequently changing the bit values based on the updated encoder values.

1. **Emulator Thread**
   * 1. First the format of the received data is checked using Regular Expressions to confirm that the received data is an instruction. If an instruction is confirmed further checks are made.
     2. Each instruction starts by first setting the 1st and 0th bit of sensor2 i.e “**CommandInProgress**” and “**Moving**” bits and on instruction completion the same bits are cleared to indicate that the instruction in completed.
     3. **Source Out** bit(bit 0) of sensor1 is set when any one of the Source or Dummy reaches the Sout position of 1585 encoder counts.
     4. **Source Overshoot** is set when the Source reaches 21,240 encoder counts i.e 4248 motor counts.
     5. **Dummy Overshoot** is set when the Dummy reaches 21,240 encoder counts i.e 4248 motor counts.
     6. **EI-** Reponsible forerasing the encoder values of the Indexer.
     7. **ES-** Reponsible forerasing the encoder values of the Source.
     8. **ED-** Reponsible forerasing the encoder values of the Dummy.
     9. **MSFn-** Responsible for moving the Source out .After moving out it is checked if Sout position is crossed or reached. If this condition is satisfied ,Source Out bit is set