**Alexander Luna**

**Homework 1 Design**

**CS 360**

**Server**

The map called players maps the players name to the player’ id.

The int called secretNumber is the secret number the client will try to guess.

The int called SHARED\_MEMORY\_SIZE defines the size of the Ring Buffer.

There struct called CircularBuffer defines the structure of the Ring Buffer.

There are five functions:

1. int generateRandomNumber()

* Generates a random number between 0 – 255 that the Clients will attempt to guess. It returns an int, which is stored in int secretNumber.

1. void createMailSlot()

* Defines and creates the mailslot that is used to read the player’s name and id from Clients. The mailslot is closed once it reads a message from the Clients saying “quit.”

1. void createRingBuffer()

* Creates a semaphore used to control access to the Ring Buffer
* Creates the Shared Memory Object
* Creates Circular Buffer object
* Compares guesses sent by Clients to the secret number

1. void createEvent()

* Creates the event that is used to signal to the Clients to stop guessing

1. int main()

* Prints out the description of the game
* Calls generateRandomNumber() and stores the int in secretNumber
* Prints the secret number
* Creates a thread to run the createMailSlot() function
  + A thread is used here because we need to keep the mailslot alive in case any new users want to join the game
* Calls createRingBuffer() to create the Ring Buffer that is used to receive guesses from the client and compare them to the secret number
* Calls createEvent() to create the Event that is used to signal to the Clients to stop sending guesses
* Waits for mailSlotThread to finish before continuing. This is to ensure that this thread is not still running after main ends
* Iterate through the players map and prints out the player’s names and their ids

The Server has the following known bugs/errors/limitations:

* The messages sent by the Client’s mailslot are printed twice.
* The server does not generate the player’s id and sends it to the Client’s mailslot
* The server does not know which process sent the correct guess

**Client**

The struct called Player stores the player’s name, player’s id and a map mapping the player’s name to the player’s id.

The Player player object is used to store the player’s information.

There struct called CircularBuffer defines the structure of the Ring Buffer.

The int called SHARED\_MEMORY\_SIZE defines the size of the Ring Buffer.

The bool moribund is used in the while loop of the Ring Buffer and is set to true after the Event gets signaled, so the Client knows to stop sending guesses in the Ring Buffer.

There are six functions:

1. char generateRandomChar()

* Generates a random uppercase letter that is used as the player’s id. It returns a char, which is stored in player.playerId.

1. int generateRandomNumber()
   * Generates a random number between 0 – 255
2. void openMailSlot()
   * Defines and opens the mailslot that is used to send the player’s name and id to the Server
3. void openRingBuffer()
   * Creates a semaphore used to control access to the Ring Buffer
   * Creates the Shared Memory Object
   * Creates Circular Buffer object
   * Calls generateRandomNumber() to send a random guess to the Ring Buffer
4. void waitForEvent()
   * Creates Event that is used to wait for the signal from the Server
   * moribund is set to true so the Client knows to stop sending guesses in the Ring Buffer
5. int main()
   * Prompts the user via the command line for the player’s name and stores it into player.playerName
   * Calls generateRandomChar() to store the random uppercase letter in player.playerId
   * Prints out the size of player.playerId, which is one byte because it is a char
   * Inserts the player’s name and player’s id into the player.playerNamesAndIds map
   * Creates a thread to run the openMailSlot() function
     + A thread is used here because we need to keep the mailslot alive to send the “quit” message so the Server’s mailslot knows the shut down
   * The system pauses to allow for more users to join before the game before it begins
   * Creates a thread to run the waitForEvent() function
     + A thread is used here because the Client needs to wait for the Event to be signaled by the Server whenever the correct guess is sent
   * Creates a thread for the openRingBuffer() function
     + A thread is used here because this function requires the waitForEvent() to finish and change moribund to true
   * Waits for the threads to finish before continuing main
   * Prints that one of the Clients guessed the correct number

The Client has the following known bugs/errors/limitations:

* The Client generates the player’s id, not the Server
* The Clients do not know which Client guessed the correct number
* Sends duplicate random numbers as guesses

**Screen Captures of the game in operation with at least 3 players**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**