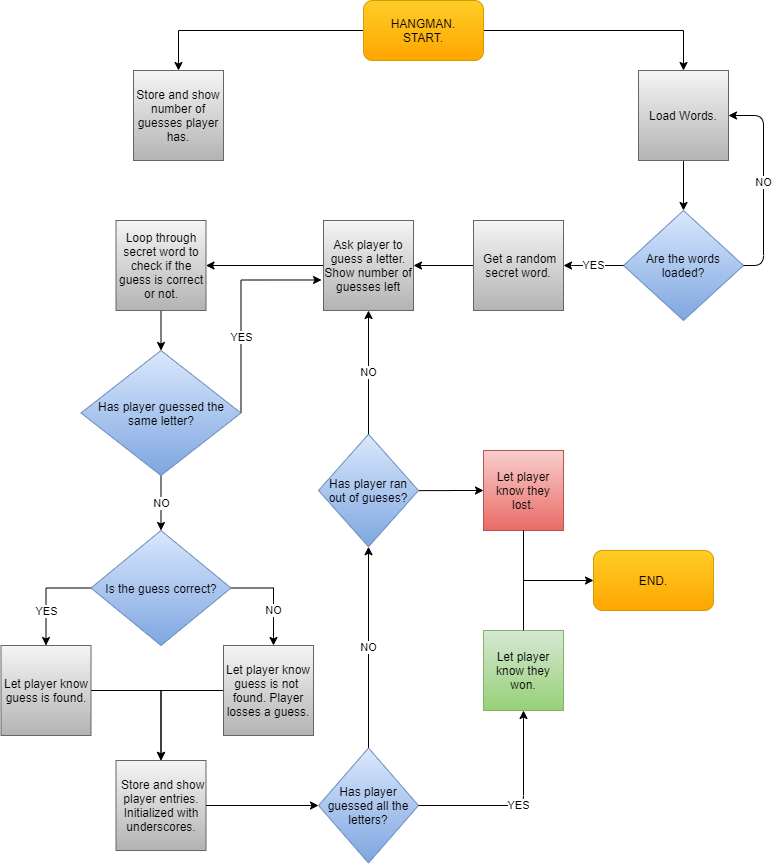
**Q1: Create a flow chart - a graphical representation of the sequence of steps needed to implement the Hangman algorithm.** Flow chart needs to be inside the lab-write up and not hand-written. For additional information and details on flow charts, see the following sites: http://www.computerhope.com/jargon/f/flowchar.htm http://users.evtek.fi/~jaanah/IntroC/DBeech/3gl\_flow.htm



**Q2: Describe how you stored the user entries for the letters.**

The user entries were stored by using the char data type to define the variable named guess. The variable guess registered each letter that was guessed one after the other as a character. This was possible by using the scanner input with the dot property and the charAt() method together at index zero to capture the user input at its starting index. When a user input is entered, it is then finally stored in a string array called showGuess. This array is also equal to the length of the secret word and was initialized with underscores to signify that it is blank. If a user enters the wrong guess, it displays those underscores and visually lets the user know how many letters to guess.

**Q3: What are the methods that your group created in your code? Describe each method in detail and why you chose to create each particular method.**

Note: Each array length in any of the following methods were equal to the length of the secret word. The methods created were:

**Grabber:**

This is a method created to return a random integer which was stored in the integer variable rand. By using the random type from the math class, I was able to set up a range from 1 to 55909 equaling the amount of words from the word list with each number between the ranges representing a word at that index.

**Word:**

This is a method created to receive an integer value and return the string of the secret word with the help of the Grabber method. This Word method is critical for the game to run because without a secret word there is no Hangman game to play. The secret word was then stored in a string variable named secretWord; to be used as one of the core components of my code. The variable was used to call upon this Word method, which in turn called on the Grabber method. The random value from the Grabber method, which acts as an index of a random secret word, was passed through this Word method to capture the secret word from the helper method loadWords within a for loop.

**blankArray:**

This is a method that returns a string array. It has 2 parameters, a string for the secret word and a char for the guessed letters. When they are passed to the method, a new string array called partialArr stores the correct letters that was guessed at its index. I needed this method in order to store the partial letters that were guessed correctly into the string array showGuess. The variable partialArr would be initialized with an underscore as well at whatever index that did not match the user entry. This is the method that enabled me to check throughout the secret word to see if the user entry is found in the word. This was done by calling on the helper method stringToChar and passing through the string of the secret word. Since the helper method returns a char, it was then possible to use an if-statement that checks if the guessed letter, which is also a char variable, is equal to any of the characters in the word at its respective indexes, using a for loop.

**stringArray:**

This is a method that returns a string array of the secret word. It has a string parameter that receives the secret word and converts it into an array of that word. I created this method in order to pass it through to another method that receives a string array.

**errorCheck:**

This method receives a string array and a char, and returns a string. This method was created to check for errors when a user enters the wrong guess or a letter that has already been guessed correctly before. In this method there is a string variable named checker that is initialized with the string Error. If any of the letters in the string array equals the guessed letter, then checker is updated with the user’s guess letter and returned to notify the user that such letter has been guessed before. If the user’s entry is not found in the string array, then checker remains as is and returned as Error, letting the user know that such letter was not found in the word. A letter in the string array is able to equal the guessed letter that is a char because it has been concatenated with an empty string, converting the guess to a string. The string arrays used to pass to this method, were the string array acquired from the stringArray method for no letters found and the showGuess array for repeated entries.

**winCheck:**

This method returns a Boolean and receives one parameter, a string array. The method is called as part of the arguments in the while section of the do-while loop. The string array showGuess is then passed to this method. In the method there is a Boolean variable named winChecker and is initialized as true. I needed this method to know if the player has accurately replaced all the underscores in the string array showGuess with the correct letters in the secret word. If any index of showGuess is equal to an underscore then winChecker is updated to false and returned to continue the game. If showGuess is filled with all the correct letters and no underscore, the method returns true and notifies the player that they have won.

**Q4: What was the most challenging part of this project for your group?**

The most challenging parts of this project were at the beginning and at the end of building the project. At the beginning it was figuring out the best logical ways to call the secret word from the helper methods, and at the end it was figuring out how to make the game know that the player has won when all the letters have been guessed correctly. This helped me in fleshing out my understanding of method calling and rehashing what I have learned so far in terms of logical operators and how to use them efficiently.

**Q5: What did your group learn/find the most useful by doing this project?**

What I learned and found useful for doing the project was being able to think critically and at times out of the box. Being able to ascertain what the compile error messages were trying to tell me and try to implement viable solutions.

**Q6: What was the most fun aspect of doing this project?**

The most fun part about this project was the self-awareness of ingenuity at play when something clicks or is figured out. It is enjoyable anticipating when I am closer to my goal of building the Hangman game.