Nicolette McDonald Final Project Design

Documentation- NONE.

**Requirements**

* Import dictionary file and select random word (20%)
* Allocate dynamic memory for inputted word and string showing correctly guessed letters **Dynamic Memory**(5%)
* Fill Guess string with ?s (5%)
* Display game image (10%)
  + Hangman picture
  + Display status of unknown word **String manipluation**
* Ask player to enter guessed letter (5%)
* Compare letter to each letter in word **Processing of array data**(10%)
* If the letter is in the word, fill it in the guessed string and add one to the count of correct guessed (15%)
* If the letter is not in the word add one to the count of strikes and update hangman image (15%)
* Check to see if hangman is complete or word is guessed (5%)
* Repeat steps 5-10 until hangman is complete or word is guessed **Recursion** (5%)
* Display end game message (5%)

**Functional Decomposition**

* main
  + ask for player 1 to input word and # of letters in the word
  + checkWord()
  + prepGuessWord()
  + turn()
  + endgame()
* checkWord()
  + make sure word is in dictionary file
  + allocate dynamic memory for word
* prepGuessWord()
  + allocate dynamic memory for string guess
  + fill string guess with ?s
* turn()
  + display()
  + ask player 2 for a letter
  + check if letter is in unknown word
    - yes-
      * add one to count of correct
      * fill in letter in the string guess
    - no-
      * add one to count of strike
      * draw no hangman
  + check4end()
  + repeat until check4end indicated game over
* display()
  + print correct hangman image
  + print guess string (completely ?s or partially filled in)
* check4end()
  + if strikes is greater than or equal to 6, end game
  + if correct is greater than or equal to # of letter in word, end game
* endGame()
  + if player wins, displays victory message
  + if player loses, displays game over message

