report.docx

Notable Obstacles:

1. The main obstacle that I faced arose when I tried to check the probe words for stars and planets. I didn’t know how I wanted to check the string, would I cut the stars and letters out? Would I rearrange letters? I struggled with the logic for some time, until I realized an easy way to get around this hump. I decided to change any stars/planets in the actual words to random capital letters. This allowed the loops to continue checking the strings without accidentally counting planets/stars multiple times for a specific element of a string.
2. Another major obstacle was determining how to calculate the final summary of data for the rounds. I ended up creating a void function that would do all the calculations with minimal information. The function required a count of what round the player was on, and the score an array of collective scores per round. From here the calculations were fairly straightforward.

Pseudocode:

1. int Main()
   1. Fill the array with the appropriate words
   2. Check that number of words is positive
      1. If not return 0, and say no words loaded
   3. Ask how many rounds to play
      1. Allow input for number
      2. If number of rounds is negative
         1. Output -1 and say rounds must be positive
   4. Loop number of rounds that player inputted
      1. Output round number
      2. Call function to play one round
      3. Call function to output round data
      4. Repeat as required
2. int runOneRound(const char words[][7], int nWords, int wordnum)
   1. Check if number of words meets specifications
   2. Find value of secret word
      1. Store the value into a string
   3. Output length of secret word
   4. While there is not an answer
      1. Ask for probe word
      2. Check if probe word is a word from the list
         1. If probe == the secret word
            1. And if 1st try

Return true and got it 1st try

* + - * 1. Else

Return true and got it on N try

* + - 1. Else
         1. Loop through probe word

If element i of probe=secret word

Increment star count

Set that element to capital S/T to represent the stars

* + - * 1. Loop through probe word

Loop through secret word

If element of probe == any element of secret word

Increment planet count

Set that element to capital P/T to represent the planets

Reset secret word (to w/o capitals)

* + - * 1. Print number of stars and planets
        2. Set planet/star counts to 0
  1. Else
     1. Print that the probe word must be between 4-6 lowercase letters
  2. Return the score

1. bool isWord(string input, int numWords, const char words[][7])
   1. Check if word between 4 and 6 letters
      1. If not, output warning
   2. Loop through entire array of words
      1. Check if probe word matches any word from the list
         1. Return true
      2. Else
         1. Increment counter
      3. If counter is the same as number of words
         1. Return false, means not in array
   3. Default, return false
2. void data(int rounds, char scr[], double counts)
   1. set float to 2 places after decimal
   2. loop through array of scores
      1. sum the scores
   3. average the sum of scores
   4. loop through array of scores
      1. check for scores smaller than min
         1. if smaller, set to min
   5. loop through array of scores
      1. check for scores larger than max
         1. if larger, set to max
   6. output summary of trial data