Atij Mahesh

Professor Smallberg

Computer Science 31

14 November 2022

Project 5 Report

One of the obstacles I overcame was finding the silver letters in the word, as I had to account for the gold letters already found and previously found silver letters. I did this by making an array of goldIndexes and silverIndexes and accounting for them when finding silver. I also ran into a problem in g++ that said I could not create variable length arrays. I solved this by creating arrays of constant size, and only utilizing set indexes of that size.

My project is relatively straightforward in the main method and first part of playOneRound, as it follows specs and catches weird cases and errors. In my gold and silver finding sections of playOneRound, I utilize arrays of size 7 (maximum word size and zero byte character) that are initialized to 0 and updated to 1 if there is a gold or silver on that index. My program then utilizes the values in this array to ward off any silver repeats, and pairs the letters “one to one,” as described in the spec.

| main routine:  *initialize constants, word list, and number of words  if the number of words are less than 1, print an error message and terminate program  if the number of words are more than the maxwords constant, print an error message and terminate program  start game, ask user how many rounds they want to play and input number  if the rounds are less than zero, print an error message and terminate program  initialize maximum and minimum tries, and total try count  game loop for number of rounds  get wordNum using randInt  initialize the hidden word  tell user hidden word length  play one round  if playOneRound is negative, print an error message and terminate program  print message in response to the number of tries  calculate min and max, and print final statement with min, max, and average*  playOneRound  *if the input has number of words or wordNum as negative, or the wordNum is greater than or equal to number of words, return -1  initialize try count  start loop  initialize constants, hiddenword, and length of hidden word  get probe word and length  if the probe word isn't within specs or isn't in the word bank, print an error message and restart loop  increment try count   if the probe word and hidden word are equal, break out of the loop  initialize and set up goldCount, silverCount, goldIndex, silverIndex, shortest and longest word lengths  search for golds loop  if the probeword and hiddenword are equal at the same place, increment goldCount and set the appropriate index in goldIndex to 1  if the probeword is shorter or equal to the length of hiddenword  silverIndex loop  if the position is a gold, skip to next character  iterate through the probe word  if at that index, goldIndex or silverIndex are 1, skip to next char  if letters match on that index, increment silvercount and put 1 in that silverIndex, and break out of the inner loop  if the probe word is longer  similar silverIndex loop as above, but switching the indexes iterating through to prevent going out of bounds  print out the gold and silver count   return try count* |
| --- |