For this program I chose to declare both the word list and the dictionary itself dynamically in the heap. The word list because the amount of data wouldn’t have fit on the stack, and the dictionary because I would have to reference it from the spellchecker. I chose the same move for my token list for tokens gained from an input line because it is possible that the number of tokens found on a token list could exceed stack limits.

I chose to declare the spellchecker as a separate entity with no connection with the dictionary except that the dictionary is passed into its check() function with a DICTIONARY argument.

I also chose to add a list of prime numbers for the double hasher to access within the hash table class. Along with that is an addprime() method to achieve the listing of the prime numbers from the prime number text file. This text file is reformatted to have each prime number on a separate line.

The double hash return a number that is R – (i mod R) where R is a prime number smaller than the table size and i is a number that is given to the function starting at 1 and increases until it the outside loop that calls the double hash finds an empty spot in the array.

My input files are named primeNumber.txt, program4input.txt, and program4wordlist.txt. They are all augmented into the program except program3input.txt, the command must be supplied by the user of course.