1. This while loop will continue to run while we do not reach the end of the file and while the next character that is read in is alphabetic (a-z and A-Z). If both of the conditions are true, the loop stores the current next_character into the next_word array at the index specified by spot. The spot variable keeps track of the position in the next_word array. After each iteration, spot is increased to move to the next index in next_word. The loop will continue until either the end of the file is reached or a non-alphabetic character is encountered. Once the loop is completed, next_word will represent the extracted word from the file.
2. If next_char is replaced with islower(next_char), next_word will store all the characters that have been read as lowercase letters.
3. To determine whether one C-style string is a prefix of another C-style string in a C++ program, we first have to compare the characters of the two strings one by one, starting at index 0. We have to iterate over the characters of both strings and check if each character in the first string is equal to the character in the second string. If one of the characters in the first is different to the character in the second string or if one of the strings reaches the null terminator we stop the iteration. After iterating through the characters we check if the second string has reached the null terminator. If the second string has reached the null terminator then the first string is a prefix of the second string and we return true. If the second string did not reach the null terminator then the fist string is not a prefix of the second string and we return false.
4. To check if one C-style string is a substring of another C-style string with the help of the is equal() and is prefix() functions, we first iterate through the characters in the first and second string, while the first string doesn’t reach the null terminator. We then check to see if the second string is a prefix of the first string, if this is true, and the two strings are not equal, and the second string doesn’t reach the null terminator, we return true. In the case of ‘a’ being a substring of the first string, we check that the strings are not equal, we check if the

first string has an ‘a’ character and that the second string is only an ‘a’, if true we return true. We still need to check and see if string one contains string two, to do so we have to: we have to iterate through the characters in the first string we do this with a while loop, where the first string does not reach the null terminator. We need a second while loop that checks that the second string does not reach the null terminator, where the indices i + j in the first string do not reach the null terminator, and that the second string at j and the first string at i +j are equal, if this is true we then check that the second string at j is not a null terminator, and that the strings are not prefixes and are not equal if this is true then we have found the substring.