My program begins by storing all dictionary words in a hash map and asks the user for a password file to read. Very little error handling is provided for reading the file as it was not a project requirement. After the user accounts are read, they are parsed into a two-dimensional string array. The format is as follows:

[{username, hashedPassword, userID, groupID, userInfo(full name), users homeDir, cmd shell}, {username, ….}].

After storing the user credentials in the array, the dictionary attack automatically begins. There are several helper functions that make up the dictionary attack. To provide an in-depth explanation for each would be arduous and redundant; so, I will keep things simple and explain one of them as the rest are the same in a sense. The string manipulations, or *mangles* are all that change from function to function. For example, the duplicateString() function takes in a salt from the first two characters of a single users encrypted password. The salt is applied to a word from the dictionary that is repeated twice with the crypt() function that is included in the assignment code. If the word *key* is input from the dictionary, *keykey* would be the final manipulated string, just before the crypt() function applies the salt.

Each of the functions perform a different mangle, I will provide a list of each at the end of this document with a few examples. The dictionary attack function calls all of these mangle functions within itself and stores each mangled word from the dictionary in a passwords list, followed by a check after each input word is mangled by all of the helper functions. The check is to determine if there is an encrypted value in the passwords list that matches the users encrypted password. If so, the value contained within the password list contains the plain text version of the encrypted password.

In order to crack each password, the dictionary attack function has to be applied to every user account, and every word in the dictionary has to be mangled so that a list of all possible combinations is checked in the passwords list.

It should be noted that when each account is passed into the dictionary attack, the users account and name is prepended to the front of the dictionary so that the accounts name and other relevant data can be manipulated with each mangle. Below is a list of each mangle with a short description for each:

1. A check on all words in the dictionary with no string manipulations.
2. A check on all words in the dictionary reversed.
3. A check on all words in the dictionary with a single prepended character for all ascii combinations (ascii 32 -126 for all combinations in this document).
4. A check on all words in the dictionary with a single appended character for all ascii combinations.
5. A check on all words in the dictionary with the first character removed.
6. A check on all words in the dictionary with the last character removed.
7. A check on all words in the dictionary with the first character capitalized, followed by the reversal of the string.
8. A check on all words in the dictionary with the first character capitalized, followed by the reversal of the string and then the removal of the first character in the string (originally the end of the string).
9. The duplicate string function from the example provided in the algorithm description mentioned earlier in this document.
10. String reflection. Ex: stringgnirts
11. String reflection reversed. Ex: gnirtsstring
12. All uppercase.
13. All lowercase.
14. A string with the first letter capitalized.
15. A string with the first and last letters capitalized.
16. A string with the first letter capitalized and an additional character added to the end for all ascii combinations.
17. A string with the last letter capitalized.
18. A string with the last letter removed.
19. All capitalized letters with the first letter lowercased.
20. All capitalized letters with the first letter lowercased and a new character appended to the end for all ascii combinations.
21. Toggle Case. Ex: AbCdEfG
22. Toggle Case reversed. Ex: aBcDeFg
23. A string with the last character replaced with one new character for all ascii combinations.