Jeremy Giese
P2 Report
Dr. Wahjudi

My password crack application requests the file containing the passwords from the user and if the filename entered doesn't exist continues to ask until a real file is given. After the file is read each line is checked in order using normal mangle methods, the dictionary and the username and last name of the user. If the password is not found within the normal mangle methods then the program attempts to append characters to the username, last name and then the dictionary again. If a password is found then that is returned to the main method, otherwise it returns a default state which signify's that it did not find the password.

The method called to test for a password contains calls to two while loops and only calls the second if the first cannot find the password. Within the first the program performs up to three mangles upon each word in the dictionary and the users last name and username. In the second, if called, it appends and prepends characters as well as uses at most three mangles. After each mangle the hash is found and tested for equivalency to the user's hash. If the hash is found to be equal then the string used to find it is returned.

The mangles used in my program are based on the ones within the document assignment. This means I have reverse, duplicate, reflect, lower, upper, toggle, capitalize, ap/pre pend and delete methods. I have a few other methods but they are not implemented because they did not yield any more results. Any method that could have more than one output accepts the string and an int to decide which result you want. For example the reflect method accepts any number but 1 to mean gnirtsstring and 1 gives you stringgnirts instead. This allowed me to copy and paste methods multiple times and lowered the time to write the code but decreased readability because I forgot which number did which function between writing sessions.

I realize that loading all passwords into a structure to test for them all at the same time would have increased the speed, however I would have to change my methods too much to accommodate for it. In its current state my program's worst complexity is O(n^2*m) where m is the number of characters to be appended to a string and n is the length of the dictionary+2. I moved the append and prepend functions outside of the primary check which sped up the algorithm. This resulted in the overall time of the program, with the given files decreasing by a factor of 2. However, given a file filled with characters preprinted and appended I feel it could cause the program to take upwards of 3 minutes.