Mehmet İPEK
2019555404

I worked on this assignment in parts, I divided it into 5 parts. First, I designed to search for passwords, then sort passwords, hash them, transfer them to the linked list and determine their power, and finally create a menu in the main function. In my report, I will briefly explain what I have done in this order.

1.Searching: First of all, while designing the searching function, I thought it was inefficient to transfer all 100000 passwords to the memory, so I created a temporary string, read the password from the file and compared it with the given password in the void password_searching(); function. I did this as a sequential search because If I had used binary search, I would have had to use more memory for binary search and sort the passwords.

2.Sorting: First, I copied 10000 passwords to a struct array and made it ready for operation in the password_reading_for_sorting(); function. I designed two functions as void sort_data_descending(); and void sort_data_ascending();, because calculating these ones with struct arrays is faster and easier to implement than working with linkedlists and reversing them with stacks. I used bubble sort in my function because although bubble sort is not effective for memory and time, it is very easy to implement.

3.Hashing: First, I designed a hash table initialization function (void initialize_hash_table();).This function resets our table and then transfers the passwords that we are going to hash. I determined the table size to be 4987, which is a prime number based on my experiments to give the least collision. I used int hashing_password(char* password); function to get a hash value from the password. I have transferred the struct containing these passwords to the table with the insert_table(); function. I applied quadratic probing as the collision resolution technique. Then I designed a search function to search in this table(void search_password_hash();).With this search function, I determined whether the hash value of the given password is in the table, and with the collision variable I kept the count of the collision.

4.Linked List: First, I copied a link list of passwords with the void insert_password_list(); function and while copying this list, I calculated the strength of the password and placed it in the list. I determined the strength of the passwords with int rate_password_list(char* password); function considering criteria such as the presence of capital letters in the password, the length of the password, and the majority of letters in the password. After all passwords were put into the list, I sorted this list with the bubble sort. The main reason I chose bubble sort was that it was easy to implement. I thought there was no need to complicate the code to sort 1000 passwords. I printed the first 10 passwords of the list which are sorted in ascending order. Then, to print the last 10 passwords, I reversed the list with a stack. I pushed all the passwords with the int push(PASS_L* data); function to stack and remove the first 10 elements from the stack with the void pop(); function and print them.

5.Menu: I wanted to design a menu in this program because I thought it would be a waste of time to run the program separately for each process. I designed the menu with a switch case and a while loop. I used system functions in the menu. system(“pause”); function is for waiting and taking an input from user and system ("cls"); function is for cleaning the console.

Mehmet İPEK

2019555404