CSE3033 - Operating Systems Assignment 1 Report
Fall 2020

Ali Reza Ibrahimzada	Huzeyfe Ayaz	Sena Dilara Yangoz
Student ID: 150119870	Student ID: 150119700	Student ID: 150119706

Q1. Our approach for solving Q1 is to maintain two arrays when running the program. Since there are 10 possible valid numbers available in the given argument, our first array is initialized with ten 0s. The purpose of the second array is to store invalid numbers. Then we read each line and check if the number is valid. If it's not, we append the number to the outOfBounds array. Otherwise, we increment the frequency of that number in the frequency array at index number. After lines are finished, first we show invalid numbers and then print the histogram.

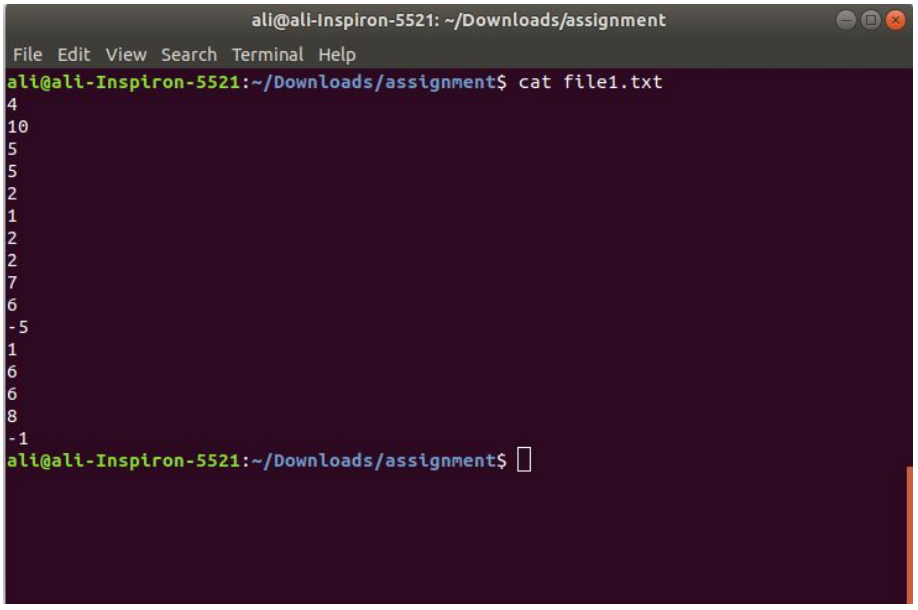

Figure 1: Test file which will be used as argument for Question 1

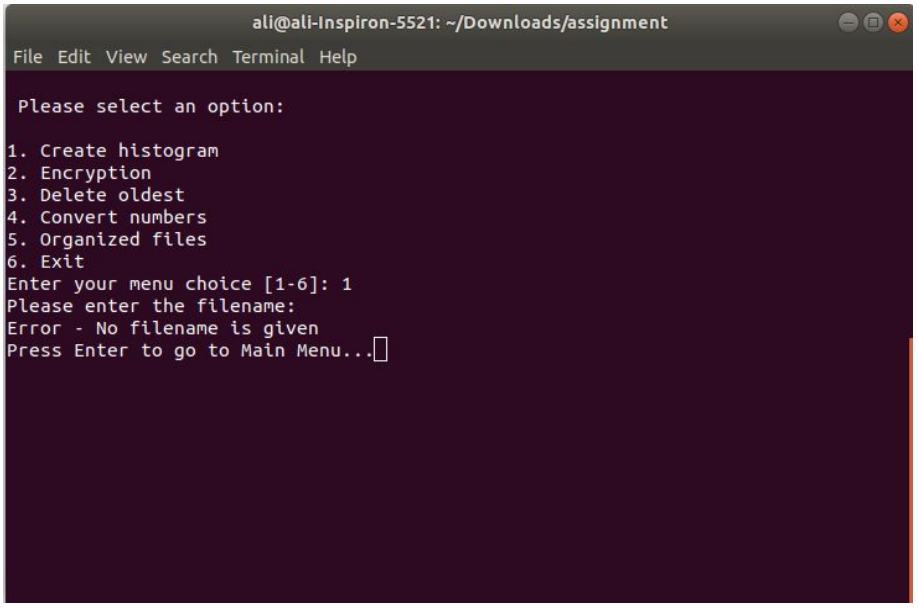

Figure 2: Error given by the program when no arguments are given

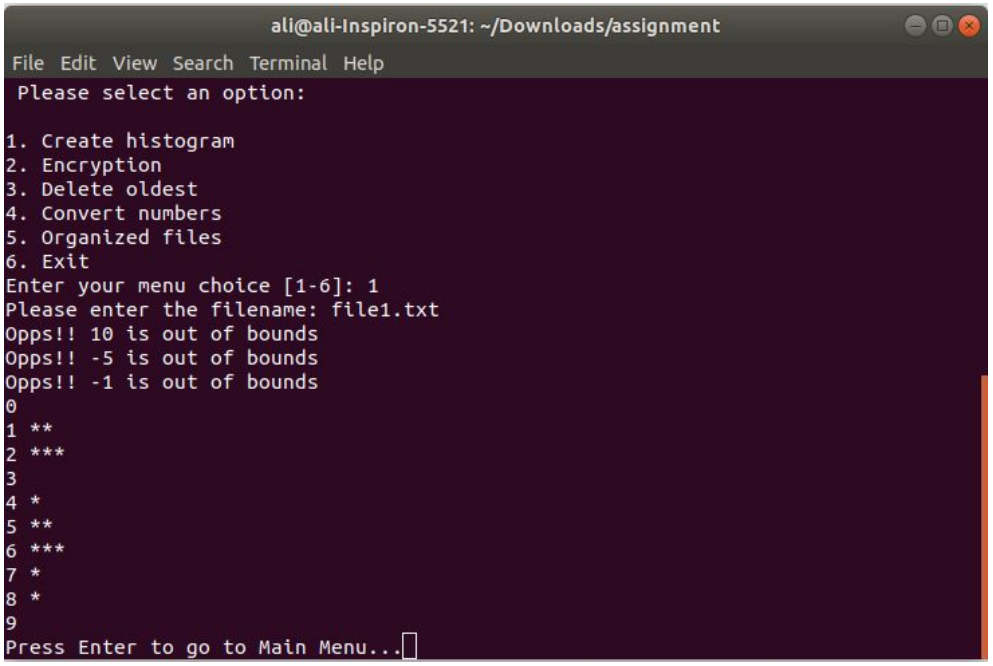

Figure 3: A successful run of Q1 using main menu

Q2. Our approach to solving Q2 lies in the array we create for lowercase english alphabet. Each character gets stored in order and we use their indices to create the cipher. In the string and number provided as arguments, we loop over each character, get its index and add the corresponding number to it. Then we use that sum as an index to find the cipher character. In case the sum is bigger than 26, we take a mod so it could start from the beginning. After finding the corresponding cipher character, we append it to a string and echo it at the end.

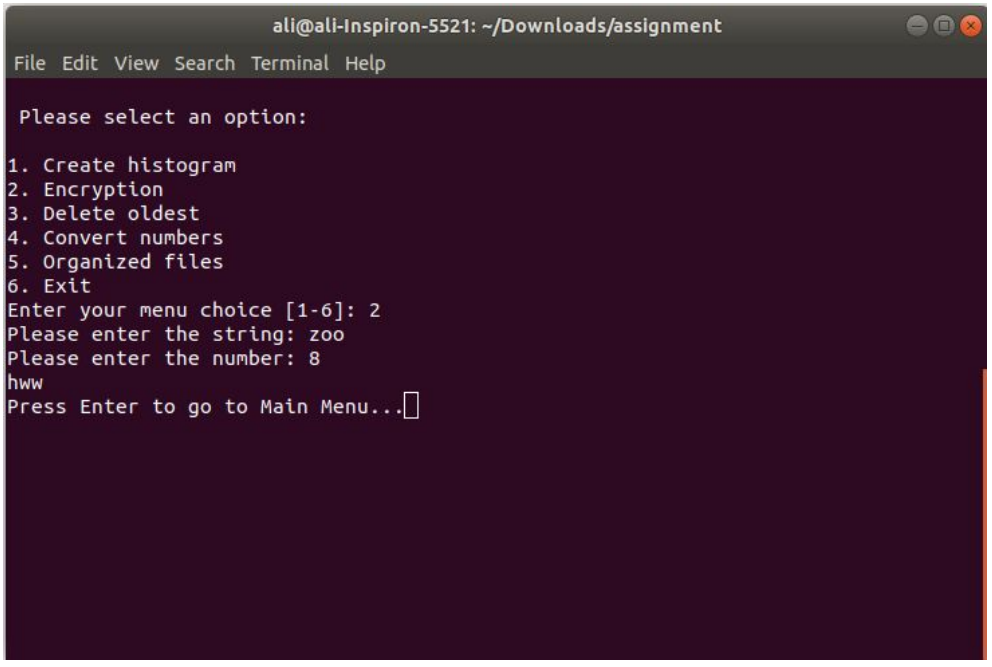

Figure 4: A successful run of Q2 with a single number

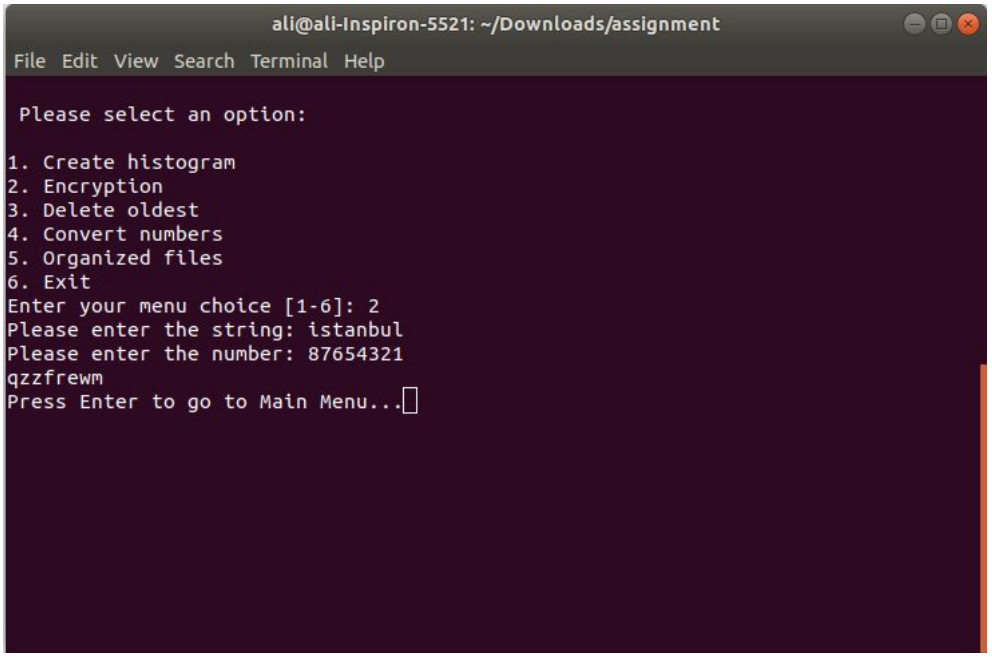

Figure 5: A successful run of Q2 when lengths of string and number are equal

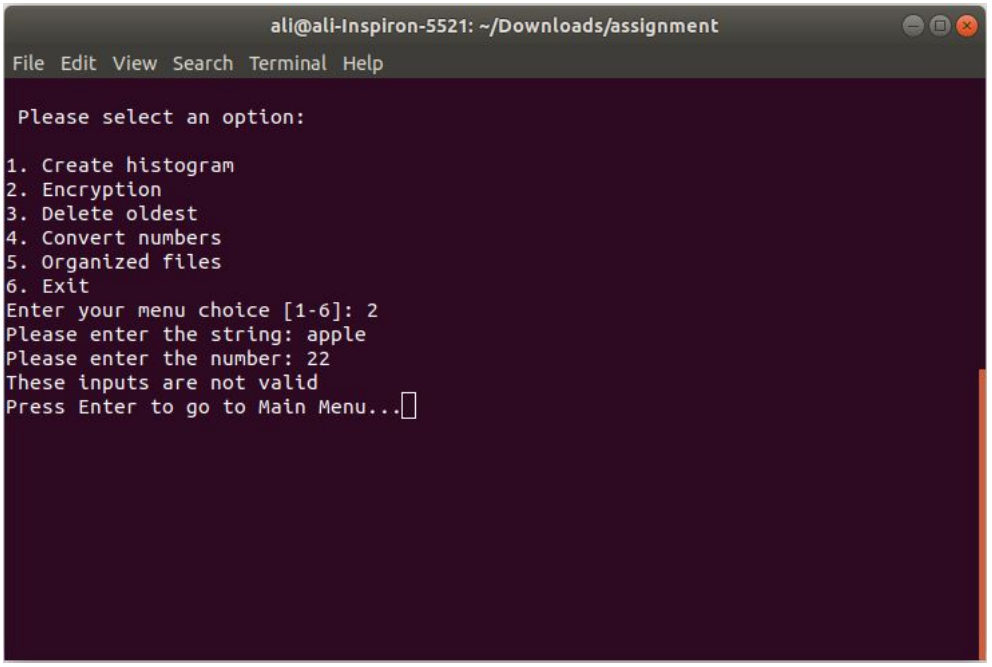

Figure 6: An unsuccessful run of Q2 because of invalid arguments

Q3. In the solution of the third question, first, we check if a path is given, if it is not given then we find the oldest file in the current working directory and ask the user if he/she wants to delete the oldest file. For the answer of the user, because we want the answer not to be case sensitive, we accept either y or Y and n or N as valid answers. But if the user types something other than these, the warning message, "Invalid Input", comes up. In case there is a path given, first, we check if the given path is valid or not. If it is not then we make the necessary warnings, if it is valid, then we find the oldest file in the given path and ask the user if she/he wants to delete the oldest file. And we repeat the operations we did in the first part for the deletion process.

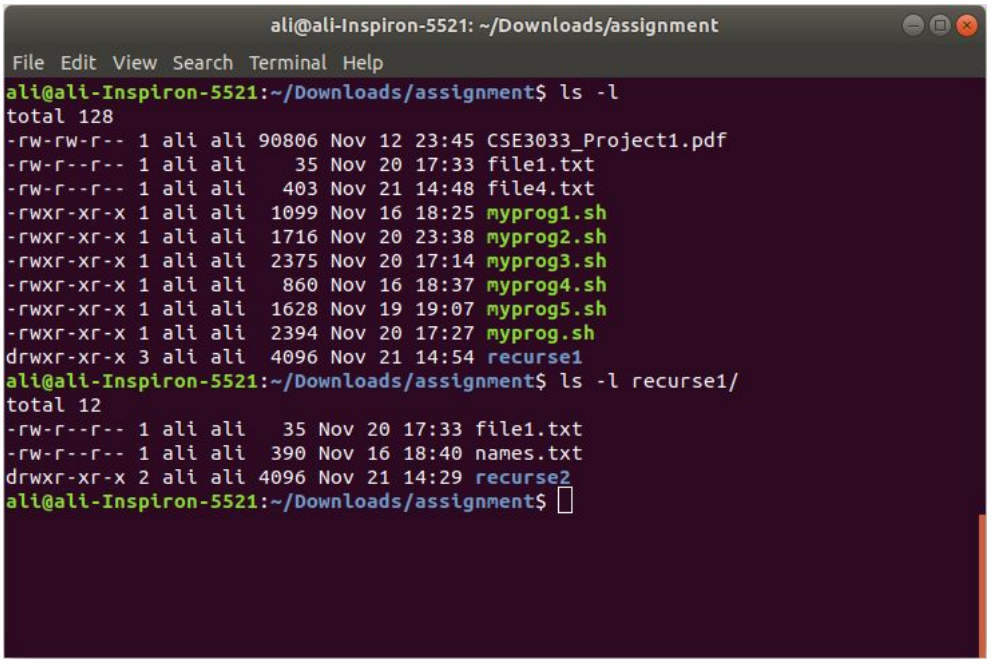

Figure 7: Directory structure before running Q3

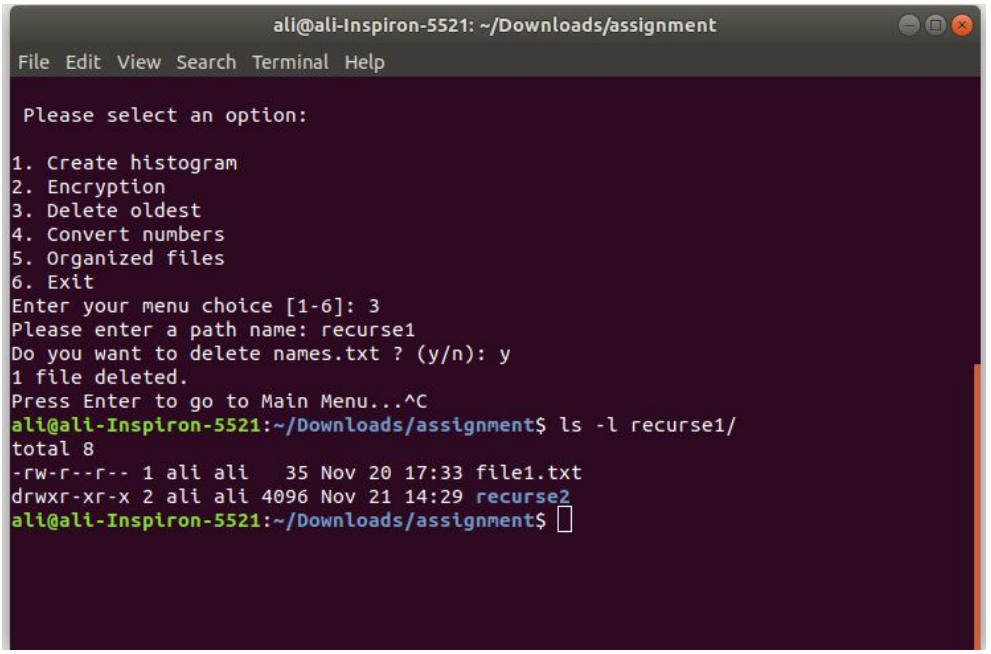

Figure 8: Successful deletion of the oldest file when a path is given

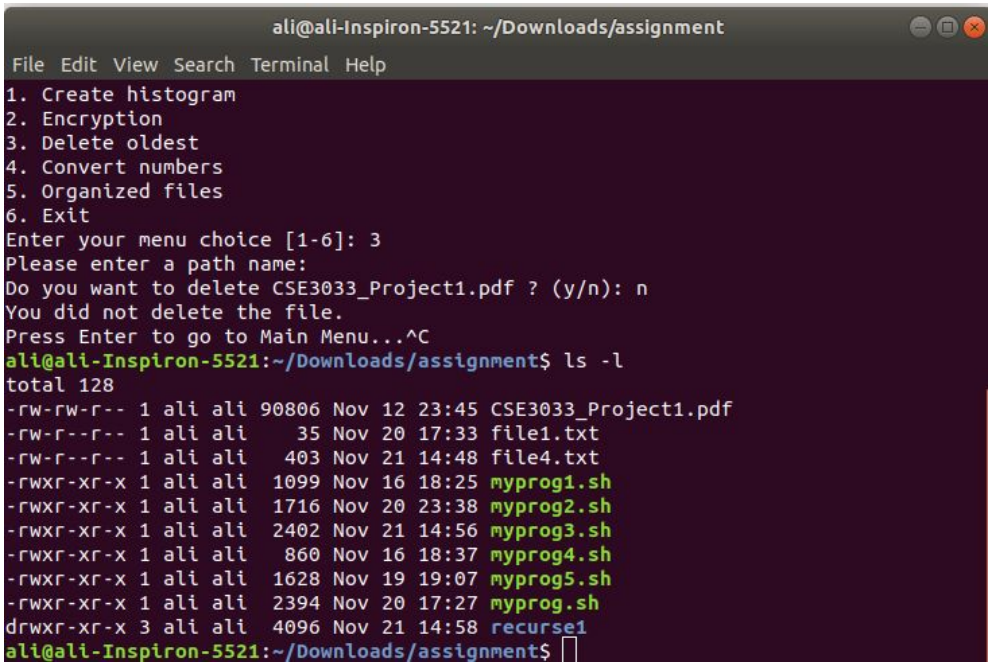

Figure 9: User can choose to not delete a file

Q4. For the solution of this question, we have started with argument checking in order to replace numbers with corresponding string representations in files, properly. Then, we have created an array of strings of digits. To achieve each digit in the array, we placed them properly in their indices. For example, zero is placed in the 0th index while nine is placed in the 9th index. This approach helps us to achieve each string representation of digits easily. We are keeping the new text in a variable and we have used regular-expression while tracing the old text in order to check whether there is a number in words or not. While looping on the text we have added the words directly to the new string variable if it does not contain a number. Otherwise, the program will give the digit as an index to array and fetch corresponding string format in order to append it to the new text variable. At the end, “Task completed!!” message will be printed to console and new text will be overwritten on the existing file. Please see Fig.10 and Fig.11 to get better intuition about the input and output.

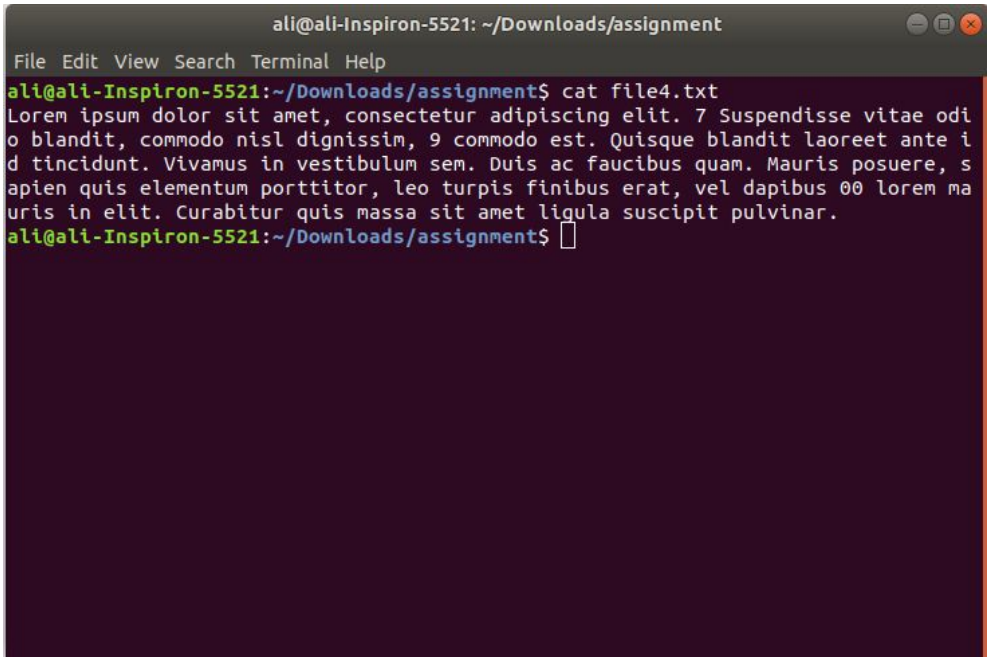

Figure 10: A sample .txt file to be given as argument for part 4

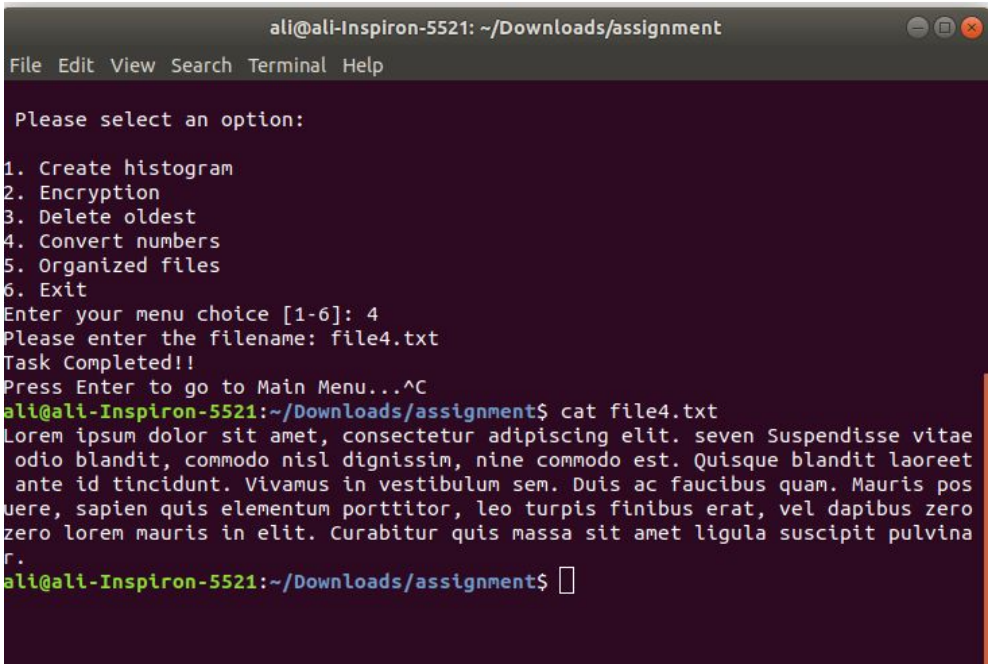

Figure 11: Successful run of part 4

Q5. When solving Q5, we first start by checking if the -R (case sensitive) option is given or not. If there is no -R option, we check for possible errors in wildcard. Then, we check if the wildcard matches any file in the current working directory. If so, we create the “copied” dir and copy the eligible files. In cases where -R and wildcard are given by the user, we first check if -R is entered properly. If it is correct, for all possible directories under the present working directory, first we change directory and then check for possible matching files. If there are any, we create a “copied” dir and copy the matching files. If there are no matching files, we let the user know and pass on.

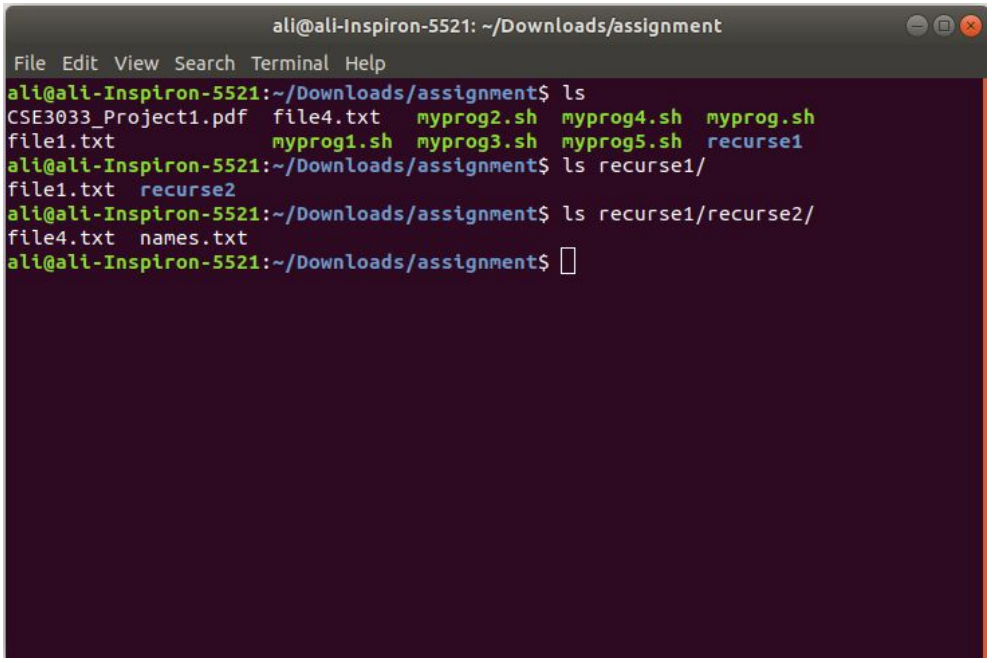

Figure 12: Directory structure before running the program

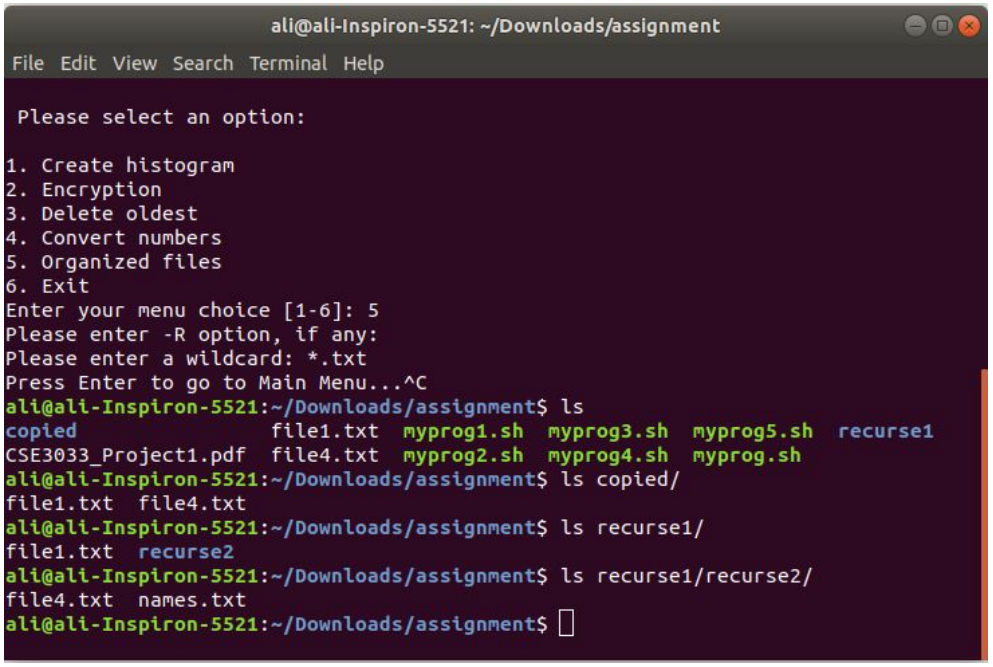

Figure 13: Directory structure after running the program without -R option

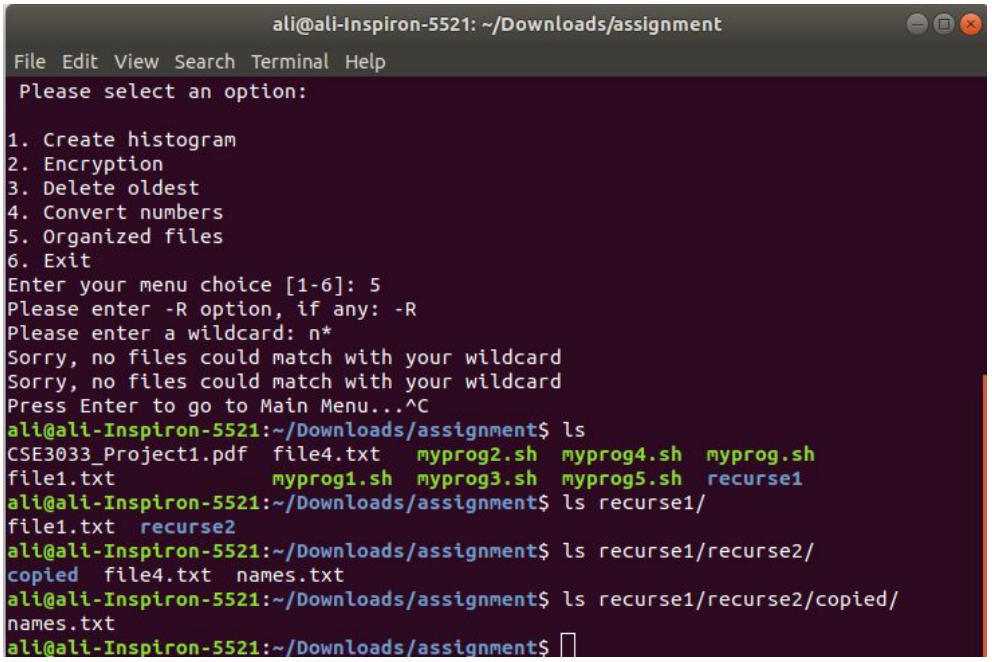

Figure 14: Directory structure after running the program with -R option