Morse Code Translator

Yezhen Wang

March 24, 2018

**Assumptions and Limitations:**

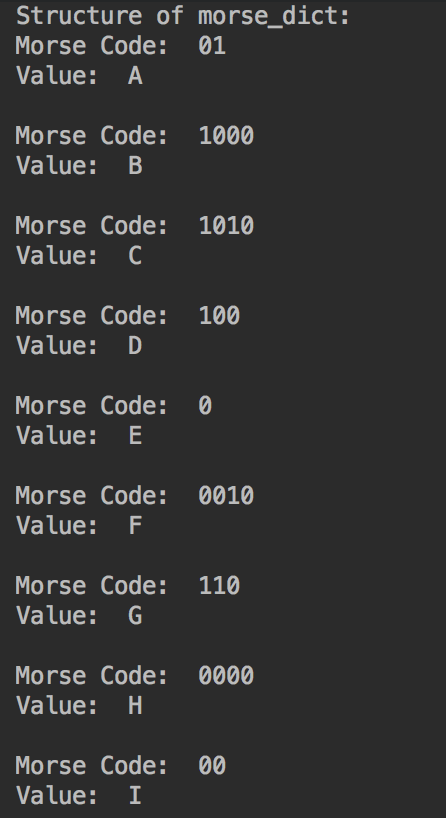
1. Since the program was implemented with functions, “\_\_name\_\_ == “\_\_main\_\_” was used in Task1.py, Task2.py, and Task3.py to make them executable.
2. The executable part in Task2.py and Task3.py was not used in Task4.py. Instead, the functions declared in Task2.py and Task3.py were called in Task4.py to remain the programming structure and logic of Task4.py.
3. Regular Expression **“^([01]+\\*?)+$"** was introduced to limit the user actions, it will only take inputs with 1, 0, and \*, and following the pattern of **“digit(s)\*digit(s)\*…(\*)”**. Asterisk is not allowed as the beginning of the input but allowed as the end of the input, and one and only one asterisk is allowed as the delimiter.

**Instruction for Execution:**

Task 1:

Simply execute the script, the structure of the dictionary contains the Morse code translation rules will be displayed in the console.

*Sample Output:*



Task 2:

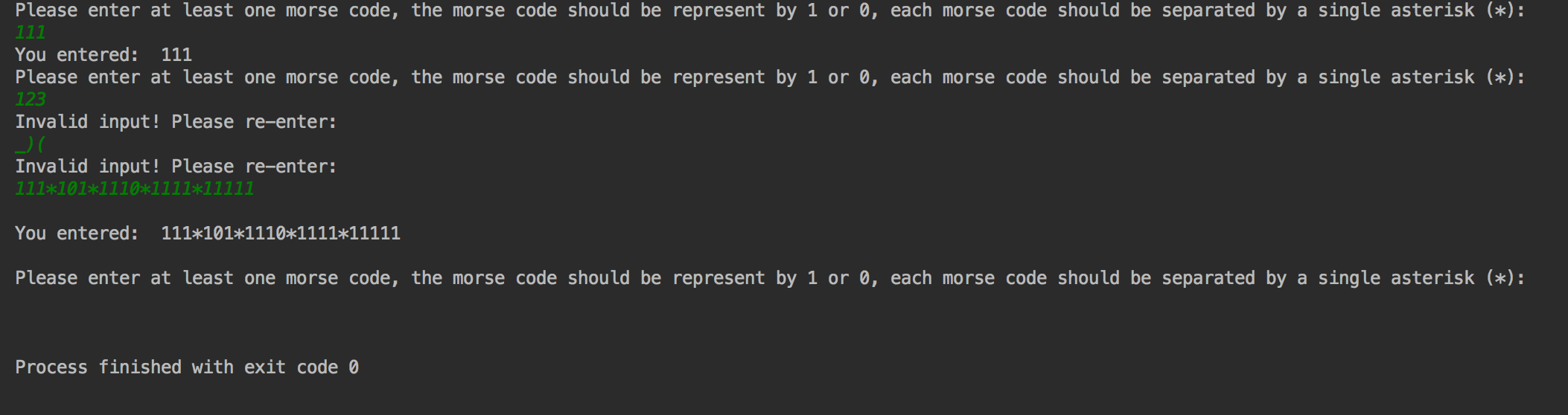
The program will prompt for user input after the file is executed, the user is allowed to keep enter contents until an empty input is received which is considered as the sign of end of input.

Regular expression is introduced in this stage to limit the user action. The allowed inputs from user are limited in 1, 0 and \*, plus user has to follow the pattern of  
 "digit(s)\*digit(s)\*digit(s)......(\*)"   
which means one and only one asterisk is allowed as delimiter.

User input is limited as shown in the prompt (only 1, 0, and \* are allowed).

The program also displays the content received from user every time user entered something and hit "Enter”.

*Sample Output:*

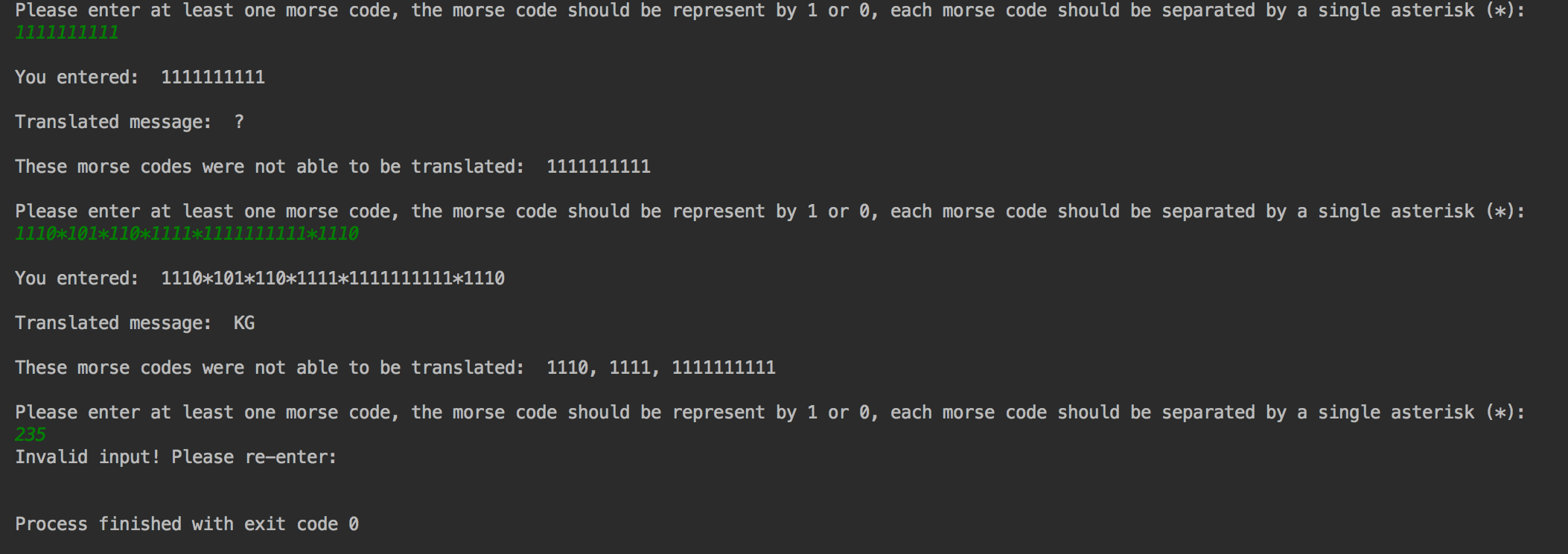
**

Task 3:

Just like Task2.py, the program will prompt for user input after it is executed, and user can keep entering text until an empty input is received.  
Instead of only display what user entered, this program will also try to translate the Morse code entered by user, display the translated message, and list the user inputs that were not translatable.

Valid but untranslatable inputs will be displayed as "?" in the translated message.

*Sample Output:*

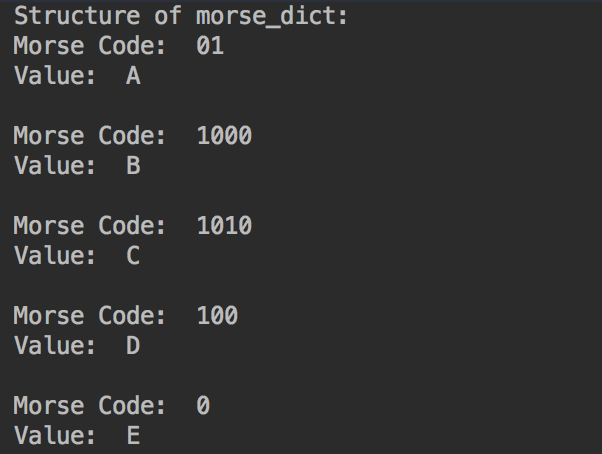


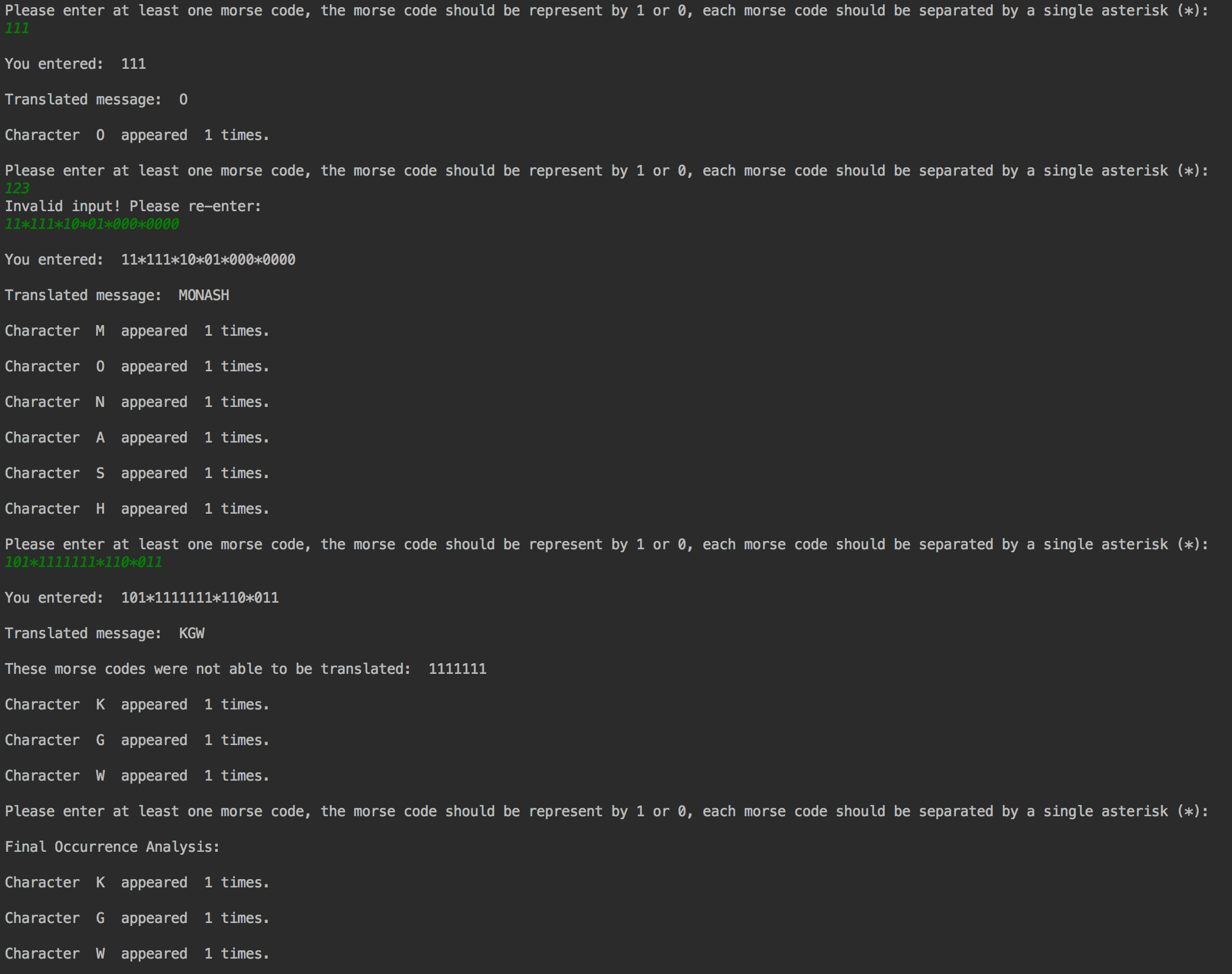
Task 4:

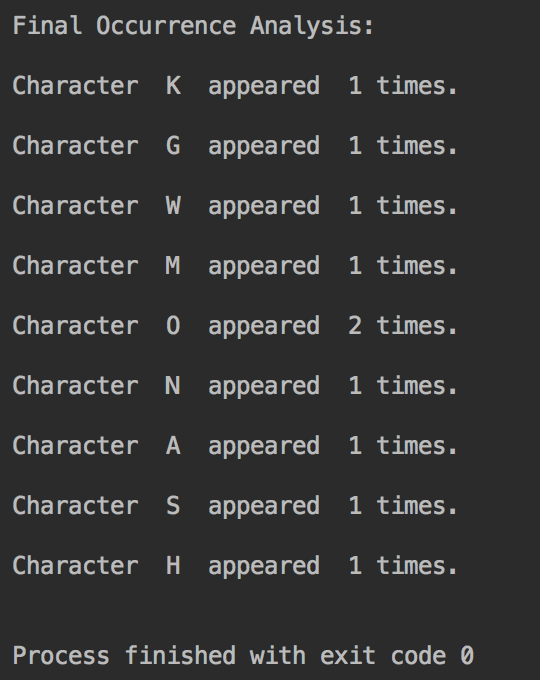
The structure of the Morse code dictionary will be displayed first, then it's pretty similar with the first steps of Task2.py and Task3.py, that user is asked to keep entering until an empty input is received.

A occurrence analysis of the current input will be generated after each time user pressed “enter”.  
When the sign of end of input is found (a.k.a. empty input), a report will be generated showing the occurrence of the characters that were entered by the user.

*Sample Outputs:*

**

**

**