Algorithms - HW 3

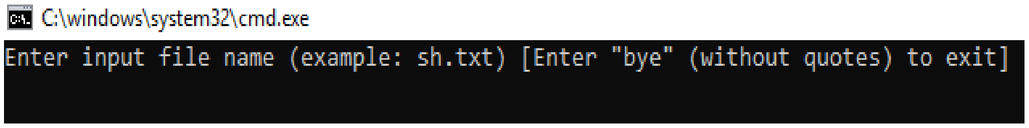
I. **Names of team members:**

- Shashank Shekhar (Unity Id: sshekha4)
- Rahul Ravindra(Unity Id: rravind)
- Akshay Podila (Unity Id: apodila)

II. -

III. Steps for the script files:

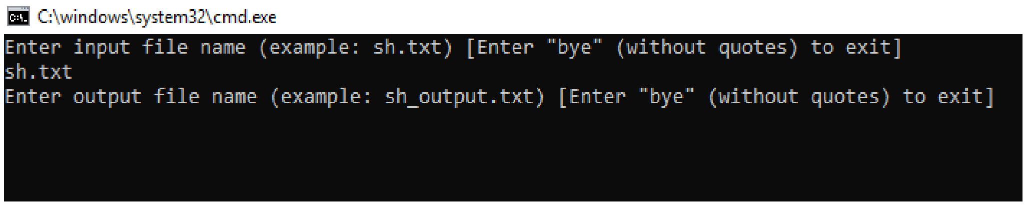
- **DIALOG Script file**



C:\windows\system32\cmd.exe

```
Enter input file name (example: sh.txt) [Enter "bye" (without quotes) to exit]
```

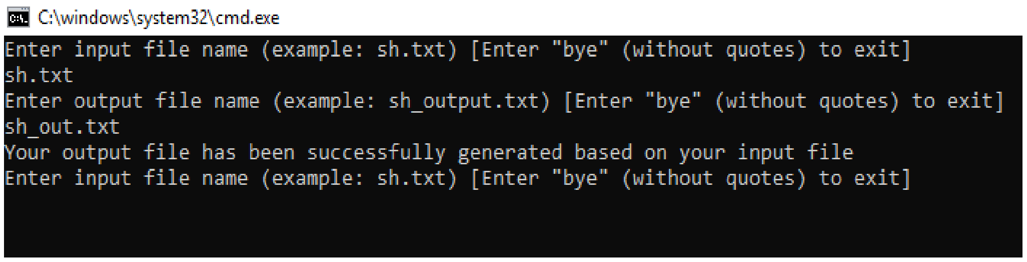
Here, enter the input file name or else enter "bye" (without quotes) to exit the script.



C:\windows\system32\cmd.exe

```
Enter input file name (example: sh.txt) [Enter "bye" (without quotes) to exit]
sh.txt
Enter output file name (example: sh_output.txt) [Enter "bye" (without quotes) to exit]
```

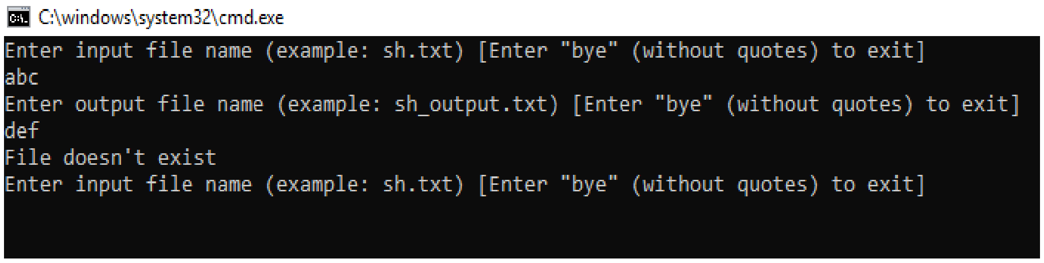
Here, enter the name for the output file that will contain the dialogs which would be generated for the given input file.



C:\windows\system32\cmd.exe

```
Enter input file name (example: sh.txt) [Enter "bye" (without quotes) to exit]
sh.txt
Enter output file name (example: sh_output.txt) [Enter "bye" (without quotes) to exit]
sh_out.txt
Your output file has been successfully generated based on your input file
Enter input file name (example: sh.txt) [Enter "bye" (without quotes) to exit]
```

Once the output files have been generated, an appropriate confirmation message will be shown to the user and the process will repeat until the user decides to type "bye" and exit the program.

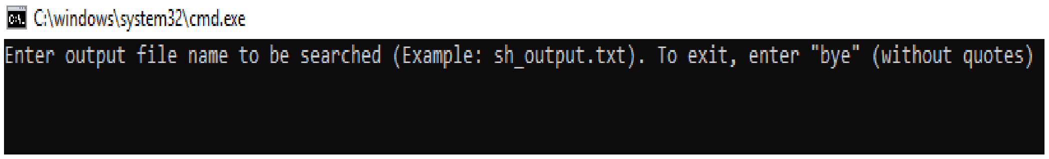


C:\windows\system32\cmd.exe

```
Enter input file name (example: sh.txt) [Enter "bye" (without quotes) to exit]
abc
Enter output file name (example: sh_output.txt) [Enter "bye" (without quotes) to exit]
def
File doesn't exist
Enter input file name (example: sh.txt) [Enter "bye" (without quotes) to exit]
```

If the file doesn't exist, appropriate message is shown to the user that the file doesn't exist.

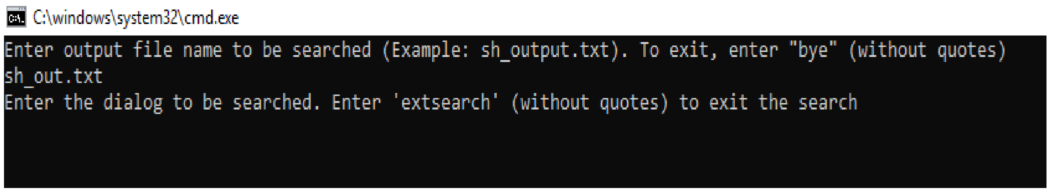
- **DIALOG SEARCH Script**



C:\windows\system32\cmd.exe

```
Enter output file name to be searched (Example: sh_output.txt). To exit, enter "bye" (without quotes)
```

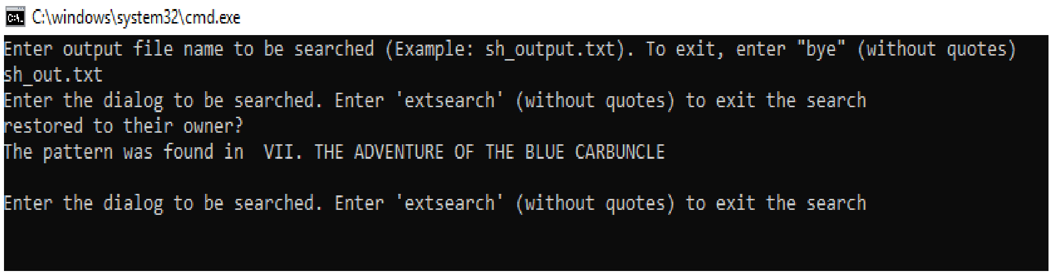
Enter the output file name (example: sh_output.txt) that needs to be searched. To exit at any time, enter "bye" without quotes.



C:\windows\system32\cmd.exe

```
Enter output file name to be searched (Example: sh_output.txt). To exit, enter "bye" (without quotes)
sh_out.txt
Enter the dialog to be searched. Enter 'extsearch' (without quotes) to exit the search
```

Enter the pattern to be searched (Example: restored to their owner?). Below output is returned:

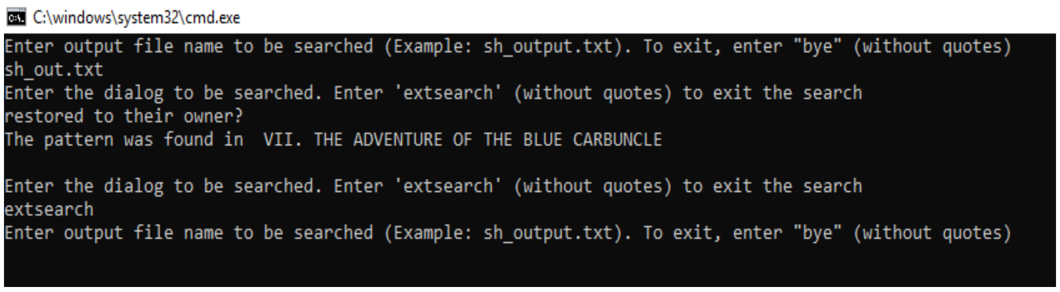


C:\windows\system32\cmd.exe

```
Enter output file name to be searched (Example: sh_output.txt). To exit, enter "bye" (without quotes)
sh_out.txt
Enter the dialog to be searched. Enter 'extsearch' (without quotes) to exit the search
restored to their owner?
The pattern was found in  VII. THE ADVENTURE OF THE BLUE CARBUNCLE

Enter the dialog to be searched. Enter 'extsearch' (without quotes) to exit the search
```

This shows that the above pattern was found in "Chapter Number. Chapter Title.
To return from the search space to the previous prompt at any point, enter "extsearch" (without quotes).



C:\windows\system32\cmd.exe

```
Enter output file name to be searched (Example: sh_output.txt). To exit, enter "bye" (without quotes)
sh_out.txt
Enter the dialog to be searched. Enter 'extsearch' (without quotes) to exit the search
restored to their owner?
The pattern was found in  VII. THE ADVENTURE OF THE BLUE CARBUNCLE

Enter the dialog to be searched. Enter 'extsearch' (without quotes) to exit the search
extsearch
Enter output file name to be searched (Example: sh_output.txt). To exit, enter "bye" (without quotes)
```

Then the process repeats where a user is asked for the output file in which the search needs to be performed. At any point to exit the script, enter "bye" (without quotes).

IV. Analysis Questions:

1. **NCSU Github URL: https://github.ncsu.edu/sshekha4/HW3**
2. Source code has is implemented by the team from scratch. We have used python version 3.7.3 for the implementation.
   In **dialog.py** code file:
   - We have used Regex Search function to search for the Chapter Titles and Number
   - We have written our custom code to look for the dialogs. Here, we have identifed smart double quotes, dumb double quotes and smart single quotes.

   In **dialog_search.py** code file:
   - We look for the pattern line by line in each of the dialogs. Whenever reading a line that contains a Chapter, we hold it in a variable that essentially points to the current chapter. Whenever a pattern is found, we output the current chapter.
3. **Part (a)**

   The **DIALOG** script is a batch file that runs on the given input files and any other input text files that follow a pattern similar to the given input files. Running the batch file give the appropriate onscreen instructions guiding the user. It takes the input text file name as well as the output text file name to be generated as inputs. If the input file doesn't exist, appropriate message is shown to the user and the user is asked to reenter the correct input file name. To exit the script, appropriate exit text (as displayed on the screen) needs to be typed on-screen.

   Part (b)

   The **DIALOG SEARCH** script is a batch file that takes the output file name in which the dialog search is to be made as input and then recursively asks the user for the dialogs to be searched until the user decides to exit. The script then asks the user to input any other output file that needs to be searched for dialogs. At any time when the user decides to exit, he can do so by following the appropriate on-screen instructions.
4. **Part -1 : All assumptions made about the input data**
   a) Every new paragraph starts with two new line characters prior to them.
   b) There is no space when we concatenate two sentences which merges the last word of the previous line and the first word of the present line, due to which we are adding a space every time we encounter a new line character.
   c) The user input file must have the same chapter format as either ***dracula.txt*** or ***sh.txt***:
   - **Dracula File Format**
     - Beginning of every chapter must be typed out as:
       CHAPTER ROMAN NUMERAL \n
       \n
       Chapter Name
     - After the index of chapters, the word "DRACULA\n" must be present before the contents of the book start.
   - **Sh File Format**

- ♦ Beginning of every chapter must be typed out as:
  Roman Numeral   Chapter Name

d) Dialog Assumptions:

- Dialogs start with double inverted commas (“) and end with double inverted commas (“). We are considering both smart and dumb double inverted commas.
- Text that is inside single inverted commas (‘) which are present after one double inverted comma but doesn't have a corresponding closing double inverted comma at the end of the paragraph is considered as a dialog.

Below is the approach used to handle each of the above-mentioned assumptions:

- In the **“sh.txt”** file, on utilizing regex string matching, the start of every chapter could be recognized.
- In the **“dracula.txt”** file, we recognized that every chapter started with the text “Chapter” followed by its chapter number, two end line characters and then the chapter name. This pattern was used to extract the Chapter Name and Chapter Title. However, because this same pattern was present in the book index as well its body, we utilized the keyword “DRACULA\n” which was present before the start of the book to record chapter names.

The approach used in **DIALOG** is simple character matching, while also keeping track of punctuation that have been parsed.

Based on the different cases of dialogs as mentioned above, we are using different flags to keep track of different kinds of punctuations. On ending the usage of a flag (one cycle), we write the data into an output file. The flags used are for:

- Checking if we have encountered a double inverted comma
- Checking if we have encountered a single inverted comma

The approach used in **DIALOG SEARCH** is taking each line in the output file of DIALOG and pattern matching it with the string we are attempting to find. On going through the output file, we store the current chapter name we are reading until the next chapter is found. The two end of line characters present before every chapter name in the output file are used to identify each chapter name / number.

Part – 2 : Additional parameters needed by the programs to run

DIALOG Program: It takes the input file name and the output file name as parameters from the script file. The script file takes these inputs from the user. The input file name refers to the file from which the dialogs are to be taken and the output file refers to the file name in which the dialogs will be saved for future use like searching for patterns inside the dialogs.

DIALOG SEARCH Program: It takes the output file name as parameter. The output file refers to the generated output file which holds all the dialogs for the corresponding input file. When the program runs, it takes user input for the pattern he wishes to search.

Both the above programs handle the scenario when an invalid file name is entered. They give appropriate message to the user and ask them to re-enter the correct file names again. The user after running the program for multiple inputs can safely terminate the program by following the onscreen instructions prompted by the batch file.

5. A user input file that has been never seen before will be acceptable to our program if the input file follows the same pattern as one of the given sample input files. The user input file must have the same chapter format as either *dracula.txt* or *sh.txt*, that is:
   - **Dracula File Format**
     - ♦ Beginning of every chapter must be typed out as:
       CHAPTER ROMAN NUMERAL \n
       \n
       Chapter Name
     - ♦ After the index of chapters, the word "DRACULA\n" must be present before the contents of the book start.
   - **SH File Format**
     - ♦ Beginning of every chapter must be typed out as:
       Roman Numeral Chapter Name

   Once the output file is generated, the user can use the second batch script to search for patterns in this output file.

   In order to determine the additional parameters used by the program, user simply needs to follow the on-screen instructions while running the batch file.

6. In order to verify that the sample input files generated the output files with the dialogs as expected, we followed the below approach:
   a) Basic manual inspection by scrolling through the output file to see if everything looks correct ie. There are no garbage values and that each of the dialogs have their start, and the end quotation marks.
   b) Then we took random dialogs from all the chapters of the input files and searched for the same in the output files. Random dialog searches from the input file into the dialog output file showed up as expected.
   c) We also checked for random dialogs containing apostrophes (like Shashank's chess game) and found that it appropriately escaped the apostrophe 's' character.

   Post this, we confirmed that the implementation works correctly.

7. For testing the DIALOG SEARCH functionality, we followed the following approach:
   a) We initially tested with a pattern containing only alphanumeric characters at the start to verify that the basic search functionality works fine.
   b) We then tested randomly for several different patterns that included special characters. For all special characters **(except hyphen)**, the pattern search works as expected. The pattern matches for some hyphen characters and doesn't match for others. This is because different hyphens have different encoding.
   c) We then searched for a few patterns containing alphanumeric and special characters that didn't exist in the output file. We couldn't find them.

Post this, we confirmed that the solution works correctly.

8. Worst case Asymptotic Time Complexity of the **DIALOG solution** as a function of m characters of total input is **O(m)**.
For the **DIALOG SEARCH solution**, the worst-case time complexity is **O(mn)** where m is the number of characters of input text and n is the number of characters in the input string.

9. **DIALOG DESIGN:**
If the same input file is to be given multiple times over a course of days, weeks or even years, we can create a hash of filenames with the file contents (such as MD5 hash). Whenever we encounter a file, we look for the filename in the hash table as the key. If no entry is found, it means that the file is encountered for the first time, If an entry is found, it means that the output for the given file was generated earlier. In that case, we check the hash value against the file name to see if the hash value calculated with the new file is the same as hash value already present. If the hash values match, we can confirm that the file contents are not modified. In this case, we know that the dialog output file we currently have for the file can be loaded into memory as it is from the disk without any further computation.

In order to identify the file, we use the filename as the key in the hash table. This will allow us a direct lookup in the hash table.

Benefit of the Approach:
1) We reduce the cost because hash table lookup based on the input filename takes constant time.
2) Once we know that the input file is not modified, we don't need to generate the output file again and write it to the disk as there are no changes. We just need to read the file from the disk.

Cost of the approach:
1) We need to maintain a hash table. It will require additional disk space to store the hash table.
2) We will be required to load the hash table each time into memory whenever the user runs the DIALOG batch script.
3) There will be computation cost involved in calculating the hash function value which is directly proportional to the size of the input text.

10. **DIALOG SEARCH DESIGN:**
If the collection of files that the program would have to search is known in advance, the files can be passed as input to the DIALOG script and their corresponding dialog output files can be prepared in advance. The user can identify the file he wanted to search by passing the output filename as an argument to the DIALOG script. This would generate the output file by the same name which is provided in the input to the DIALOG batch script. This file would later be used by the DIALOG SEARCH script as input when the user performs the pattern searches on the dialog text in this file.

When an additional file is received, it can be run on the dialog program using the DIALOG batch script which would generate the output file that will be added to the corpus.

Benefit of the approach:

1) We can create the dialog output files beforehand and keep them in corpus. This will save time when the user performs the search for the pattern on the dialog output files.

Cost of the approach:

1) If the input file is changed, then the corpus needs to be updated. For this, an additional mechanism must be kept in place that periodically checks for the any updates to the initial input files and correspondingly updates the dialog output files.