# **Assignment 5 – Testing the Word Term Frequencies Calculating Program**

**RunLog:**

* Environment: Windows 10 OS | Run on March 1st 2018 | 11:00PM || Python pyc file was decompiled and ran as py file
* **As of March 2nd 2018**
  + BlackBoxTesting
    - Total Bugs found : **5 bugs (Test ID: 2, 5, 6, 9, 11)**
    - Total Issues found : **1 issue (Test ID: 3)**
  + Stress Testing
    - Total Issues found :  **2 Issues (Test ID: )**
  + Combinational Testing
  + Internationalization Testing
* Please refer the table on the next page. Bugs are red coded

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test ID || Execution Order || Type** | **Summary** | **Input** | **Expected Output** | **Procedure** | **Test Type** | **Actual Output** |
| 1 || correct | Summary information using input file | mike.txt containing “mike was here” | 3 tokens 3 unique tokens | 1. Execute the command “python tf.pyc -s -i mike.txt” 2. Validate output is correct. | Functional Validation |  |
| 2 || Incorrect - **Bug** | Verbose Mode:  -Used other language, say Tamil language in input file | input.json file containing “தமிழ்” | “Can process only English language statements” since I assume this program does not support different languages | 1. Execute the command “python termFrequency.py -v” then “input.json”  2. Validate output is having info stating “Non-English language not supported” | Functional and Internationalization Validation | Vivek-Pc@kev MINGW64 /e/Marist/Semester2/SoftwareVerificationAndMaintenance/termFrequency\_2\_  $ python termFrequency.py -v  Please enter a file name: input.json  Debug mode!  Input file: input.json  Whole Input File Content: தமிழ்  ===================  Output file: ./termFreqOutput.txt  Term Frequencies:  "தம": {"count": "1", "tokenIds": ["0"]}  Traceback (most recent call last):  File "termFrequency.py", line 170, in <module>  output(i, wordKey[0])  File "termFrequency.py", line 41, in output  outputFile.write('"%s": {"count": "%s", "tokenIds": %s}' % (wordKey, len(sentenceDict[wordKey]), str(sentenceDict[wordKey]).replace("'", '"')))  File "C:\DEV\PYTHON3\lib\encodings\cp1252.py", line 19, in encode  return codecs.charmap\_encode(input,self.errors,encoding\_table)[0]  UnicodeEncodeError: 'charmap' codec can't encode characters in position 1-2: character maps to <undefined> |
| 3 || Feature Request || **Issue** | Verbose Mode:  -Used empty file | Input.json file contains nothing | Information stating like “No input found in the input.json file” | 1. Execute the command “python termFrequency.py -v” then “input.json”  2. Validate output is having info stating “File contains no input” | Functional Testing | Vivek-Pc@kev MINGW64 /e/Marist/Semester2/SoftwareVerificationAndMaintenance/termFrequency\_2\_  $ python termFrequency.py -v  Please enter a file name: input.json  Debug mode!  Input file: input.json  Whole Input File Content:  ===================  Output file: ./termFreqOutput.txt  Term Frequencies: |
| 4 || Correct Output | Verbose Mode:  -Used Code inside the file | Input.json file contains the python programing code | X tokens with X unique tokens | 1. Execute the command “python termFrequency.py -v” then “input.json”  2. Validate that output is correct | Functional Testing | Used factorial python program inside the file and the output came with no issue. This is because, the pgm converts all the inputs inside the input.json file as string and then process them ignoring the symbols |
| 5 || Incorrect output || **Bug** | Verbose Mode:  -Used combination of symbols to verify the consistency of ignorance. The “\_” is not being parsed well  - The “\_” should also be parsed like other symbols. | Input.json file contains “hello\_hello hello hello  hai-hai hai hai  abc\*\*\*abc abc abc  pop(pop) pop pop” | X tokens with X unique tokens should be obtained | 1. Execute the command “python termFrequency.py -v” then “input.json”  2. Validate that the output | Functional Testing | Vivek-Pc@kev MINGW64 /e/Marist/Semester2/SoftwareVerificationAndMaintenance/termFrequency\_2\_  $ python termFrequency.py -v  Please enter a file name: input.json  Debug mode!  Input file: input.json  Whole Input File Content: hello\_hello hello hello  hai-hai hai hai  abc\*\*\*abc abc abc  pop(pop) pop pop  ===================  Output file: ./termFreqOutput.txt  Term Frequencies:  "hai": {"count": "4", "tokenIds": ["3", "4", "5", "6"]}  "abc": {"count": "4", "tokenIds": ["7", "8", "9", "10"]}  "pop": {"count": "4", "tokenIds": ["11", "12", "13", "14"]}  "hello": {"count": "2", "tokenIds": ["1", "2"]}  "hello\_hello": {"count": "1", "tokenIds": ["0"]} |
| 6 || Incorrect output || **Bug** | Verbose Mode:  -Used symbols to verify the consistency of ignorance of it.  - The “\_\_\_\_\_\_\_\_” symbol used inside the input.json file  - Those are still symbols and they are not parsed and ignored. | Input.json file contains “------------------  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_ \_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_ -----\_\_\_\_-----\_\_\_\_\_  ~~~~  ~~~~  ``````````` ^^^^^^^^^^ ^^^ $$$” | No token output should be obtained | 1. Execute the command “python termFrequency.py -v” then “input.json”  2. Validate the output | Functional Testing | Vivek-Pc@kev MINGW64 /e/Marist/Semester2/SoftwareVerificationAndMaintenance/termFrequency\_2\_  $ python termFrequency.py -v  Please enter a file name: input.json  Debug mode!  Input file: input.json  Whole Input File Content: ------------------  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_ \_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_ -----\_\_\_\_-----\_\_\_\_\_  ~~~~  ~~~~  ``````````` ^^^^^^^^^^ ^^^ $$$  ===================  Output file: ./termFreqOutput.txt  Term Frequencies:  "\_\_\_\_\_": {"count": "2", "tokenIds": ["4", "8"]}  "\_\_\_\_\_\_": {"count": "2", "tokenIds": ["5", "6"]}  "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_": {"count": "1", "tokenIds": ["0"]}  "\_\_\_": {"count": "1", "tokenIds": ["1"]}  "\_\_": {"count": "1", "tokenIds": ["2"]}  "\_\_\_\_\_\_\_\_": {"count": "1", "tokenIds": ["3"]}  "\_\_\_\_": {"count": "1", "tokenIds": ["7"]} |
| 7 || correct output | Verbose Mode:  -Used symbol combinations again to verify the ignorance of it. | Input.json file contains “[][][][]  '''''' ''''' ;;;;; ;;;; :::: ;::::: .. .. ,, ,,, ??? /// ~~~````` !! !! @@ @@ ## $$ % ^ & \* ( ) - +=” | No tokens should be obtained | 1. Execute the command “python termFrequency.py -v” then “input.json”  2. Validate that the output should contain nothing | Functional Testing | Vivek-Pc@kev MINGW64 /e/Marist/Semester2/SoftwareVerificationAndMaintenance/termFrequency\_2\_  $ python termFrequency.py -v  Please enter a file name: input.json  Debug mode!  Input file: input.json  Whole Input File Content: [][][][]  '''''' ''''' ;;;;; ;;;; :::: ;::::: .. .. ,, ,,, ??? /// ~~~`````  !! !! @@ @@ ## $$ % ^ & \* ( ) \_ - +=  ===================  Output file: ./termFreqOutput.txt  Term Frequencies:  "\_": {"count": "1", "tokenIds": ["0"]} |
| 8 || correct output | Output file mode usage -o  -Used the input file that contains few bigger values | Input.txt contains 3,880,800 words with 50,409 lines | X tokens | 1. Execute the command “python termFrequency.py -v” then “input.json” and validate the output  2. Verify that the given output file is obtained with no issues | Functional Testing | Vivek-Pc@kev MINGW64 /e/Marist/Semester2/SoftwareVerificationAndMaintenance/termFrequency\_2\_  $ python termFrequency.py -o term.txt  Time Before inputting the fileName: 2018-03-02 12:41:22.169234  Please enter a file name: input.txt  Time Before completing the file write function: 2018-03-02 12:41:38.581536  Time After completing the file write function: 2018-03-02 12:41:38.586522 |
| 9 || Incorrect Output || **Bug** | Output file mode usage with other options -d -s  -Used the output file mode along with other options like -d or -s or both -d -s along with -o while executing the command, the given output file not created  - For example,  **Positive case:** refer 8 above  $ python termFrequency.py -o output.txt  -When above command used, the output.txt file is created with no issues.  **Negative case:**  $ python termFrequency.py -o -d -s output.txt  -Now, when this command executed, the output.txt file is not created. | Input.txt contains “Hello hai hello world hai hello” | X tokens | 1. Execute the command “python termFrequency.py -v” then “input.json” and validate the output | Functional Testing | Vivek-Pc@kev MINGW64 /e/Marist/Semester2/SoftwareVerificationAndMaintenance/termFrequency\_2\_  $ python termFrequency.py -o -d -s output.txt  Time Before inputting the fileName: 2018-03-02 12:52:58.017764  Please enter a file name: input.txt  Time Before completing the file write function: 2018-03-02 12:53:00.341114  Time After completing the file write function: 2018-03-02 12:53:00.346119  Summary:  Tokens: 9  Unique Tokens: 6  Max Fequency: ('hello', 3)  Min Fequency: 1 |
| 10 || Correct Code | Verbose mode with -a sorting feature testing  -Used the input.txt file containing the unsorted words | Input.txt contains “ | X tokens | 1. Execute the command “python termFrequency.py -v” then “input.json” and validate the output | Functional Tesging | Vivek-Pc@kev MINGW64 /e/Marist/Semester2/SoftwareVerificationAndMaintenance/termFrequency\_2\_  $ python termFrequency.py -v  Time Before inputting the fileName: 2018-03-02 12:57:01.796715  Please enter a file name: input.txt  Debug mode!  Input file: input.txt  Whole Input File Content: ﻿Hello Ada Joey Ada Hello  ===================  Output file: ./termFreqOutput.txt  Term Frequencies:  "hello": {"count": "2", "tokenIds": ["0", "4"]}  "ada": {"count": "2", "tokenIds": ["1", "3"]}  "joey": {"count": "1", "tokenIds": ["2"]}  Time Before completing the file write function: 2018-03-02 12:57:03.437732  Time After completing the file write function: 2018-03-02 12:57:03.442211  Vivek-Pc@kev MINGW64 /e/Marist/Semester2/SoftwareVerificationAndMaintenance/termFrequency\_2\_  $ python termFrequency.py -v -a  Time Before inputting the fileName: 2018-03-02 12:57:32.105676  Please enter a file name: input.txt  Debug mode!  Input file: input.txt  Whole Input File Content: ﻿Hello Ada Joey Ada Hello  ===================  Output file: ./termFreqOutput.txt  Term Frequencies:  "ada": {"count": "2", "tokenIds": ["1", "3"]}  "hello": {"count": "2", "tokenIds": ["0", "4"]}  "joey": {"count": "1", "tokenIds": ["2"]}  Time Before completing the file write function: 2018-03-02 12:57:36.949347  Time After completing the file write function: 2018-03-02 12:57:36.954351 |
| 11 || Incorrect Output || **Bug** | Input Mode  -Used the input mode and inputted the words  - The input is being received again and again  - There should be a way to stop accepting and process the input given. | Inputting in the command console like “Hello world hello world  still accepting the input  not valid...!!!” | X tokens | 1. Execute the command “python termFrequency.py -v” then “input.json” and validate the output | Functional Testing | Vivek-Pc@kev MINGW64 /e/Marist/Semester2/SoftwareVerificationAndMaintenance/termFrequency\_2\_  $ python termFrequency.py -n  Hello world hello world  still accepting the input  not valid...!!!  Traceback (most recent call last):  File "termFrequency.py", line 125, in <module>  content = sys.stdin.read()  KeyboardInterrupt |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **STRESS TESTING** | | | | | | |
| **Test ID** | **Execution Order || Type** | **Summary of what I did** | **Input** | **Expected Output** | **Procedure** | **Actual Output Obtained: (trimmed output a little to be view legibly)** |
| 1 | 1 || Correct Output | Normal file with normal inputs | Input.txt contains one line | X tokens | 1. Execute the command “python termFrequency.py -v” then “input.json” and validate the output | Vivek-Pc@kev MINGW64 /e/Marist/Semester2/SoftwareVerificationAndMaintenance/termFrequency\_2\_  $ python termFrequency.py -v  Please enter a file name: input.txt  Debug mode!  Input file: input.txt  Whole Input File Content: "hello world hai hello wolod world hello hello"  ===================  Output file: ./termFreqOutput.txt  Term Frequencies:  "hello": {"count": "4", "tokenIds": ["0", "3", "6", "7"]}  "world": {"count": "2", "tokenIds": ["1", "5"]}  "hai": {"count": "1", "tokenIds": ["2"]}  "wolod": {"count": "1", "tokenIds": ["4"]}  Time Before completing the file write function: 2018-03-02 12:06:06.435161  Time After completing the file write function: 2018-03-02 12:06:06.436153 |
| 2 | 2 || Correct Output | Small file testing | Input.txt contains 122 words | X tokens | 1. Execute the command “python termFrequency.py -v” then “input.json” and validate the output | Vivek-Pc@kev MINGW64 /e/Marist/Semester2/SoftwareVerificationAndMaintenance/termFrequency\_2\_  $ python termFrequency.py -v  Please enter a file name: input.txt  Debug mode!  Input file: input.txt  Whole Input File Content: Stress testing is a type of performance testing focused on determining an application’s robustness, availability, and reliability under extreme conditions. The goal of stress testing is to identify application issues that arise or become apparent only under extreme conditions. These conditions can include heavy loads, high concurrency, or limited computational resources. Proper stress testing is useful in finding synchronization and timing bugs, interlock problems, priority problems, and resource loss bugs. The idea is to stress a system to the breaking point in order to find bugs that will make that break potentially harmful. The system is not expected to process the overload without adequate resources, but to behave (e.g., fail) in an acceptable manner (e.g., not corrupting or losing data).  ===================  Output file: ./termFreqOutput.txt  ……  Time Before completing the file write function: 2018-03-02 12:18:31.055440  Time After completing the file write function: 2018-03-02 12:18:31.067437 |
| 3 | 3 || Correct Output | Big file testing | Input.txt contains 122,000 words | X tokens | 1. Execute the command “python termFrequency.py -v” then “input.json” and validate the output | Time Before completing the file write function: 2018-03-02 12:21:47.030644  Time After completing the file write function: 2018-03-02 12:21:47.035623 |
| 4 | 4 ||Correct Output | Bigger file testing | Input.txt contains 3,880,800 words with 50,409 lines | X tokens | 1. Execute the command “python termFrequency.py -v” then “input.json” and validate the output | Time Before inputting the fileName: 2018-03-02 12:38:41.372525  Time Before completing the file write function: 2018-03-02 12:39:23.800657  Time After completing the file write function: 2018-03-02 12:39:23.805682 |
| 5 | 5 || **Memory Error – I guess Issue is with my system** | Bigger file testing | Input.txt contains 17,791,500 words with 147,918 lines | X tokens | 1. Execute the command “python termFrequency.py -v” then “input.json” and validate the output | ===================  Traceback (most recent call last):  File "termFrequency.py", line 140, in <module>  tokens = re.findall('[\\w]+', content.lower().replace('\n', ' ').replace('\r', ' ').replace('. ', ' ').replace('; ', ' ').replace(', ', ' '))  MemoryError |
| 6 | 6 || **Memory Error – I guess Issue is with my system** | Bigger file testing | Input.txt contains 24,766,000 words with 203,204 lines | X tokens | 1. Execute the command “python termFrequency.py -v” then “input.json” and validate the output | Traceback (most recent call last):  File "termFrequency.py", line 140, in <module>  tokens = re.findall('[\\w]+', content.lower().replace('\n', ' ').replace('\r', ' ').replace('. ', ' ').replace('; ', ' ').replace(', ', ' '))  MemoryError |
|  |  |  |  |  |  |  |