Beijing Institute of Technology

Software Design Document

Name : Aminu Ibrahim Hashim (ID: 1820151002)

Christopher Jason Sjarif (ID: 1820151020)

Kembabazi Barbara Gamukama Yihong (ID:1820162088)

Project : Text Analyzer

Table of Contents

1.0 Introduction

Purpose

This document's purpose is to provide a fully detailed description of the Text Analyzer program as well as the program's architectural design. It provides information that can assist any necessary software development for this program.

Scope

This software design document focuses on a high-level framework of the Text Analyzer program's critical parts and functionality. In here, we focus on the program's major functions and software components as well as the interactions between the user and the program.

Overview

The Text Analyzer program is a user-based program, which is capable of reading, analyzing and modifying any given text file according to the user's choice. The user can select a file, count words and characters, find a word or phrase with in the file as well as replace it with another word, alter the casing of the text as well as combine two different text files into one.

2.0 General Description

Software Description

The Text Analyzer program is a C++ written program that is specially designed to assist a user in analyzing and formatting a chosen file text of personal choice. The program runs on a few functions which enable it to count characters and words, finding and replacement of words or phrases, changing of the casings o the words and combining two different text files. The Text Analyzer program is created for files of the text .doc format only which works best when run on the Mac OS, but is also modified to run on the Windows OS, UNIX, and Linux OS.

Software Functions

```
std::string getPassage();
std::string getToUpperAll();
std::string getToLowerAll();
std::vector<std::string> getWords();
std::vector<std::string> getSentences();
std::unordered_map<std::string, int> getWordOccurence();
void findWord(std::string word);
void findWordInsenstive(std::string word);
void replaceWord(std::string word, std::string replacement);
void capitalizeFirstLetter();
int noBlankSpaces(int flag = 0);
int noCapitalLetters(int flag = 0);
int noPunctuationMarks(int flag = 0);
int noSentences(int flag = 0);
int noWords(int flag = 0);
void printToUpperAll();
void printToLowerAll();
void printWordOccurences();
void printWords();
void printSentences();
void printPassage();
bool joinFile(const std::string fileName2);
void lowerCase(std::string& strToConvert);
bool isSentencePunc(char character);
bool capitalLetter(char character);
int findCaseInsensitive(std::string data, std::string toSearch, int pos = 0);
```

Limitations and Assumptions

- The program doesn't create a text file when run, however it requires the computer to have the already needed text file that is to be analyzed already saved in the system. This limits the Text Analyzer program from writing a new document.

- The program can be successfully run on Windows OS, Unix, Linux and Mac OS, whereby Mac OS is the most preferred system, since the program was originally created for the Mac OS. H9owever, this doesn't limit it's ability and functionality when run on other operating systems.

- The program can join two text files together; however, it is limited to the combination of at most two text files which already exist within the computer system. The combined files are saved as one new document automatically within the system hence the user cannot save the files personally. Despite the creation of a new file due to the combination, the Text Analyzer doesn't write independent files.

- The Text Analyzer File can only read and analyze txt files i.e. documents that are saved in the doc. Format otherwise it isn't able to analyze and modify documents that re saved in other formats that includes images and tables.

3.0 Functional Requirements

- Select file: The user of the program inputs a file name that exists within the computer that he/she wants to analyze. With the input name given the program checks within the computer system, opens and saves the text file within it for analyzing and modification.

- Count: The program counts the characters, words, sentences, word occurrences, number of capital letters, punctuation marks that are present in the text file. A list is generated and shown to the user via the user interface.

- Find: It possess the ability to locate a word that the user is searching. The user is requested to input a word or phrase that they want to find in the text and the result shows them the position and sentence number that is appears in. It occurs to two formats which are case sensitive and case insensitive whereby the latter shows all the number f times the word has appeared regardless of its casing format.

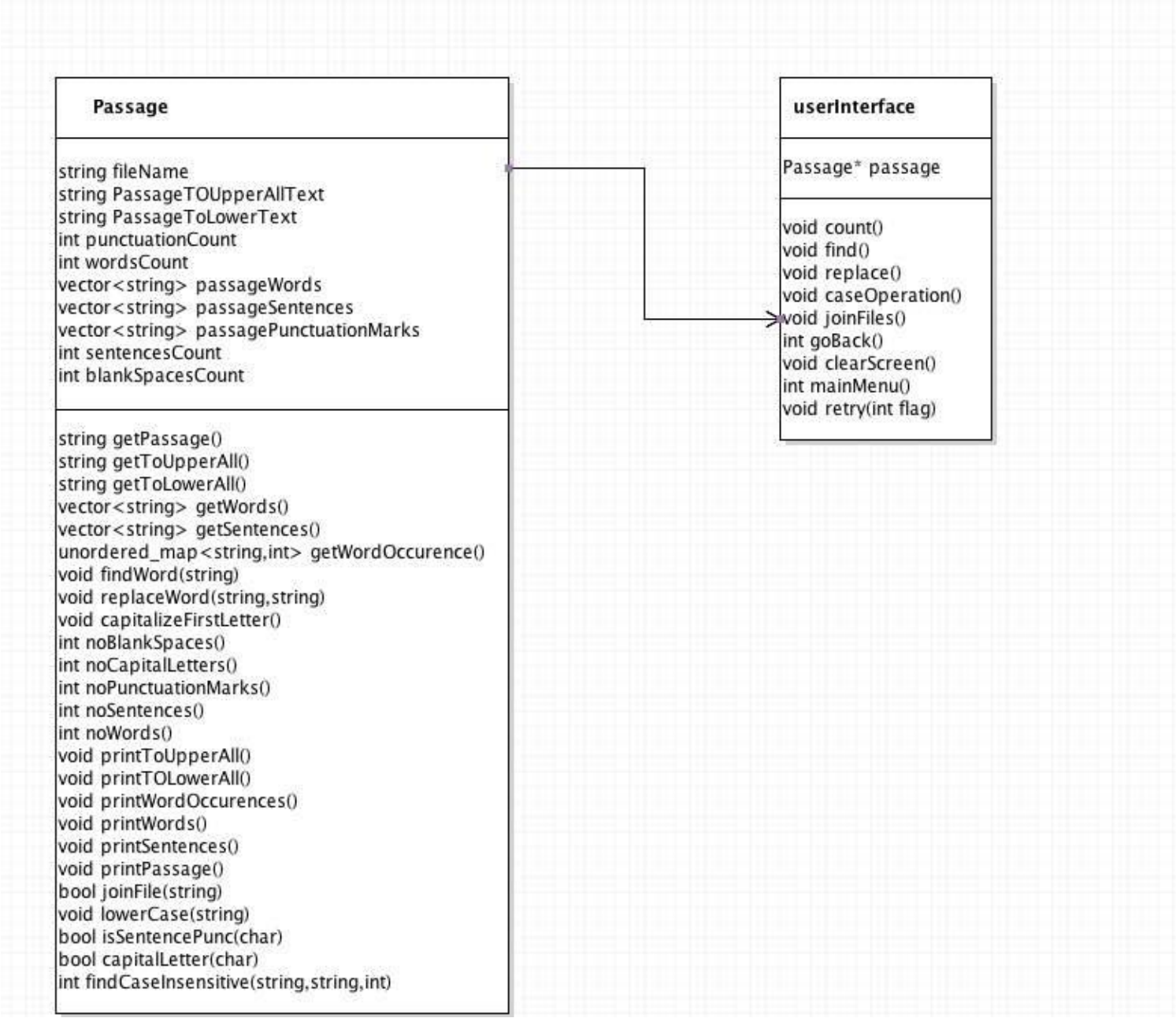
- Replace: This functionality occurs with relation to the find function. The user can replace a word or phrase according to

- Case Operations: This occurs in three forms namely; uppercase format where the all the characters are changed into upper casing, lower case format where all the characters are changed into lower case format, and sentence case; here, all the first letters of every sentence is converted into capital format.

- Join Files: With reference to the first opened and analyzed text file, the program can join it with another text file which is opened by the user after this option is chosen. However, this function can only work on two text files.

- Exit: This shuts down the program and clears all data that has been running within the program.

4.0 System Architecture



5.0 Sub-system Details

- Passage::Passage(std::string file); This is the constructor for the Passage Class, it sets the passage file as the file name that was passed to it. From which the program can identify that text file that is to be analyzed within the system.

- std::string Passage::getPassage() ; The Text Analyzer program gets the passage Text and stores it in the passage Text String, where it will be able to analyze and format the text file accordingly.

- void Passage::printPassage() ; Prints out a Passage heading and the text of the passage below for the user.

- std::vector<std::string> Passage::getWords() ; Gets all the words in the passage and stores them in a string vector sets the word occurrence vector with the number of occurrence of each word which is used in the count functionality later.

- int Passage::noWords(int flag); Prints out the number of words if the default flag is not changed .

- void Passage::printWordOccurences() ; The program prints headers and a list of words and word occurrences respectively on the user interface.

- std::unordered_map<std::string, int> Passage::getWordOccurence() ; returns an ordered map containing all the words in the text file and the number of times they each occur.

- std::vector<std::string> Passage::getSentences() ; The program gets all the sentences in the passage, sentences are a group of words or phrases that end with a period (.). The function stores the sentences in a string vector which is saved later for analysis.

- void Passage::printWords() ; Via the user interface the program prints a header ----------All Words---------- and a list of words in the text file beneath.

- void Passage::printSentences() ;Prints a header ----------All Sentences---------- and a list of sentences in the text file is presented below.

- std::string Passage::getToUpperAll() ; A function that changes the file contents into upper case format.

- void Passage::printToUpperAll() ; Prints the contents in the text file in Upper casing format after the above function has run successfully.

- std::string Passage::getToLowerAll(); Changes all the contents within the text file into lowercase format.

- void Passage::printToLowerAll() ; Prints a header ----------toLowerAll---------- and the text file in lower case beneath the header.

- int Passage::noSentences(int flag) ; Returns a count of the number of sentences in the passage if default flag is not changed, it prints the number of sentences that are contained in the text file.

- int Passage::noBlankSpaces(int flag) ; counts the number of blank spaces in the passage, assuming the default flag has not been altered with, it prints the number of blank spaces that are present in the text file.

-

- int Passage::noCapitalLetters(int flag) ; Counts the number of capital letters in the passage, assuming the default flag is not changed, it prints the number of capital letters within the text file

- bool Passage::capitalLetter(char character) ; Enables the program to check the characters of the text file and confirm whether they are capital, if it is returns true otherwise returns false.

- int Passage::noPunctuationMarks(int flag ; This counts the number of punctuation marks in the passage, if default flag is not changed, it prints the number of punctuation marks that are in the text file.

- void Passage::findWord(std::string word) ; Enables the program to identify a word in the passage and prints all the positions and sentences it was found in.

- int Passage::findCaseInsensitive(std::string data, std::string toSearch, int pos); Find Case Insensitive Sub String in a given substring

- void Passage::findWordInsenstive(std::string word); Locates a word in a passage regardless of its casing format, it counts the word as the same.

- void Passage::replaceWord(std::string word, std::string replacement) ; Enables the replacement of a "word" with a "replacement" in the text file.

- bool Passage::joinFile(const std::string fileName1) ; joins the contents of the originally opened text file with the content of another text file.

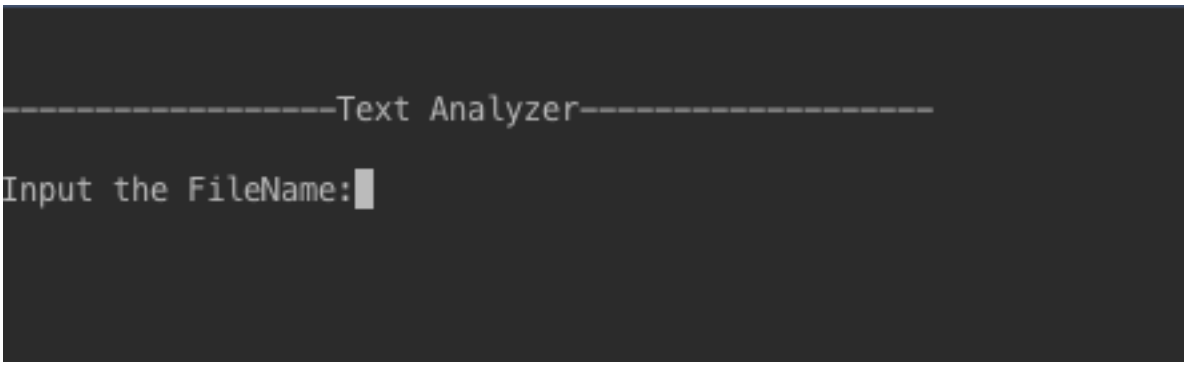
- void Passage::capitalizeFirstLetter() ; Changes the text file to sentence case. Here it affects only the first letter of every sentence that is within the text file to capital format.

- void Passage::lowerCase(std::string& strToConvert) ; Enable manipulation of the characters into lower case format.

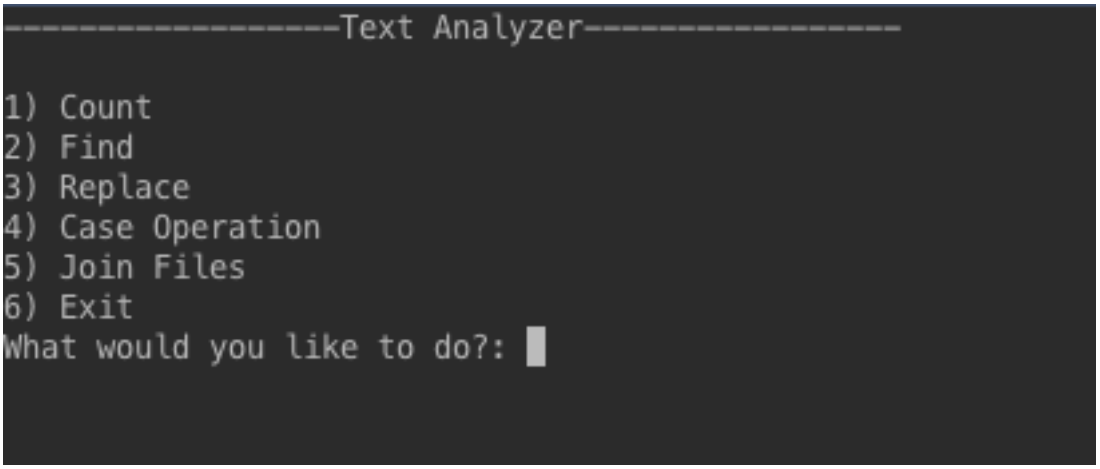
- bool Passage::isSentencePunc(char character) ; Checks to see if a character is a punctuation mark used to denote the end of a sentence. (! . ?), and counts them separately from the other characters like letters and words.

6.0 Software Interface

In The First Interface, the System asks them to input the name of the file



Then the system displays to the user a new prompt listing his available options



If the user selects the count option, the system counts and displays the details of the text file to the user

```
--------------------Text Analyzer--------------------
----------------------COUNT-------------------------
No of Sentences: 10

No of Words: 40

Number of blank spaces: 39

Number of Capital Letters: 10

Number of punctuation marks: 12



---------------------WORD OCCURRENCES-----------------

 WORD:OCCURRENCE

 10:1
 9:1
 8:1
 5:1
 6:1
 4:1
 7:1
 3:1
 2:1
 1:1
 is:10
 word:9
 word:1
 This:10

press 0 to go back to the Main Menu:
```

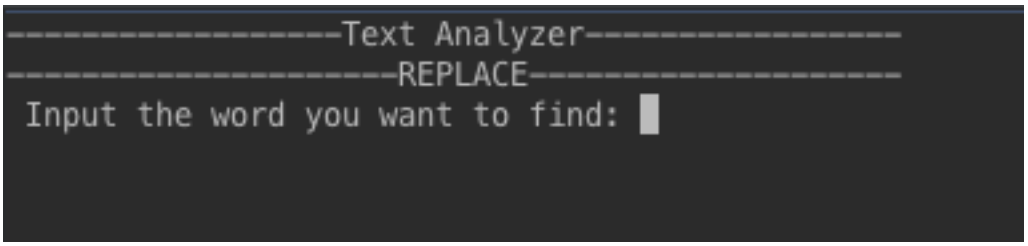
if the user wishes to find a word or phrase, he is given an option to choose case sensitive or insensitive search. Input apart from 1 or 2 makes the program to exit.

```
--------------------Text Analyzer--------------------
----------------------FIND--------------------------
Input 1 for case sensitive or 2 for case search Insensitive:
```

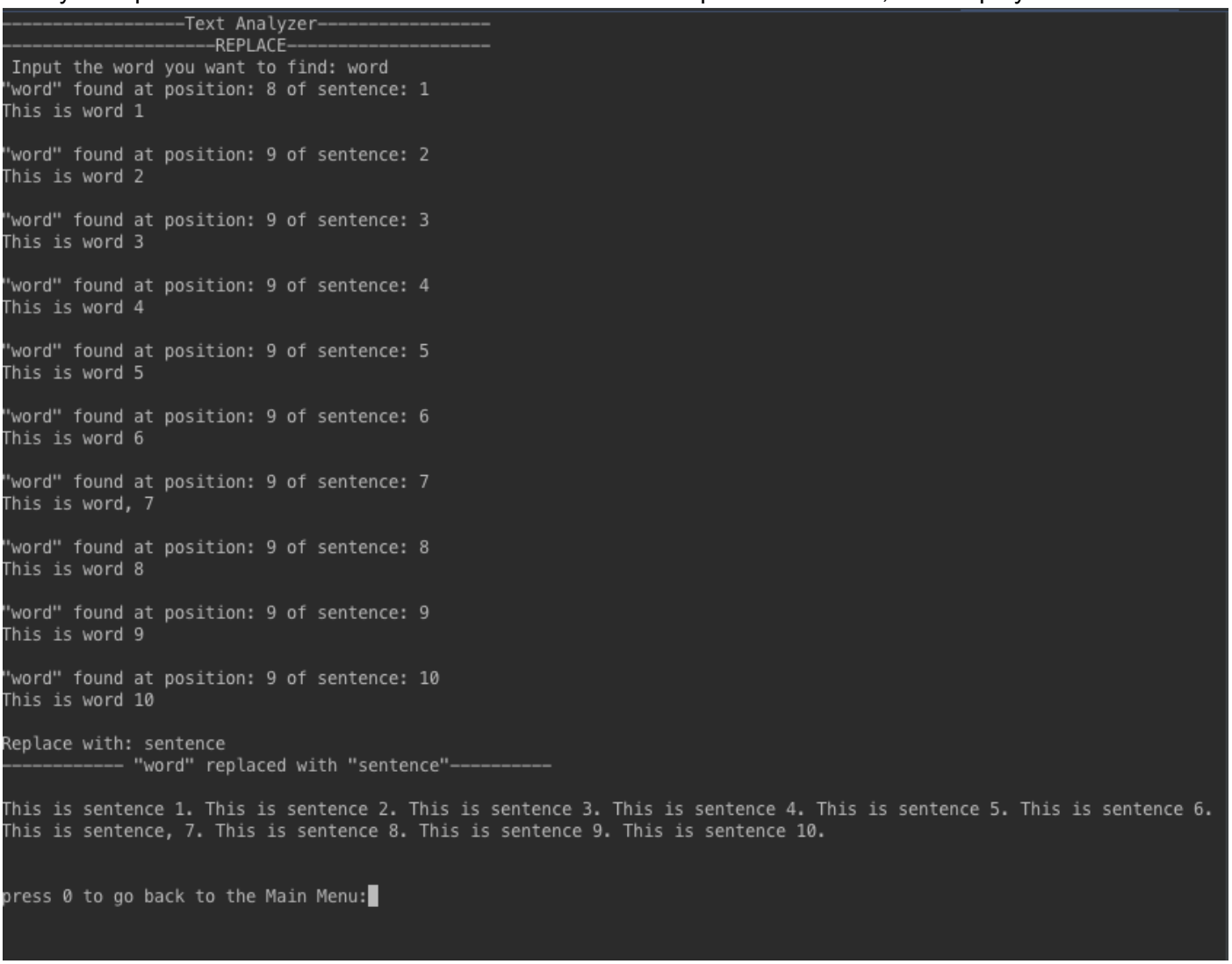
the position and sentence the word or phrase was found is printed

```
------------------Text Analyzer-----------------
--------------------FIND-----------------------
Input 1 for case sensitive or 2 for case search Insensitive: 2
Input the word you want to find: word
"word" found at position: 8 of sentence: 1
This is word 1

"word" found at position: 9 of sentence: 2
This is word 2

"word" found at position: 9 of sentence: 3
This is word 3

"word" found at position: 9 of sentence: 4
This is word 4

"word" found at position: 9 of sentence: 5
This is word 5

"word" found at position: 9 of sentence: 6
This is word 6

"word" found at position: 9 of sentence: 7
This is word, 7

"word" found at position: 9 of sentence: 8
This is word 8

"word" found at position: 9 of sentence: 9
This is word 9

"word" found at position: 9 of sentence: 10
This is word 10


Press 1 to return to previous menu
Press 2 to return to Main menu
Press 3 to return Exit
input:
```

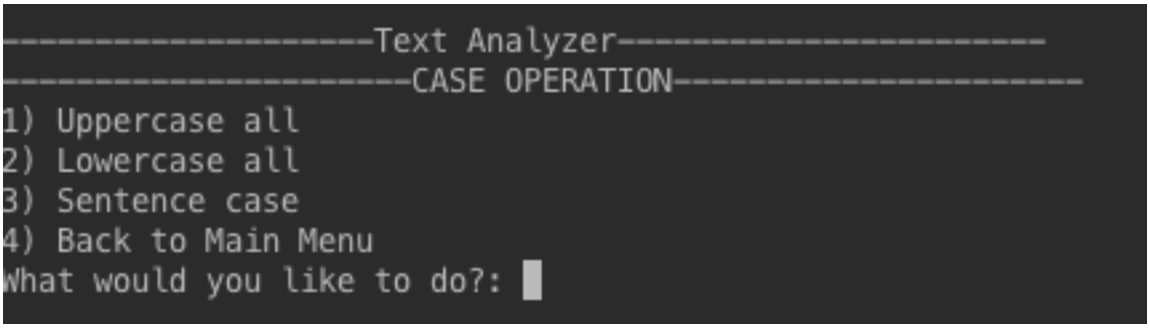
For the replace operation, user is asked for input



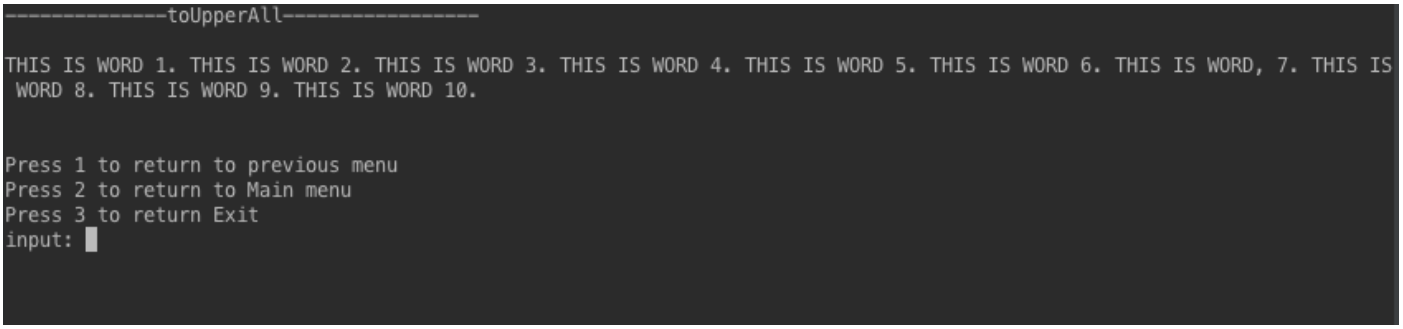
the system prints instances where its found and ask's for a replacement word, and displays to the user.



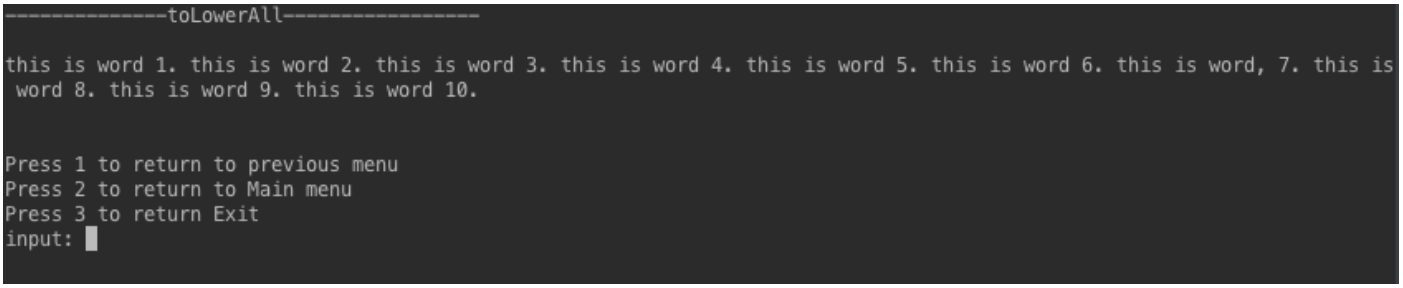
Displays the case operations to the user and asks for input .



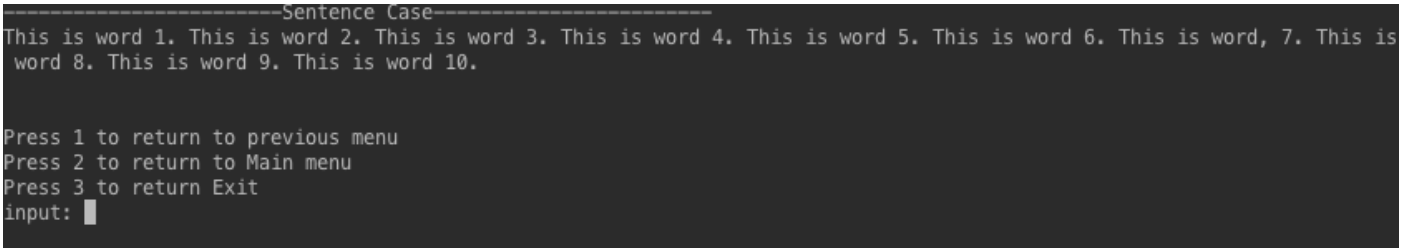
if user selects 1



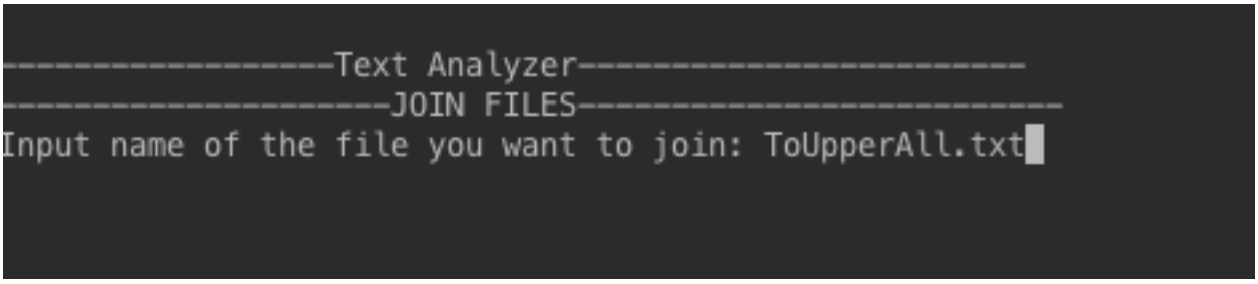
if user selects 2



if user selects 3



Joining two files together asks the user to input the name of the second file.



if it is joined successfully it notifies the user, if they weren't the user is also notified