COURSE NAME: Data Structures and Algorithms

STUDENT: Ertuğrul ŞENTÜRK

HOMEWORK SUBJECT: Word Changer Application

Function Definitions:

1- createSkipTable: Creates a Boyer-Moore table for given text

2- is_equal: Checks equality for two character and with case sensitive option

3- boyer_moore: Uses Boyer-Moore algorithm to find a text into bigger text and returns all instances

4- change_findings: Replaces all words in given indexes

5- read_file: Reads a file and stores into the char array.

6- write_file: Writes a char array into a file

Algorithm:

1- File name, word to be searched, word to be changed and case sensitive status information received from the user.

2- The file with the received file name was opened in read mode and the contents were saved in the char array named text.

3- boyer_moore function was called. In this function, a skip table was created for the text to be searched first. If case sensitive is selected while creating the skip table, the value of both lowercase and uppercase letters in the table has been changed.

4- Using the created skip table and boyer_moore algorithm, each match on the text were saved in an array called instances.

5- The size of this array was doubled and exponentially increased if the array was full.

6- The change_findings function was called to edit the text using the instances array. This function compared the length of the text to be replaced with the length of the searched text.

7- In order not to change the unprocessed characters of the array while making changes on the text array; If the text to be changed is shorter, the text string has been edited from beginning to end, if long, from end to beginning.

8- If the text to be changed is shorter, the elements of the array are shifted back by a variable t with an initial value of 0. This t variable has been increased in each loop by the difference between the text to be replaced and the searched text.

9- The instances array elements are checked with the index of the text array for matches. The text to be replaced was copied into the text string on given index. The index of the text string also shifted by the amount of text to be changed.

10- This process is repeated for each instance array by increasing the variable k in each match.

11- After the arrangements on the array were completed, array size is reduced by the last value of t by realloc.

12- If the text to be changed is longer, since the array will be evaluated from beginning to end, the variable t is assigned to size of instances array * size difference between the texts. t value decreased in each loop.

13- In that case the size of the array increased with re-alloc at the beginning.

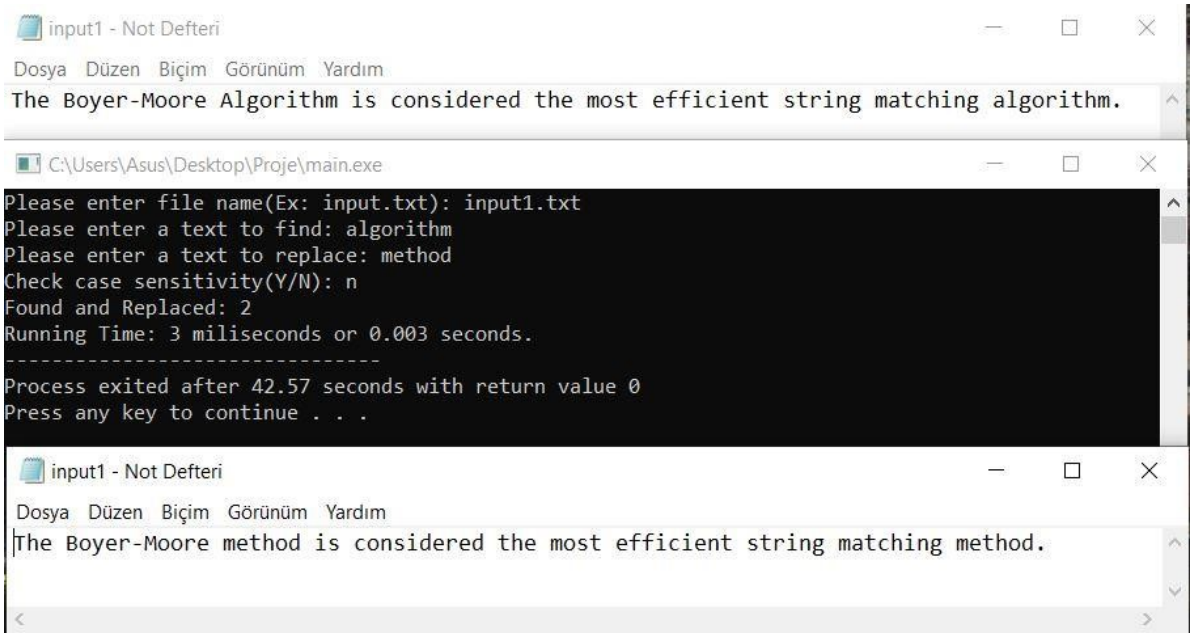
14- Also, text edited into end to beginning.

15- Edited array returned from the function and printed into the file.

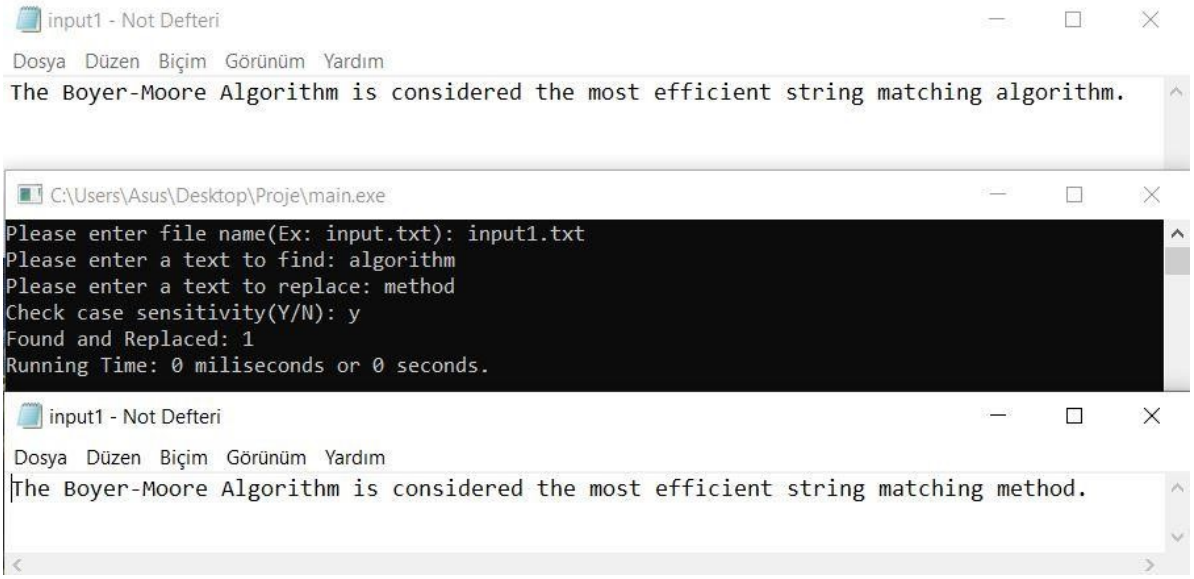
16- Processing time has calculated with clock_t class and also printed to user.

Screenshots:

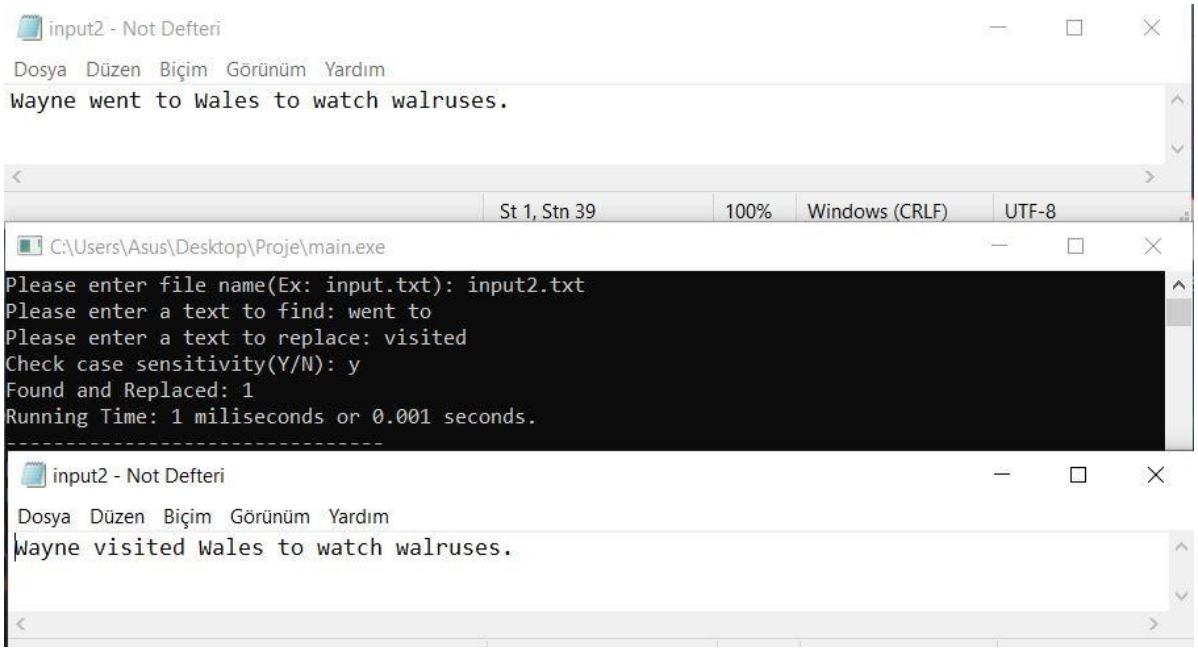
Input 1 without case sensitive:



Input 1 with case sensitive:



Input 2:



Time Evaluation:

We used basic “lorem impsum dolor sit amet” text for testing purposes.

Test-1 - “lorem”, “new text” replaced in various sized text.

```
Please enter file name(Ex: input.txt): yuz.txt
Please enter a text to find: lorem
Please enter a text to replace: newtext
Check case sensitivity(Y/N): y
Found and Replaced: 4
Running Time: 1 miliseconds or 0.001 seconds.
```

One hundred letters

```
Please enter file name(Ex: input.txt): bin.txt
Please enter a text to find: lorem
Please enter a text to replace: newtext
Check case sensitivity(Y/N): y
Found and Replaced: 39
Running Time: 10 miliseconds or 0.01 seconds.
```

One thousand letters

```
Please enter file name(Ex: input.txt): onbin.txt
Please enter a text to find: lorem
Please enter a text to replace: newtext
Check case sensitivity(Y/N): y
Found and Replaced: 385
Running Time: 13 miliseconds or 0.013 seconds.
```

Ten thousand letters

```
Please enter file name(Ex: input.txt): yuzbin.txt
Please enter a text to find: lorem
Please enter a text to replace: newtext
Check case sensitivity(Y/N): y
Found and Replaced: 3846
Running Time: 19 miliseconds or 0.019 seconds.
```

One hundred thousand letters

```
Please enter file name(Ex: input.txt): milyon.txt
Please enter a text to find: lorem
Please enter a text to replace: newtext
Check case sensitivity(Y/N): y
Found and Replaced: 38462
Running Time: 69 miliseconds or 0.069 seconds.
```

One million letters

```
Please enter file name(Ex: input.txt): onmilyon.txt
Please enter a text to find: lorem
Please enter a text to replace: newtext
Check case sensitivity(Y/N): y
Found and Replaced: 384616
Running Time: 565 miliseconds or 0.565 seconds.
```

Ten million letters

```
Please enter file name(Ex: input.txt): yuzmilyon.txt
Please enter a text to find: lorem
Please enter a text to replace: newtext
Check case sensitivity(Y/N): y
Found and Replaced: 3846154
Running Time: 5540 miliseconds or 5.54 seconds.
```

One hundred million letters

```
Please enter file name(Ex: input.txt): milyar.txt
Please enter a text to find: lorem
Please enter a text to replace: newtext
Check case sensitivity(Y/N): y
Found and Replaced: 38461539
Running Time: 55338 miliseconds or 55.338 seconds.
```

Billion letters

Test-2- Text size is same but word to search changed. Word to replace is same.

```
Please enter file name(Ex: input.txt): yuzmilyon.txt
Please enter a text to find: d
Please enter a text to replace: 1234567890
Check case sensitivity(Y/N): y
Found and Replaced: 3846154
Running Time: 7512 miliseconds or 7.512 seconds.
```

1 letter long

```
Please enter file name(Ex: input.txt): yuzmilyon.txt
Please enter a text to find: lorem ipsu
Please enter a text to replace: 1234567890
Check case sensitivity(Y/N): y
Found and Replaced: 3846154
Running Time: 6192 miliseconds or 6.192 seconds.
```

10 letters long

```
Please enter file name(Ex: input.txt): yuzmilyon.txt
Please enter a text to find: lorem ipsum dolor si
Please enter a text to replace: 1234567890
Check case sensitivity(Y/N): y
Found and Replaced: 3846154
Running Time: 5104 miliseconds or 5.104 seconds.
--------------------------------
```

20 letters long

Test-3 Text size is same but word to replace changed. Word to search is same.

```
Please enter file name(Ex: input.txt): yuzmilyon.txt
Please enter a text to find: lorem
Please enter a text to replace: 1
Check case sensitivity(Y/N): y
Found and Replaced: 3846154
Running Time: 5631 miliseconds or 5.631 seconds.
```

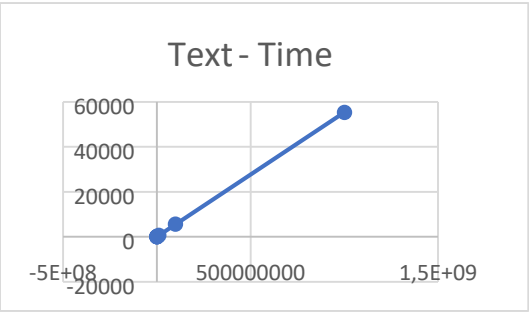
1 character

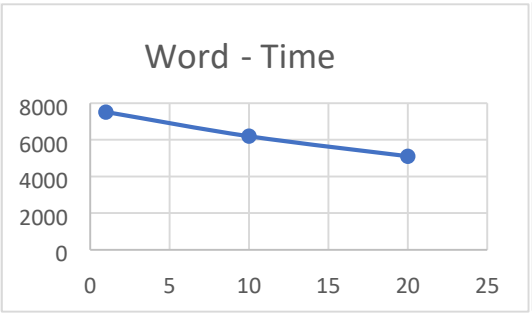
```
Please enter file name(Ex: input.txt): yuzmilyon.txt
Please enter a text to find: lorem
Please enter a text to replace: 1234567890
Check case sensitivity(Y/N): y
Found and Replaced: 3846154
Running Time: 6614 miliseconds or 6.614 seconds.
```

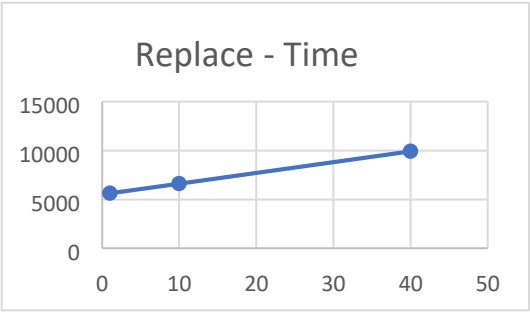
10 character

```
Please enter file name(Ex: input.txt): yuzmilyon.txt
Please enter a text to find: lorem
Please enter a text to replace: 1234567890123456789012345678901234567890
Check case sensitivity(Y/N): y
Found and Replaced: 3846154
Running Time: 9925 miliseconds or 9.925 seconds.
--------------------------------
```

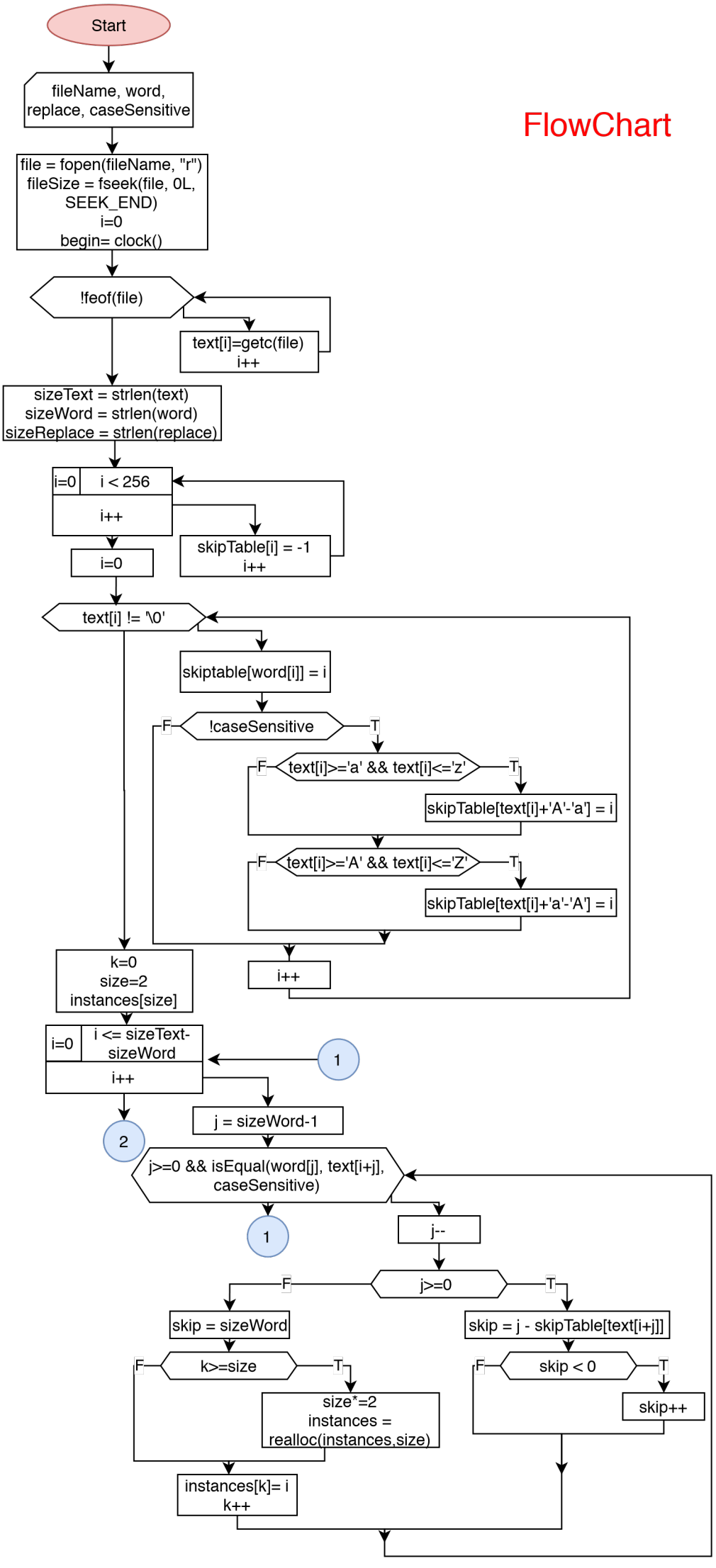
40 character

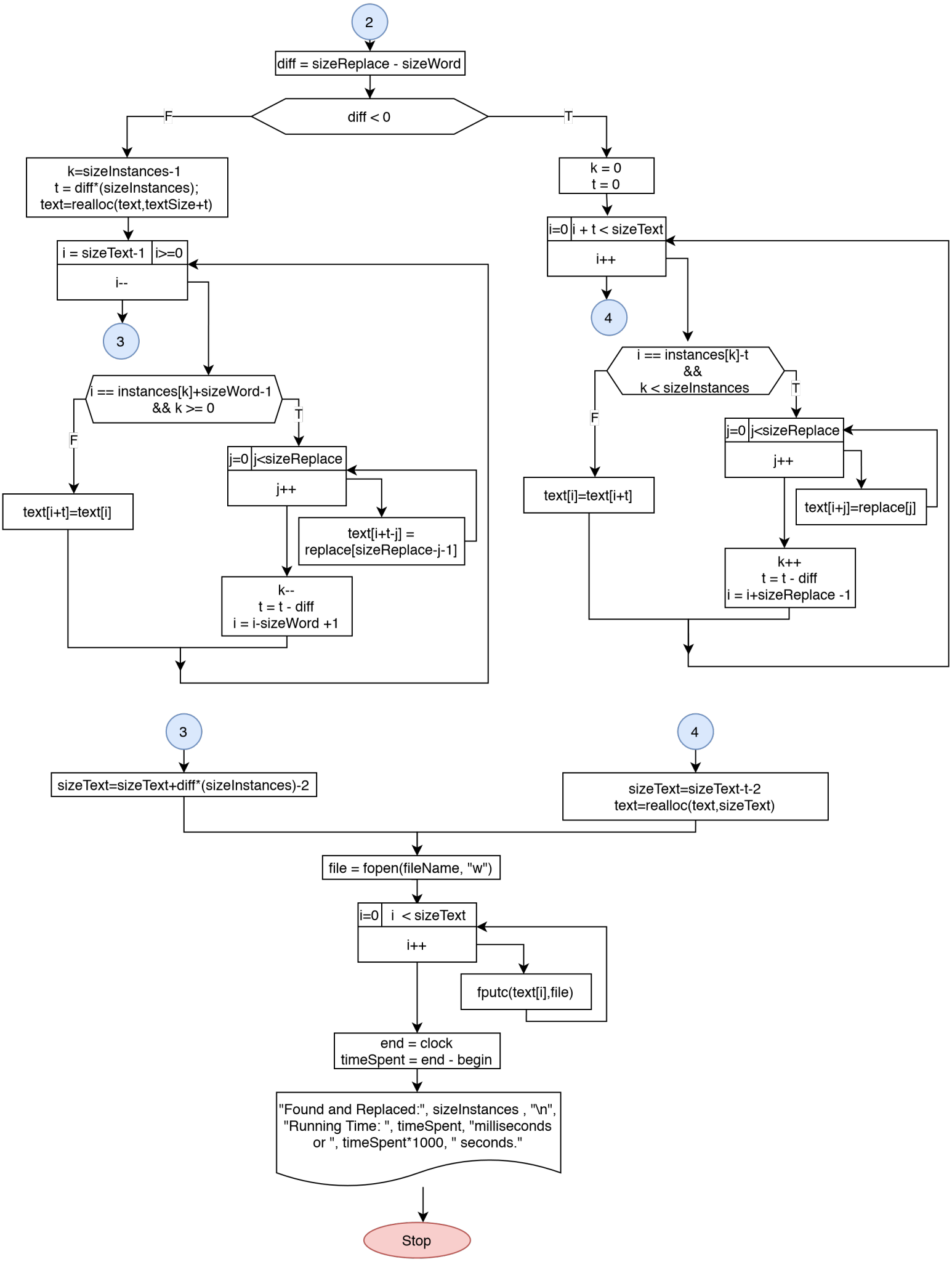






FlowChart


Start
fileName, word, replace, caseSensitive
file = fopen(fileName, "r")
fileSize = fseek(file, 0L, SEEK_END)
i=0
begin= clock()
!feof(file)
text[i]=getc(file)
i++
sizeText = strlen(text)
sizeWord = strlen(word)
sizeReplace = strlen(replace)
i=0
i < 256
i++
skipTable[i] = -1
i++
i=0
text[i] != '\0'
skiptable[word[i]] = i
F
!caseSensitive
T
F
text[i]>='a' && text[i]<='z'
T
skipTable[text[i]+'A'-'a'] = i
F
text[i]>='A' && text[i]<='Z'
T
skipTable[text[i]+'a'-'A'] = i
i++
k=0
size=2
instances[size]
i=0
i <= sizeText-sizeWord
i++
1
2
j = sizeWord-1
j>=0 && isEqual(word[j], text[i+j], caseSensitive)
1
j--
F
j>=0
T
skip = sizeWord
skip = j - skipTable[text[i+j]]
F
k>=size
T
size*=2
instances = realloc(instances,size)
F
skip < 0
T
skip++
instances[k]= i
k++


2
diff = sizeReplace - sizeWord
diff < 0
F
T
k=sizeInstances-1
t = diff*(sizeInstances);
text=realloc(text,textSize+t)
i = sizeText-1
i>=0
i--
3
i == instances[k]+sizeWord-1
&& k >= 0
F
T
text[i+t]=text[i]
j=0
j<sizeReplace
j++
text[i+t-j] =
replace[sizeReplace-j-1]
k--
t = t - diff
i = i-sizeWord +1
k = 0
t = 0
i=0
i + t < sizeText
i++
4
i == instances[k]-t
&&
k < sizeInstances
F
T
text[i]=text[i+t]
j=0
j<sizeReplace
j++
text[i+j]=replace[j]
k++
t = t - diff
i = i+sizeReplace -1
3
sizeText=sizeText+diff*(sizeInstances)-2
4
sizeText=sizeText-t-2
text=realloc(text,sizeText)
file = fopen(fileName, "w")
i=0
i < sizeText
i++
fputc(text[i],file)
end = clock
timeSpent = end - begin
"Found and Replaced:", sizeInstances , "\n",
"Running Time: ", timeSpent, "milliseconds
or ", timeSpent*1000, " seconds."
Stop