

-Project 1-

ENCS 313

Student : Raji Abdallah , section 1

Student : Anas Yassin , section 4

Raji : 1180195

Anas : 1180837

Task 1 : After the user enter the name of main file that has the original text and the summary ratio , the file contents will go through a number of edits.

```
1 #!/bin/bash
2 echo "Please enter the file name " #the main file that has the sentences
3 read fileName
4 echo "Please Enter the summary ratio "
5 read summaryRatio|
```

- **TASK 2:** First , the sentences will be tokenized based on this marks (. ? !) , we used it to split based on it , when the program reaches any of that it will make a new line , to make every sentence in a single line .Then the program will convert every small letter to capital one , the text will be redirected to the temp file that contains every sentence in a single line

```
7 if [ ! -e $fileName ] #check if the file exist
8 then
9     echo "Error ,,the file isn't exist "
10    exit 1
11 elif [ -e "$fileName" ]
12 then
13     cat $fileName | tr '[:!?:]' '\n' > temp #tekonization the sentences based on [ . ? ! ], it will make a new line .
14     cat temp | tr '[A-Z]' '[a-z]' > $fileName #convert all charactars to small letters.
15 fi
```

- Second step : The program will delete and single word of these (stop words) from each line [I, a, an, as, at, the, by, in, for, of, on, that] using sed command. Then it will remove the spaces that was in [. ! ?] places.

```
19 cat $fileName | sed "s/\<i\>//g" | sed "s/\<a\>//g" | sed "s/\<an\>//g" | sed "s/\<as\>//g" > temp #
20
21 cat temp | sed "s/\<at\>//g" | sed "s/\<the\>//g" | sed "s/\<by\>//g" | sed "s/\<in\>//g" > $fileNar
22
23 cat $fileName | sed "s/\<for\>//g" | sed "s/\<on\>//g" | sed "s/\<that\>//g" > temp
24
25 cat temp | tr -s ' ' ' ' > $fileName #to delete the repeated spaces.
```

- Then the program will delete every duplicated word from each sentence to make sure that the intersection will not increase by 1 for same word 2 times . The temp file contents should be redirected to the result file that will compare between the sentences by the required way .

```
27 awk '{ while (++i<=NF) printf(!a[$i]++) ? $i FS : "" ;i=split("",a); print ""}' $fileName > result # this code to delete the duplicated words
```

TASK 3 & 4

- It's very important to remove the empty lines that came from splitting the text, because it will make some problem while comparing between sentences, the program will search (using grep command) for the lines that doesn't contain anything and copy the other lines to the file temp. Then it will copy the temp contents to the result file that we will work with it.

```
29 grep "[^\\^$]" result > temp # to delete the empty lines .  
30 cat temp > result  
..
```

- We declared the variables we will use in the comparing loops here and initialized all of them by zero. The array indexes should be initialized by zero to not have a syntax error, because if we don't do that the array contents will be spaces.

```

36 interSection=0
37 union=0
38 numOfLines=$(cat result | wc -l)
39 similarity=0 #to calculate the the similarity of the word with others
40 z=0
41 declare -a centrality=()
42 while [ $z -lt $numOfLines ] #assign all indexes to zero
43 do
44     centrality[$z]=0
45     z=$((z+1))
46 done

```

- the first while loop will reach line by line n number of lines, then the second loop will reading line by line until (n-1) line.
- third loop will reach all the words in line i so we need fourth loop to reach all the words in the next line to compare the word in the previous line.
- and we find the similarity of each line compared, and the centrality.

```

50 while [ $i -le $numOfLines ] #to reach the all lines
51 do
52     k=0
53     j=1
54     k=$((i+1))
55     while [ $j -le $((($numOfLines - $i)) ) ] # to reach the next lines after line i
56     do
57         union=0
58         interSection=0
59         similarity=0
60         str1=$(cat result | sed -n ${i}p)          # To get the i line from the result file
61         for word1 in $str1                          # loop will pass through each word in the string to compare it with words in the other one
62         do
63             str2=$(cat result | sed -n ${k}p) #To get every word in line k((line i+1)
64             numOfWords1=$(echo $str1 | wc -w) #num of words to calculate the union
65             numOfWords2=$(echo $str2 | wc -w)
66             union=$((($numOfWords1 + $numOfWords2)) #union between the 2 lines
67             for word2 in $str2 #to compare every word in the line k
68             do
69                 if [ "$word1" = "$word2" ] #check if the words the same.
70                 then
71                     interSection=$((($interSection + 1)) #increment the intersection value by 1
72                 fi
73             done
74         done
75         similarity=$(echo "scale=3;$interSection/$union" | bc -l ) #We used bc command to convert the integer number to float number.
76         centrality[$i-1]=$(echo "scale=3;${centrality[$((i-1))]}+$similarity" | bc -l) # adding similarity of the line compare to the other lines.
77         centrality[$k-1]=$(echo "scale=3;${centrality[$((k-1))]}+$similarity" | bc -l) # to make sure that the compare will be between line and lines who's after it.
78         k=$((k+1)) #increment k that reach the other lines.
79         j=$((j+1)) #increment j that make sure the program will not go back to the last line.
80     done
81     i=$((i+1)) #increment i to reach the all lines in the main file .
82 done

```

TASK 5

- Here we declared an array to assign every line to a single index , we will use this array to sort the lines while sorting the centrality array , to keep every centrality to its line.

```
86 declare -a lines=() # here i declared an array to assign every line in an index , i will use it in sorting based on centrality .
87 i=0
88
89 while [ $i -lt $numOfLines ] # to initialize the all indexes contents to 0.
90 do
91     lines[$i]=0
92     i=$((i+1))
93 done
94
95
96 for ((j=0 ; j < $numOfLines;j++)) #to assign all lines in array , to reach every line separately .
97 do
98     k=$((j+1)) # to start from line j+1
99     lines[$j]=$ (cat result | sed -n ${k}p)
100 done
```

- Then we sorted the arrays (the array that has the lines and the array that has the centrality),we got the array's length from number of elements in the centrality array to use it in sorting . We compared every element with the other elements.

```

104 arrayLength=${#centrality[@]}
105 while [ $i -lt $arrayLength ] #here we sorted the centrality array with lines array ..we need nested loops
106 do
107     temp=0 #this variable to switch between the array index contents
108     j=$((i+1))
109     while [ $j -lt $arrayLength ]
110     do
111         if (( $(echo " ${centrality[$j]} > ${centrality[$i]}" | bc -l) )) #compare between the i index with next indexes
112         then
113             #here is the switching between centrality values and lines.
114
115             temp=${centrality[$i]}
116             centrality[$i]=${centrality[$j]}
117             centrality[$j]=$temp
118             str=${lines[$i]}
119             lines[i]=${lines[j]}
120             lines[j]=$str
121         fi
122         j=$((j+1)) #increment j value by 1
123     done
124     i=$((i+1)) #increment i value by 1
125 done

```

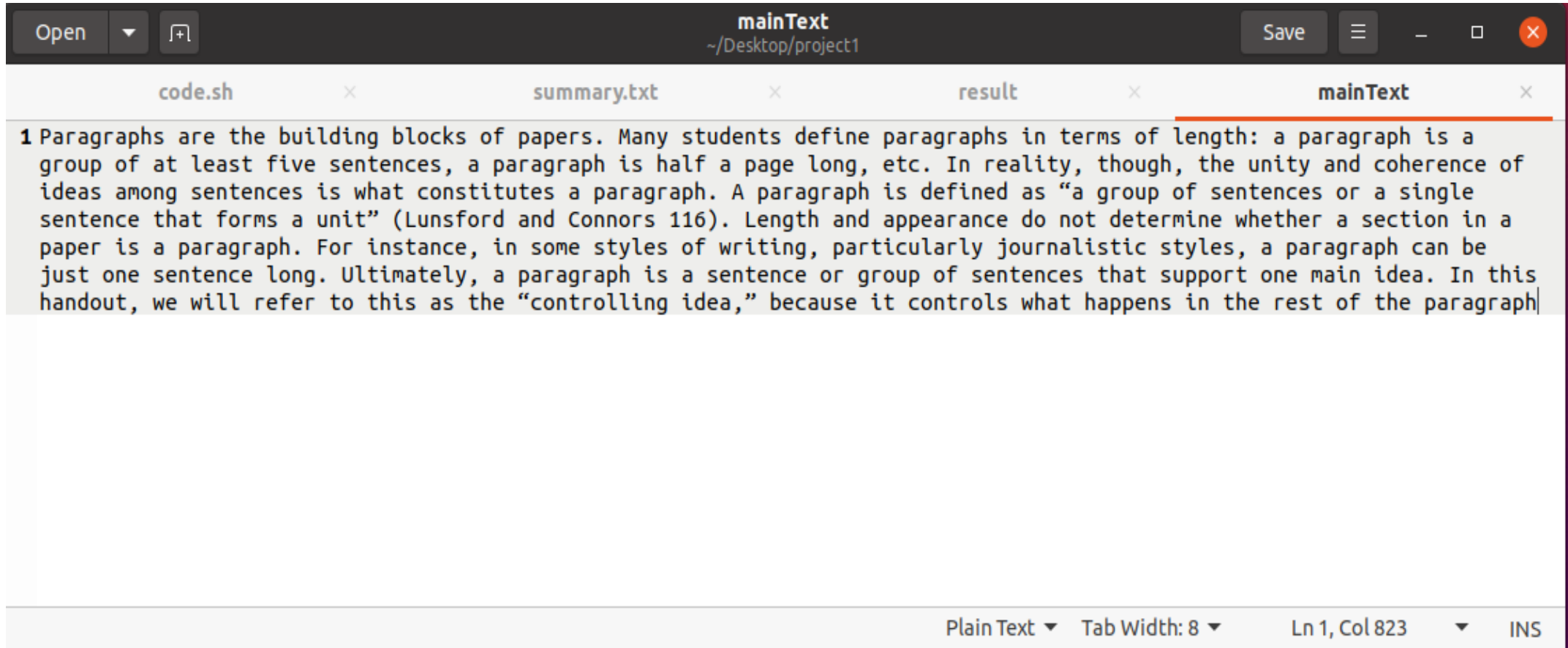
- We declared the variable ratioOfLines by multiply number of lines in the main text by summary ration that we took it from the user. Then we checked if the summary.txt file is empty or not to make sure that we won't append to any other things , the while loop will check every time if the l value more than the rationOfLines , if it's not , the program will add the line from the sorted array to the summary.txt file .

```

128 ratioOfLines=$(echo "scale=2; $summaryRatio * $numOfLines" | bc -l ) # this is the number of lines will be shown in the final file (summary.txt)
129 : '
130     if the ratio of lines result was a floating point number , it will be approximate .
131 '
132 i=0
133 echo -n "" > summary.txt
134 while true
135 do
136     if (( $(echo " $i < $ratioOfLines" | bc -l) ))
137     then
138         echo ${lines[$i+1]} >> summary.txt # appending the final lines to the summary.txt file. after make sure that the file is empty.
139         i=$((i+1))
140     else
141         break
142     fi
143 done

```


- The main file before editing :



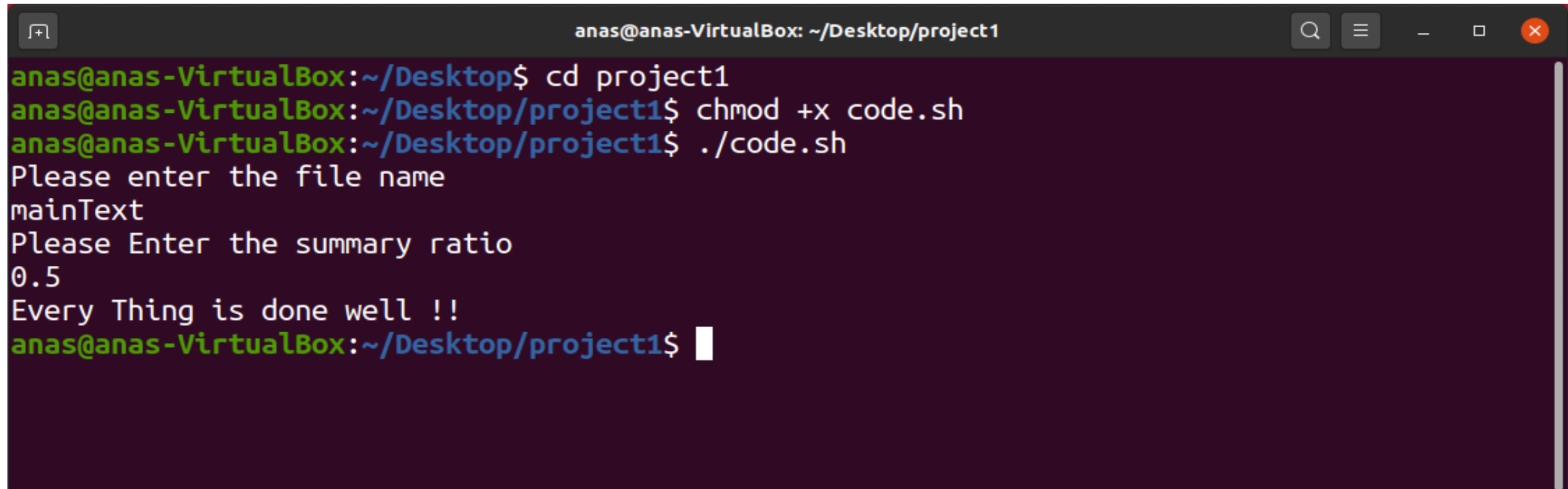
The screenshot shows a code editor window with a dark theme. The title bar at the top reads 'mainText' and the path is '~/.Desktop/project1'. There are buttons for 'Open', 'Save', and window controls. Below the title bar, there are four tabs: 'code.sh', 'summary.txt', 'result', and 'mainText'. The 'mainText' tab is active and highlighted with a red underline. The text in the 'mainText' tab is as follows:

```
1 Paragraphs are the building blocks of papers. Many students define paragraphs in terms of length: a paragraph is a group of at least five sentences, a paragraph is half a page long, etc. In reality, though, the unity and coherence of ideas among sentences is what constitutes a paragraph. A paragraph is defined as "a group of sentences or a single sentence that forms a unit" (Lunsford and Connors 116). Length and appearance do not determine whether a section in a paper is a paragraph. For instance, in some styles of writing, particularly journalistic styles, a paragraph can be just one sentence long. Ultimately, a paragraph is a sentence or group of sentences that support one main idea. In this handout, we will refer to this as the "controlling idea," because it controls what happens in the rest of the paragraph|
```

At the bottom of the editor, there is a status bar showing 'Plain Text', 'Tab Width: 8', 'Ln 1, Col 823', and 'INS'.

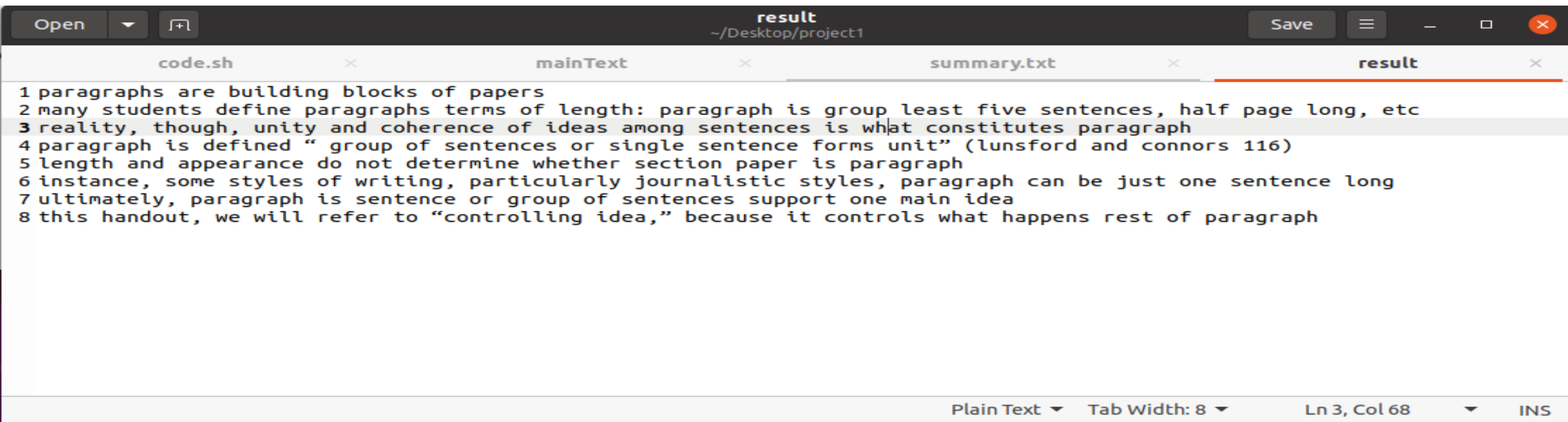
- Then it will automatically create temp & result files .

THE RUN AND OUTPUT

A terminal window titled 'anas@anas-VirtualBox: ~/Desktop/project1'. The window has a dark background with a search icon, a menu icon, and window control buttons (minimize, maximize, close) in the top right corner. The terminal shows the following sequence of commands and output:

```
anas@anas-VirtualBox:~/Desktop$ cd project1
anas@anas-VirtualBox:~/Desktop/project1$ chmod +x code.sh
anas@anas-VirtualBox:~/Desktop/project1$ ./code.sh
Please enter the file name
mainText
Please Enter the summary ratio
0.5
Every Thing is done well !!
anas@anas-VirtualBox:~/Desktop/project1$
```

- Here is the result file the contains the the sorted lines before taking the final result ((the final number of lines)).

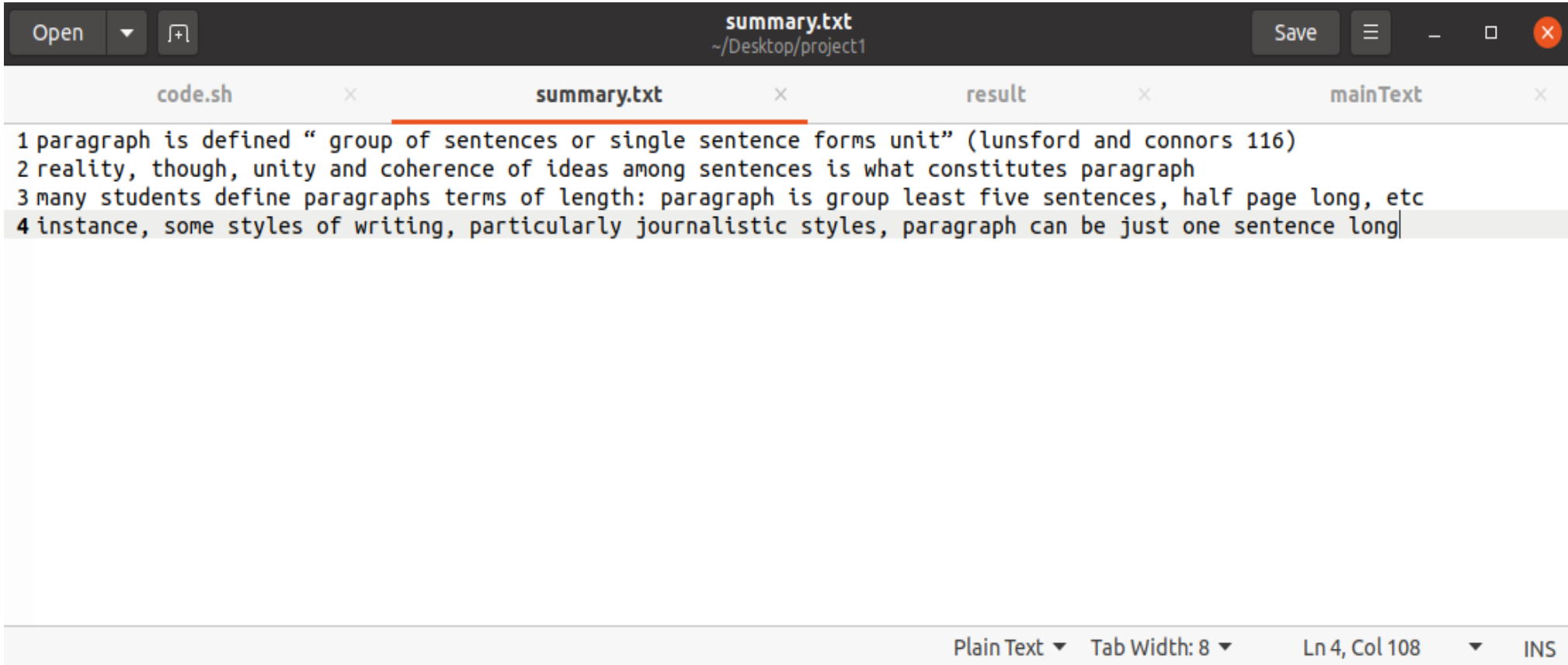


The screenshot shows a code editor window titled "result" with the path "~/Desktop/project1". The editor has four tabs: "code.sh", "mainText", "summary.txt", and "result". The "result" tab is active and displays the following text:

```
1 paragraphs are building blocks of papers
2 many students define paragraphs terms of length: paragraph is group least five sentences, half page long, etc
3 reality, though, unity and coherence of ideas among sentences is what constitutes paragraph
4 paragraph is defined "group of sentences or single sentence forms unit" (lunsford and connors 116)
5 length and appearance do not determine whether section paper is paragraph
6 instance, some styles of writing, particularly journalistic styles, paragraph can be just one sentence long
7 ultimately, paragraph is sentence or group of sentences support one main idea
8 this handout, we will refer to "controlling idea," because it controls what happens rest of paragraph
```

The status bar at the bottom indicates "Plain Text", "Tab Width: 8", "Ln 3, Col 68", and "INS".

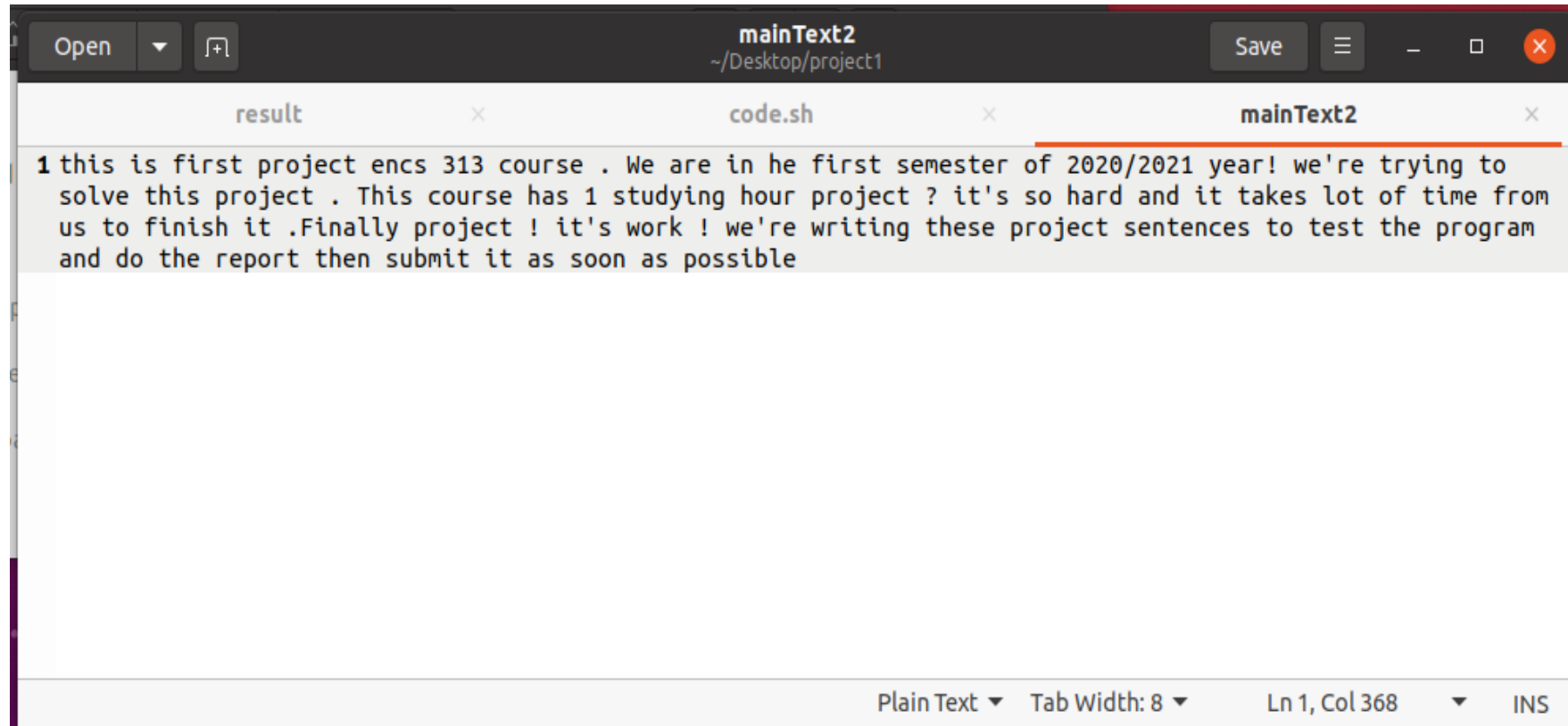
- Here's the final result will be in the summary.txt file, it will be created automatically .



The screenshot shows a code editor window with a dark theme. The title bar at the top indicates the file is 'summary.txt' located at '~/Desktop/project1'. The editor has four tabs: 'code.sh', 'summary.txt' (which is the active tab), 'result', and 'mainText'. The 'summary.txt' tab contains four lines of text, with the fourth line highlighted. The status bar at the bottom shows 'Plain Text', 'Tab Width: 8', 'Ln 4, Col 108', and 'INS'.

```
1 paragraph is defined " group of sentences or single sentence forms unit" (lunsford and connors 116)
2 reality, though, unity and coherence of ideas among sentences is what constitutes paragraph
3 many students define paragraphs terms of length: paragraph is group least five sentences, half page long, etc
4 instance, some styles of writing, particularly journalistic styles, paragraph can be just one sentence long
```

- TEST 2 :
- The main file that has the main paragraph before edti :

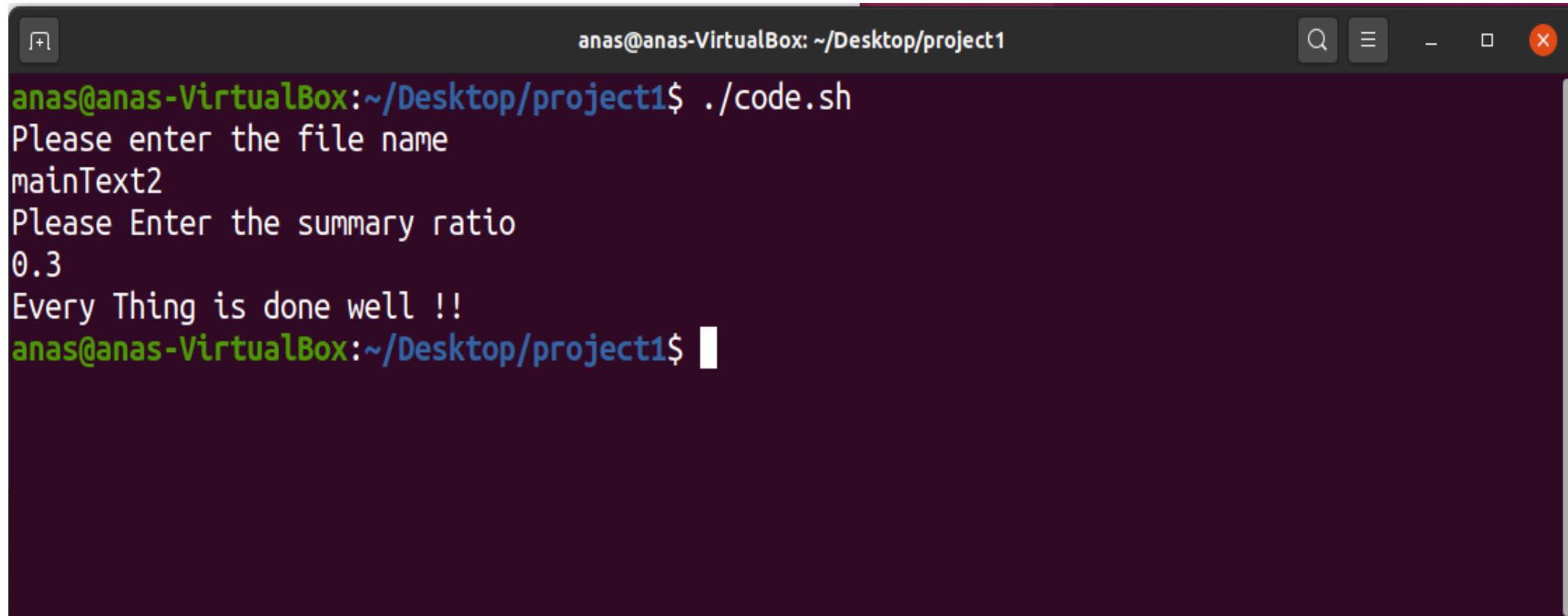


The image shows a screenshot of a text editor window. The window has a dark title bar with the text "mainText2" and the path "~/Desktop/project1". There are buttons for "Open", "Save", and window control icons. Below the title bar, there are three tabs: "result", "code.sh", and "mainText2". The "mainText2" tab is active and shows a paragraph of text. The text is as follows:

```
1 this is first project encs 313 course . We are in he first semester of 2020/2021 year! we're trying to  
solve this project . This course has 1 studying hour project ? it's so hard and it takes lot of time from  
us to finish it .Finally project ! it's work ! we're writing these project sentences to test the program  
and do the report then submit it as soon as possible
```

At the bottom of the window, there is a status bar with the text "Plain Text", "Tab Width: 8", "Ln 1, Col 368", and "INS".

THE RUN AND OUTPUT

A terminal window with a dark purple background and a grey title bar. The title bar contains the text 'anas@anas-VirtualBox: ~/Desktop/project1' and standard window control icons (search, menu, zoom, close). The terminal shows the execution of a script named 'code.sh'. The prompt is 'anas@anas-VirtualBox:~/Desktop/project1\$'. The script prompts for a file name, which is 'mainText2', and a summary ratio, which is '0.3'. The script then outputs 'Every Thing is done well !!' and returns to the prompt.

```
anas@anas-VirtualBox:~/Desktop/project1$ ./code.sh
Please enter the file name
mainText2
Please Enter the summary ratio
0.3
Every Thing is done well !!
anas@anas-VirtualBox:~/Desktop/project1$
```

THE RESULT FOR EXAMPLE 2

The screenshot shows a code editor window with three tabs: **result**, **code.sh**, and **mainText2**. The **result** tab is active and contains the following text:

```
1 this is first project encs 313 course
2 we are he first semester of 2020/2021 year
3 we're trying to solve this project
4 this course has 1 studying hour project
5 it's so hard and it takes lot of time from us to finish
6 finally project
7 it's work
8 we're writing these project sentences to test program and do report then submit it soon possible
```

Below the main editor, a smaller window titled **summary.txt** is open, showing the first three lines of the same text:

```
1 this is first project encs 313 course
2 this course has 1 studying hour project
3 finally project
```

The status bar at the bottom of the editor indicates the current file is **Plain Text**, with a **Tab Width: 8**, and the cursor is at **Ln 3, Col 16**. A small status bar on the right side of the summary.txt window shows **Ln 6, Col 4** and **INS**.