-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.

```
#!/bin/bash
echo "Please enter the file name " #the main file that has the sentences
read fileName
echo "Please Enter the summary ratio "
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

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

```
cat $fileName | sed "s/\<i\>//g" | sed "s/\<a\>//g" | sed "s/\<an\>//g" | sed "s/\<as\>//g" > temp #

cat temp | sed "s/\<at\>//g" | sed "s/\<the\>//g" | sed "s/\<by\>//g" | sed "s/\<in\>//g" > $fileNar

cat $fileName | sed "s/\<for\>//g" | sed "s/\<on\>//g" | sed "s/\<that\>//g" > temp

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 .

```
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 form 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 contains any thing 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 .

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

```
interSection=0
union=0
numOfLines=$(cat result | wc -l)
similarity=0 #to calculate the the similarity of the word with others
z=0
declare -a centrality=()
while [ $z -lt $numOfLines ] #assign all indexes to zero
do
        centrality[$z]=0
        z=$(($z+1))
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 campaired, and the centrality.

```
while [ $i -le $numOfLines ] #to reach the all lines
do
        k=0
        j=1
        k=$(($i+1))
        while [ $j -le $(($numOfLines - $i)) ] # to reach the next lines after line i
        do
                union=0
                interSection=0
                similarity=0
                str1=$(cat result | sed -n ${i}p)          # To get the i line from the result file
                for word1 in $str1             # loop will pass through each word in the string to compare it with words in the other one
                do
                        str2=$(cat result | sed -n ${k}p) #To get every word in line k((line i+1)
                        numOfWords1=$(echo $str1 | wc -w) #num of words to calculate the union
                        numOfWords2=$(echo $str2 | wc -w)
                        union=$(($numOfWords1 + $numOfWords2)) #union between the 2 lines
                        for word2 in $str2 #to compare every word in the line k
                        do
                                if [ "$word1" = "$word2" ] #check if the words the same.
                                then
                                        interSection=$(($interSection + 1)) #increment the intersection value by 1
                                fi
                        done
                done
                similarity=$(echo "scale=3;$interSection/$union" | bc -l ) #We used bc command to convert the integer number to float number.
                centrality[$i-1]=$(echo "scale=3;${centrality[$(($i-1))]}+$similarity" |bc -l) # adding similarity of the line compare to the other lines.
                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.
                k=$(($k+1)) #increment k that reach the other lines.
                j=$(($j+1)) #increment j that make sure the program will not go back to the last line.
        done
        i=$(($i+1)) #increment i to reach the all lines in the main file .
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.

```
declare -a lines=() # here i decalred an array to assign every line in an index , i will use it in sorting based on centrality .
i=0

while [ $i -lt $numOfLines ] # to initialize the all indexes contents to 0.
do
        lines[$i]=0
        i=$(($i+1))
done


for ((j=0 ; j < $numOfLines;j++)) #to assign all lines in array , to reach every line separately .
do
        k=$(($j+1)) # to start from line j+1
        lines[$j]=$(cat result | sed -n ${k}p)
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.

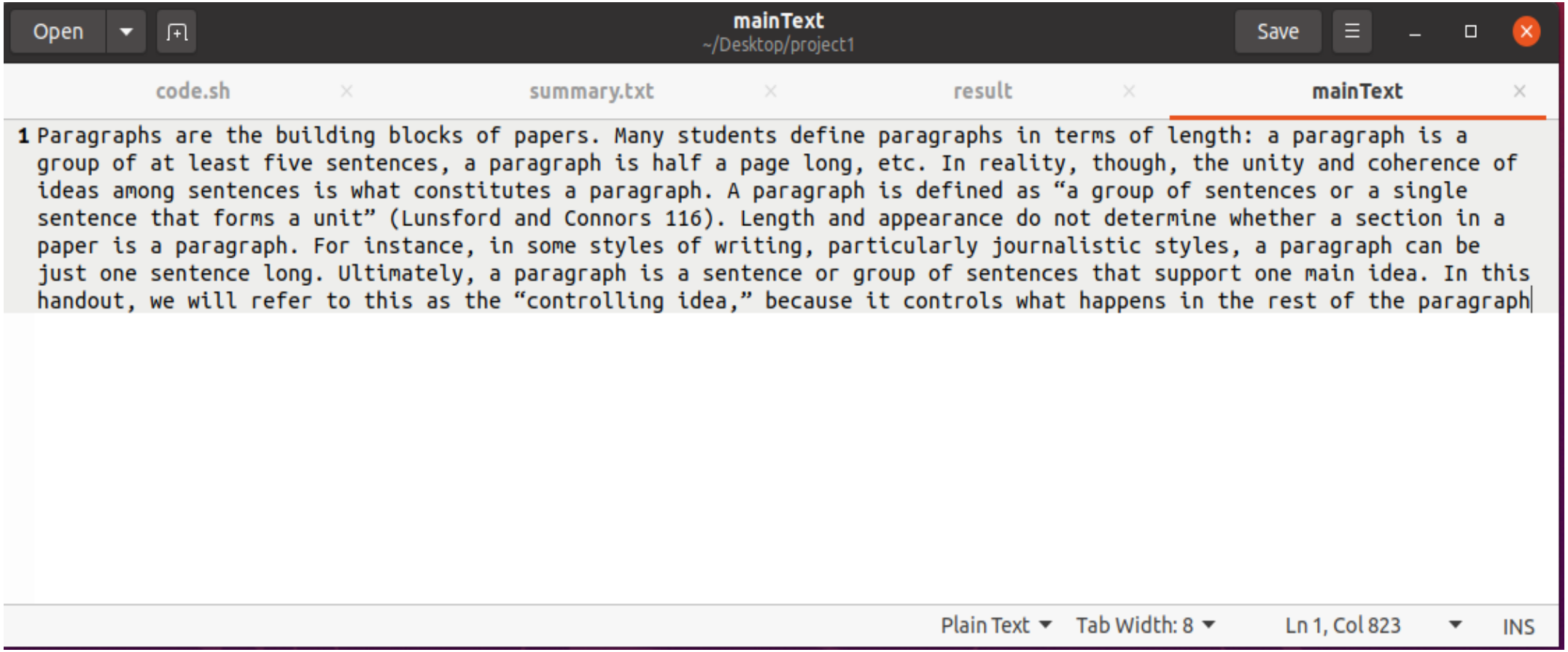
```
arrayLength=${#centrality[@]}
while [ $i -lt $arrayLength ] #here we sorted the centrality array with lines array ..we need nested loops
do
        temp=0 #this variable to switch between the array index contents
        j=$(($i+1))
        while [ $j -lt $arrayLength ]
        do
                if (( $(echo " ${centrality[$j]} > ${centrality[$i]}" | bc -l) )) #compare between the i index with next indexs
                then
                        #here is the switching between centrality values and lines.

                        temp=${centrality[$i]}
                        centrality[$i]=${centrality[$j]}
                        centrality[$j]=$temp
                        str=${lines[$i]}
                        lines[i]=${lines[j]}
                        lines[j]=$str
                fi
                j=$(($j+1)) #increment j value by 1
        done
        i=$(($i+1)) #increment i value by 1
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 I value more than the rationOfLines , if it’s not , the program will add the line from the sorted array to the summary.txt file .

```
ratioOfLines=$(echo "scale=2; $summaryRatio * $numOfLines" | bc -l ) # this is the number of lines will be showen in the final file (summary.txt)
: '
        if the ratio of lines result was a floating point number , it will be approximate .
'
i=0
echo -n "" > summary.txt
while true
do
        if (( $(echo " $i < $ratioOfLines" | bc -l) ))
        then
                echo ${lines[$i+1]} >> summary.txt # appending the final lines to the summary.txt file. after make sure that the file is empty.
                i=$(($i+1))
        else
                break
        fi
done
```

- The main file before editing :



- Then it will automatically create temp & result files .

THE RUN AND OUTPUT

```
anas@anas-VirtualBox: ~/Desktop/project1
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)).

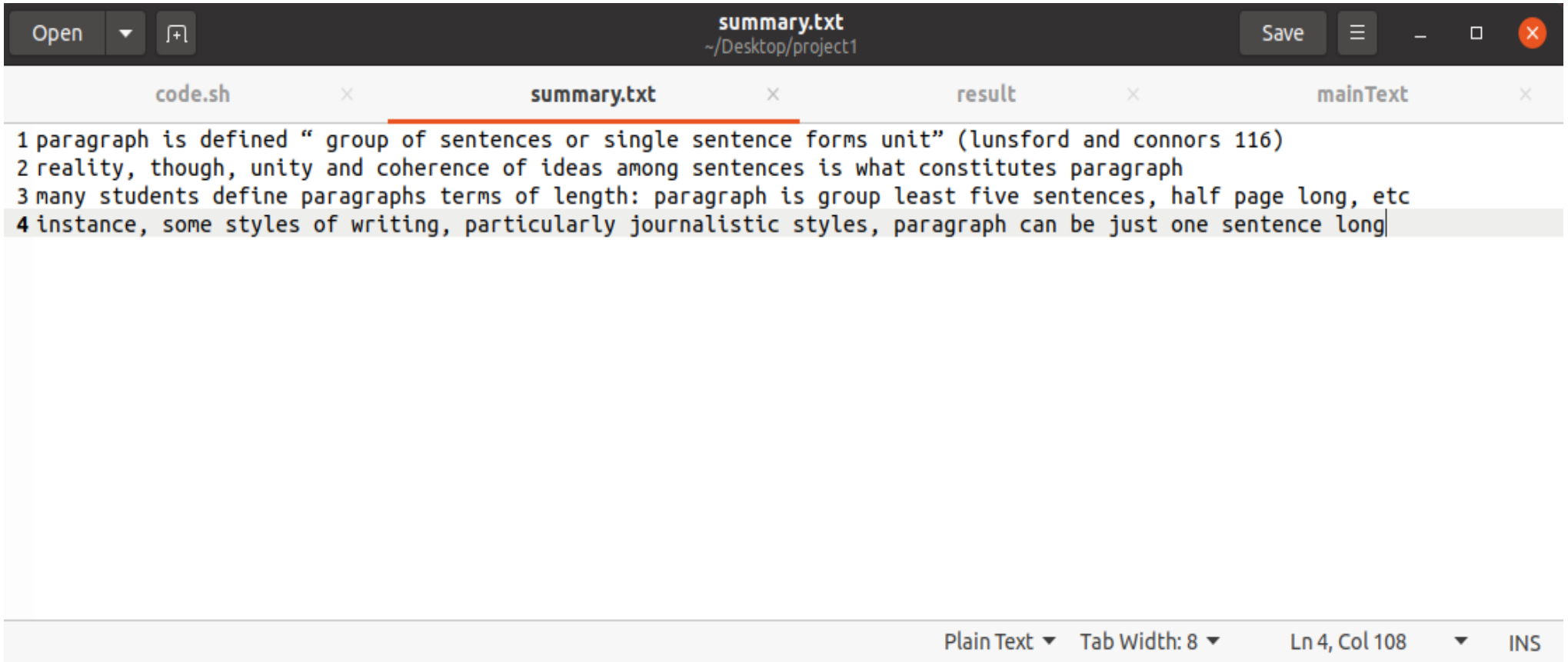
Open | result ~/Desktop/project1 | Save

code.sh | mainText | summary.txt | result

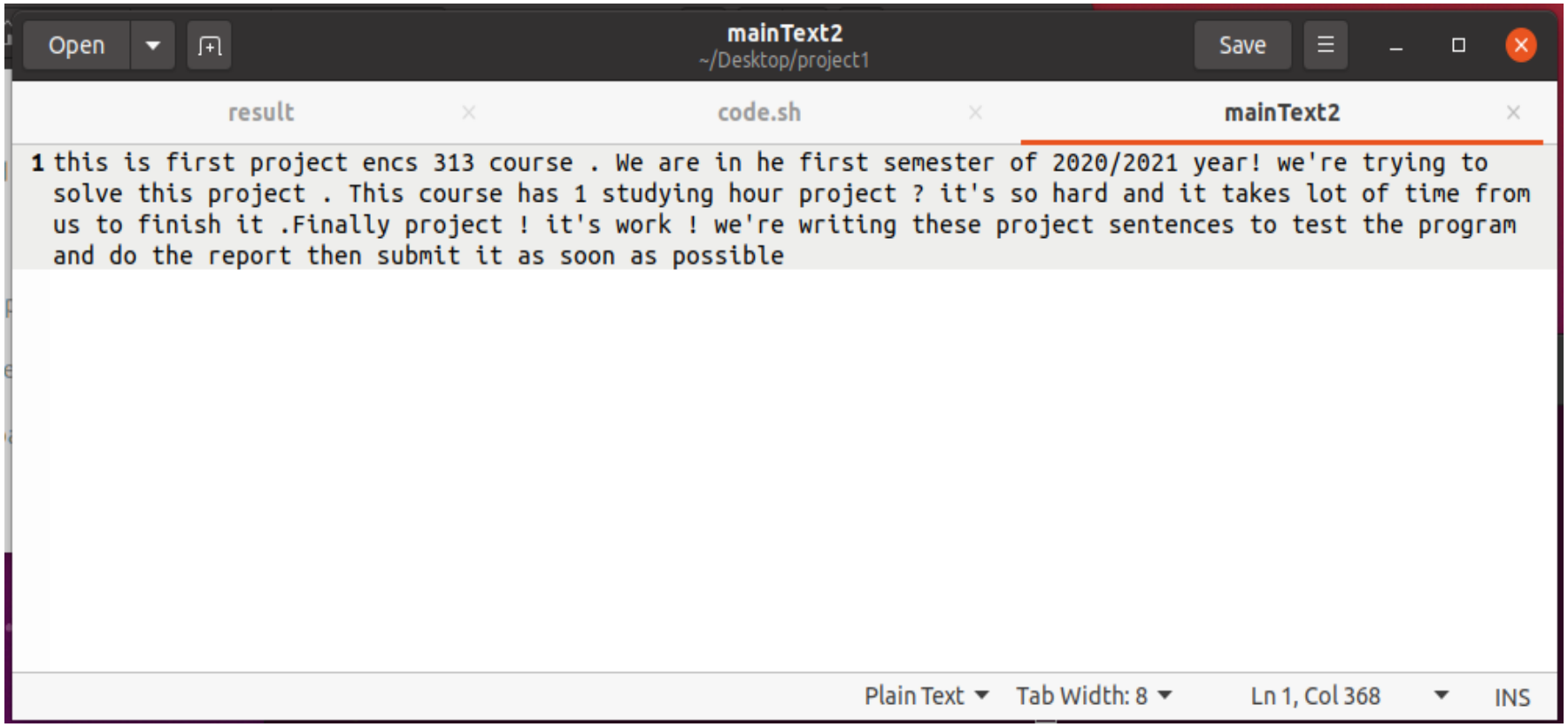
```
paragraphs are building blocks of papers
many students define paragraphs terms of length: paragraph is group least five sentences, half page long, etc
reality, though, unity and coherence of ideas among sentences is what constitutes paragraph
paragraph is defined " group of sentences or single sentence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

Plain Text | Tab Width: 8 | Ln 3, Col 68 | INS

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



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



THE RUN AND OUTPUT

```
anas@anas-VirtualBox: ~/Desktop/project1
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

Open result ~/Desktop/project1 Save

result code.sh mainText2

```
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
```

n 6, Col 4 INS

Open summary.txt ~/Desktop/project1 Save

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

Plain Text Tab Width: 8 Ln 3, Col 16 INS