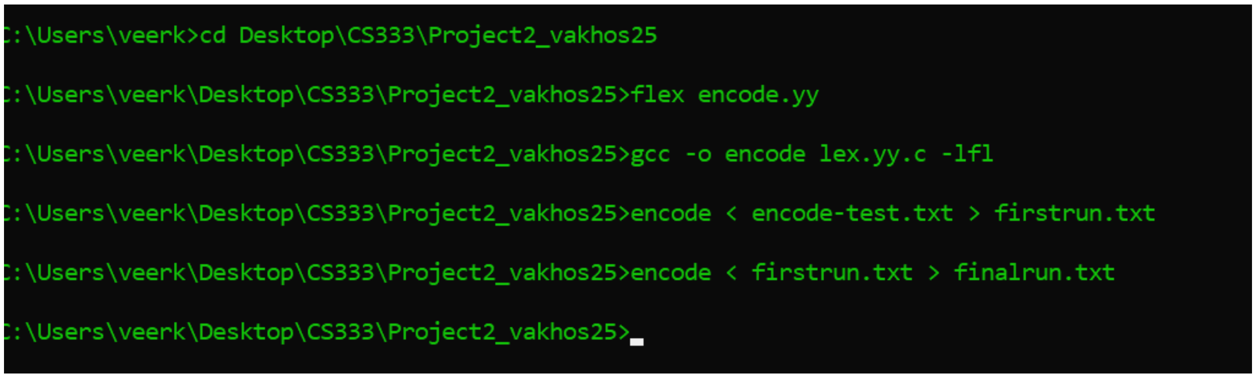
Veer Khosla

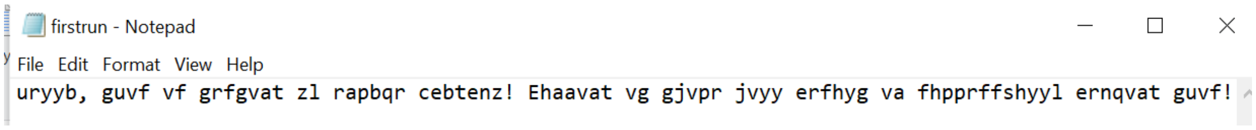
Project 2

1. Task 1



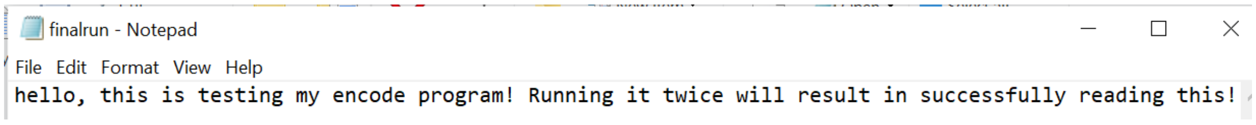
```
:\Users\veerk>cd Desktop\CS333\Project2_vakhos25

:\Users\veerk\Desktop\CS333\Project2_vakhos25>flex encode.yy

:\Users\veerk\Desktop\CS333\Project2_vakhos25>gcc -o encode lex.yy.c -lfl

:\Users\veerk\Desktop\CS333\Project2_vakhos25>encode < encode-test.txt > firstrun.txt

:\Users\veerk\Desktop\CS333\Project2_vakhos25>encode < firstrun.txt > finalrun.txt

:\Users\veerk\Desktop\CS333\Project2_vakhos25>
```

Firstrun.txt



```
uryyb, guvf vf grfgvat zl rapbqr cebtenz! Ehaavat vg gjvpr jvyy erfhyg va fhpprffshyyl ernqvat guvf!
```

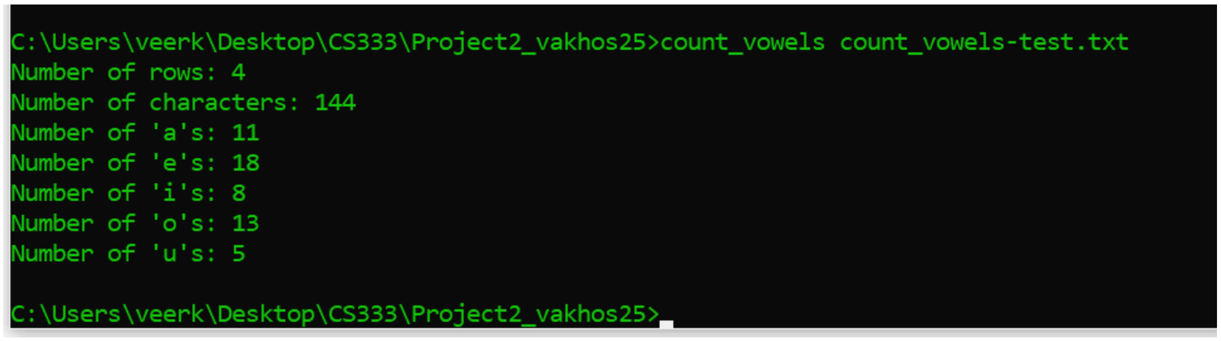
Finalrun.txt



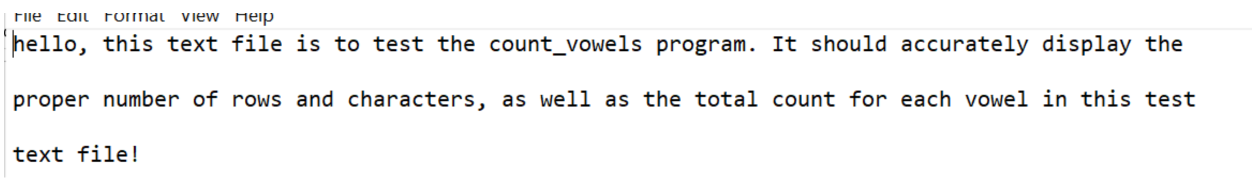
```
hello, this is testing my encode program! Running it twice will result in successfully reading this!
```

The task is to create a Flex program named "encode" that encrypts input text using a Caesar cipher with a shift of 13 positions. Only alphabetical characters ([a-z] or [A-Z]) are shifted, while non-alphabetical characters remain unchanged. If the shifted value exceeds the range of 'z' or 'Z', it wraps back to the beginning of the alphabet. The program's output is the firstrun and finalrun text files. It demonstrates the encryption process by providing example inputs and their corresponding encrypted outputs.

2. Task 2



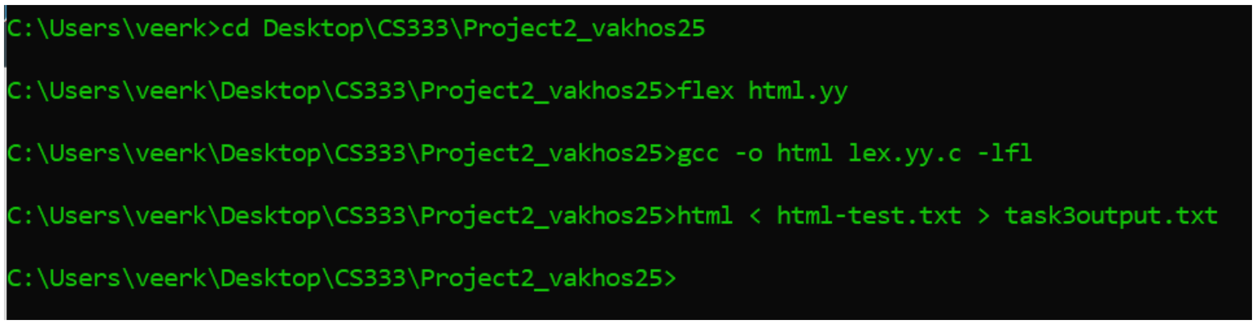
```
C:\Users\veerk\Desktop\CS333\Project2_vakhos25>count_vowels count_vowels-test.txt
Number of rows: 4
Number of characters: 144
Number of 'a's: 11
Number of 'e's: 18
Number of 'i's: 8
Number of 'o's: 13
Number of 'u's: 5

C:\Users\veerk\Desktop\CS333\Project2_vakhos25>
```



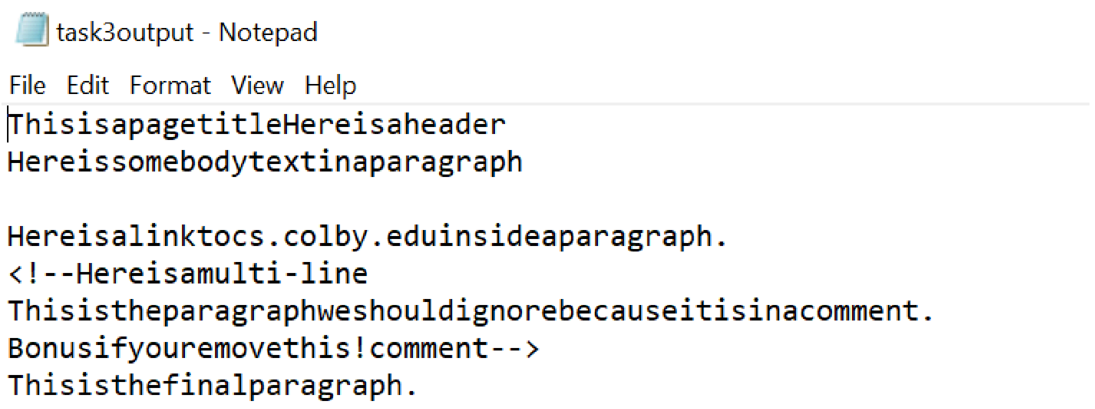
```
hello, this text file is to test the count_vowels program. It should accurately display the

proper number of rows and characters, as well as the total count for each vowel in this test

text file!
```

This program calculated the total number of rows and characters, as well as the frequency of occurrence for each vowel, considering both lowercase and uppercase instances. The program was tested using the text file shown above, and the terminal above details each of the vowel occurrences.

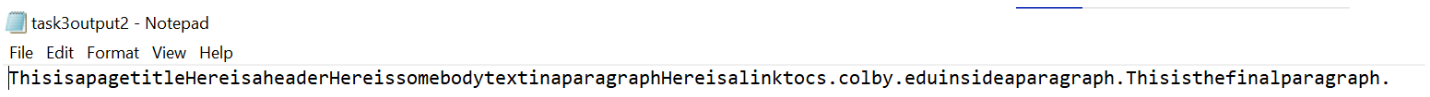
3. Task 3



```
C:\Users\veerk>cd Desktop\CS333\Project2_vakhos25

C:\Users\veerk\Desktop\CS333\Project2_vakhos25>flex html.yy

C:\Users\veerk\Desktop\CS333\Project2_vakhos25>gcc -o html lex.yy.c -lfl

C:\Users\veerk\Desktop\CS333\Project2_vakhos25>html < html-test.txt > task3output.txt

C:\Users\veerk\Desktop\CS333\Project2_vakhos25>
```



```
ThisisapagetitleHereisaheader
Hereissomebodytextinaparagraph

Hereisalinktocs.colby.eduinsideaparagraph.
<!--Hereisamulti-line
Thisistheparagraphweshouldignorebecauseitisinacomment.
Bonusifyouremovethis!comment-->
Thisisthefinalparagraph.
```



```
ThisisapagetitleHereisaheaderHereissomebodytextinaparagraphHereisalinktocs.colby.eduinsideaparagraph.Thisisthefinalparagraph.
```

I created a Flex program to clean up HTML files. It removed all tags and comments and also got rid of extra spaces, except when there's a <p> tag, where it kept a blank line. The program went through the HTML file, deleting tags and comments, and tidying up spaces. It left a blank line for paragraphs (<p> tags) to maintain the layout. After running the program, the HTML file looked neat without any tags, comments, or unnecessary spaces, keeping the paragraphs intact. Testing with different HTML files showed the program worked reliably for cleaning up HTML content.