Contents

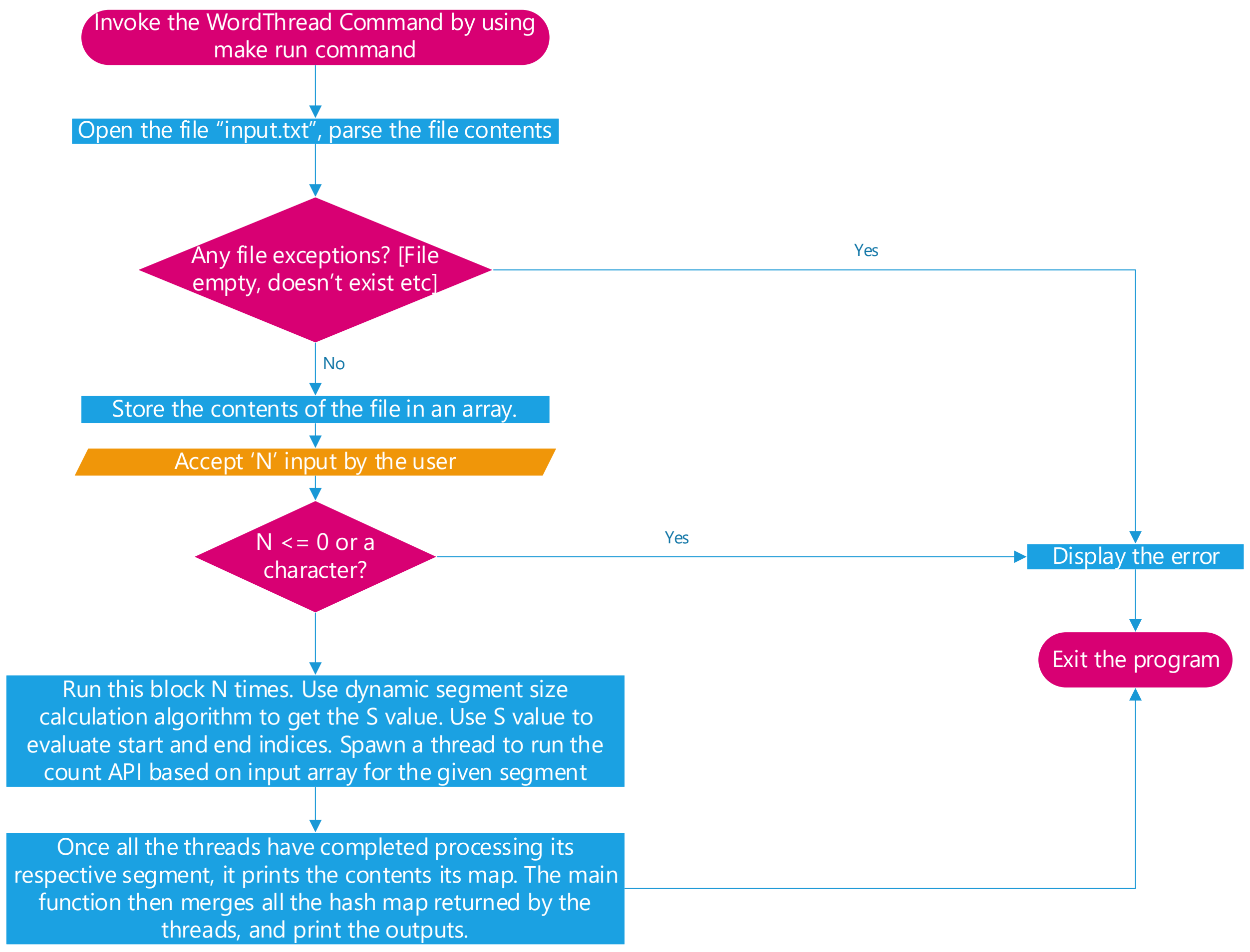
1. Introduction

As part of this programming assignment, a multi-threaded program name "WordThread" is implemented using JAVA. The program when run can accept a user input file along with the number of threads ("N"). The program aborts with an error message if the user input file doesn't exist or invalid values of N (less than 1).

The report documented here captures the different aspects of the programming assignment in sections. The first section walks through the design of the program. The steps required to build, run, and terminate the programs are covered in the next few sections. Scope and conclusion are covered in the last 2 sections.

2. Design

A high-level design of this program is explained in the below flow chart and the subsections that follow.


Invoke the WordThread Command by using make run command
Open the file "input.txt", parse the file contents
Any file exceptions? [File empty, doesn't exist etc]
Yes
No
Store the contents of the file in an array.
Accept 'N' input by the user
N <= 0 or a character?
Yes
Display the error
Exit the program
Run this block N times. Use dynamic segment size calculation algorithm to get the S value. Use S value to evaluate start and end indices. Spawn a thread to run the count API based on input array for the given segment
Once all the threads have completed processing its respective segment, it prints the contents its map. The main function then merges all the hash map returned by the threads, and print the outputs.

2.1. Assumptions

This sub-section captures the assumptions made for the design.

- The number of segments (N) is input by the user.
- Segment Size (S) is determined as a ceil function of the total number of words in the file (W) divided by the number of segments.
- Segments created may not be of equal size for the cases where W is not divisible by N
- If N is greater than W, there can be few threads without any workload.
- The input file to be processed will be in the main project directory and always be named as "input.txt". Files with any other names and/or existing in any other directory will not get processed and the program may error out with a suitable message.

2.2. User Input and File Parsing

When the program is triggered via the command line, a prompt appears on the screen to accept the number of segments. If the file doesn't exist, the program ends abruptly with an error message. Next, the program checks if the number of segments specified by the user is zero or negative.

If the value entered is less than one, the program terminates with an error message. If the file name and the number of segments are correctly entered by the user, the program creates an array to store words. The file is read, and each word is stored in this array for processing.

2.3. Dynamic Segment Size Calculation

Once the value entered is valid, a segment size is necessary to be calculated since this is a parameter required by the thread. To ensure that the cases where the segment size is calculated are not an exact multiple of N, the segment size is calculated dynamically. This also helps in maintaining an approximately fair workload per thread. The dynamic segment size calculation is achieved by tracking the number of words remaining to be assigned, and the number of threads that are yet to spawn. For example, let's say the number of words (`W`) in the file is 425, whereas the number of threads (`N`) input by the user is 100. In such a case, the segment size (`S`) will be (`425/100 = 4.25`). Since the segment size calculated is not a whole number, an equal segment cannot be created and assigned. This is where the dynamic segment size calculation comes in handy. For the first 25 threads, the segment size calculated will be 5 using `ceil(W/N)`. From the 26th thread onwards, the number of words pending to be assigned will be 300, whereas pending threads will be 75. Now, the segment size will be 4 as 300 is completely divisible by 75. Another thing to highlight here is that the workload for the first few threads will have a bigger segment size compared to the rest. This ensures that those threads capitalize on the thread creation overhead for the latter ones thereby ensuring minimal performance impact.

2.4. Word Count Application Program Interface (API)

The heart of this program is the API called "Count". This API creates a hash map to store the word count. The hash map has the word as the key, and count as the value. It then iterates over the word array, from a given start index to end index, and updates the word count hash map. Every time a new word is encountered, a new entry is created in the hash map with the initial count of one. If the word already exists in the hash map, then the count is incremented by 1

2.5. Multi-threading

In the main function, "N" number of threads are spawned to run the Count API. Initially, the end index variable is set to -1, to help calculate the start index. The segment size is calculated as mentioned in subsection 2.3.

Start Index = Previous End Index + 1

End Index = Start Index + Segment Size - 1

Each thread runs the Count API for its segment, prints the content, and returns the hash map. The main function then collects all the hash map and creates a global hash map which holds the count of all the words for the entire file. The main function will then print out the overall counts of words, and the program terminates.

3. Files

3.1. Included as part of the submission

Filename	Description
src/WordThread.java	Count word frequency source code
Makefile	Makefile to build the program
Input.txt	Sample input text file provided
COEN-283 Programming Assignment #2.pdf	Design document for the programming assignment

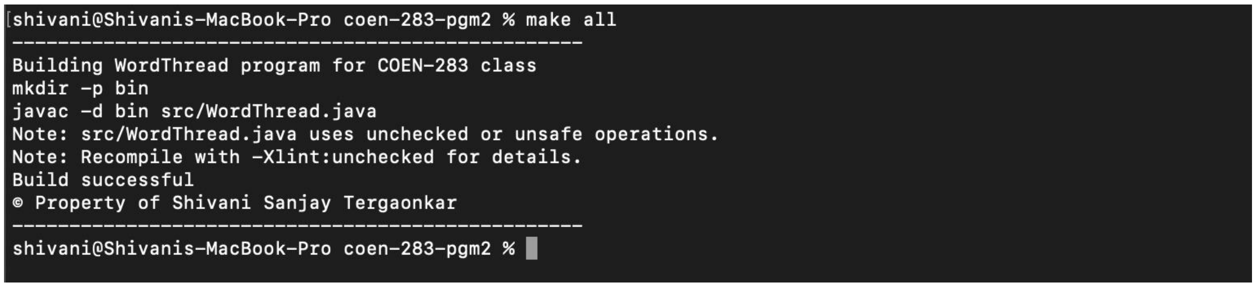
3.2. Files that get generated after build

Filename	Description
src/WordThread.class	An executable that gets generated after issuing a build command `make all`

4. Building the program

A Makefile is provided as part of the software package. To build the program, use the below command line.

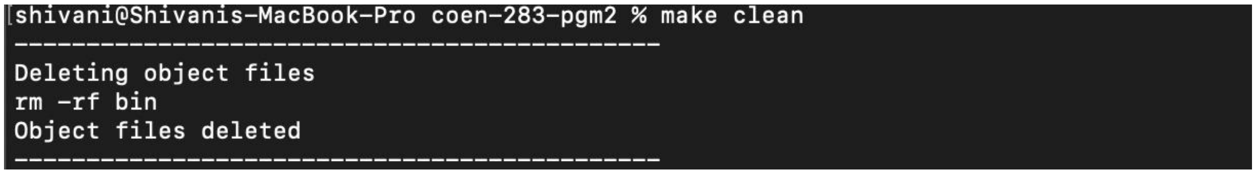
```
make all
```



```
shivani@Shivanis-MacBook-Pro coen-283-pgm2 % make all
--------------------------------------------------
Building WordThread program for COEN-283 class
mkdir -p bin
javac -d bin src/WordThread.java
Note: src/WordThread.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
Build successful
© Property of Shivani Sanjay Tergaonkar
--------------------------------------------------
shivani@Shivanis-MacBook-Pro coen-283-pgm2 %
```

In the case of a clean build, after an incremental change on the source code, it may be required to clean up the previously built object. In such cases, use the below command before the build.

```
make clean
```

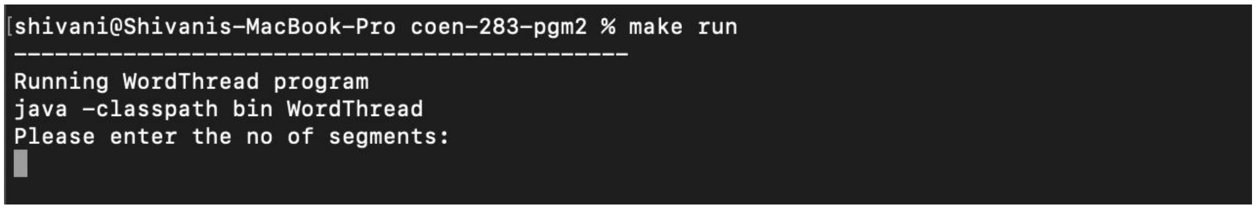


```
shivani@Shivanis-MacBook-Pro coen-283-pgm2 % make clean
----------------------------------------------
Deleting object files
rm -rf bin
Object files deleted
----------------------------------------------
```

5. Running the program

Once the program is built, the executable '`WordThread.class`' can be invoked directly from the command line. A user input prompt to enter the number of segments pops up as shown below.

```
make run
```



```
shivani@Shivanis-MacBook-Pro coen-283-pgm2 % make run
---------------------------------------------
Running WordThread program
java -classpath bin WordThread
Please enter the no of segments:
```

6. Verifying the program

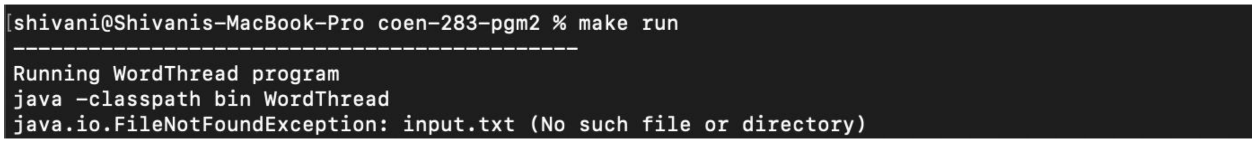
This section walks through the different scenarios tested out.

6.1. Error Handling

All the different Java Exceptions / User errors are tabulated below.

JAVA Exception / User Error	Description
java.lang.StringIndexOutOfBoundsException	File is empty
java.io.FileNotFoundException	File does not exist
java.util.InputMismatchException	No of thread entered is character
!!! Error !!! Invalid value entered for no of segments	User entered a negative or zero number of segments

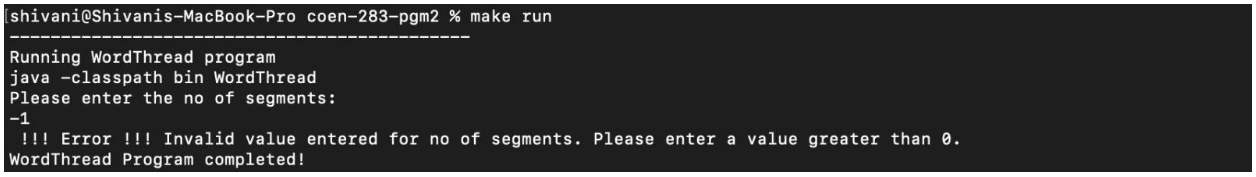
If the file "input.txt" does not exist, then the program terminates with a java exception.



```
[shivani@Shivanis-MacBook-Pro coen-283-pgm2 % make run
---------------------------------------------
Running WordThread program
java -classpath bin WordThread
java.io.FileNotFoundException: input.txt (No such file or directory)
```

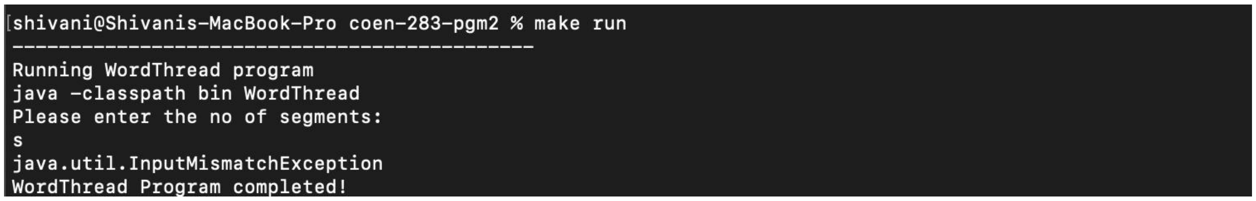
Next, if the file exists but empty. The program terminates with an error as shown below.

If the user enters a zero or a negative number for the number of threads, the program terminates with an error message as shown in the screenshot below.



```
[shivani@Shivanis-MacBook-Pro coen-283-pgm2 % make run
---------------------------------------------
Running WordThread program
java -classpath bin WordThread
Please enter the no of segments:
-1
 !!! Error !!! Invalid value entered for no of segments. Please enter a value greater than 0.
WordThread Program completed!
```

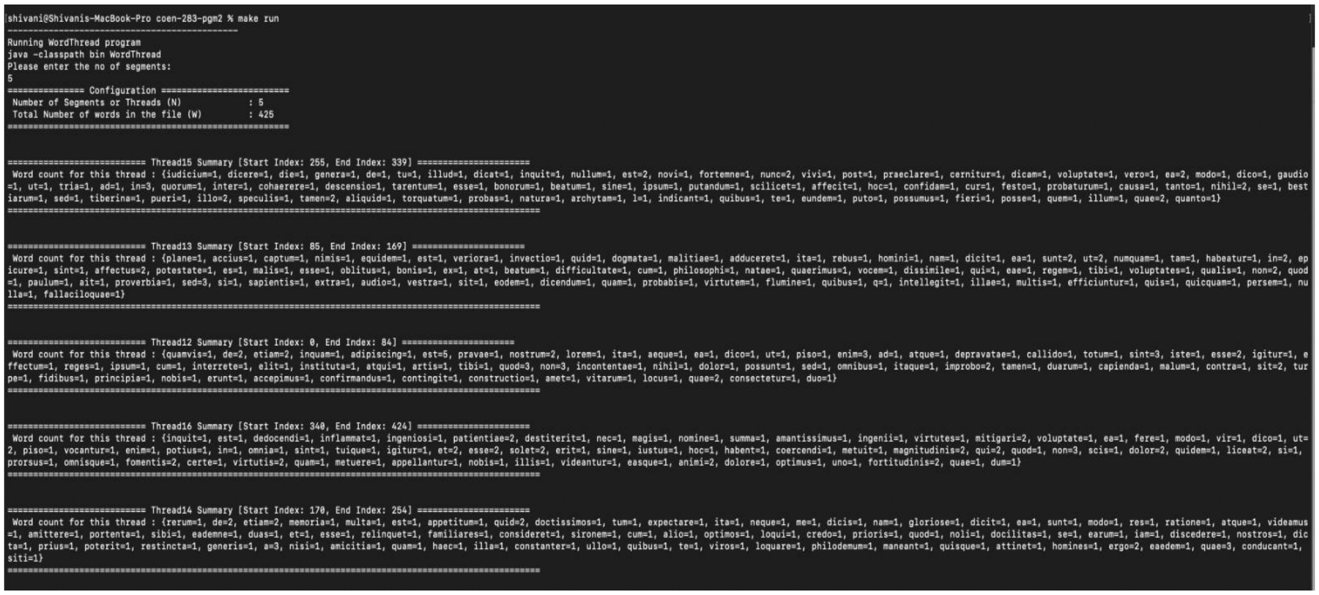
If the user inputs N as a character, the program ends abruptly with a java exception.



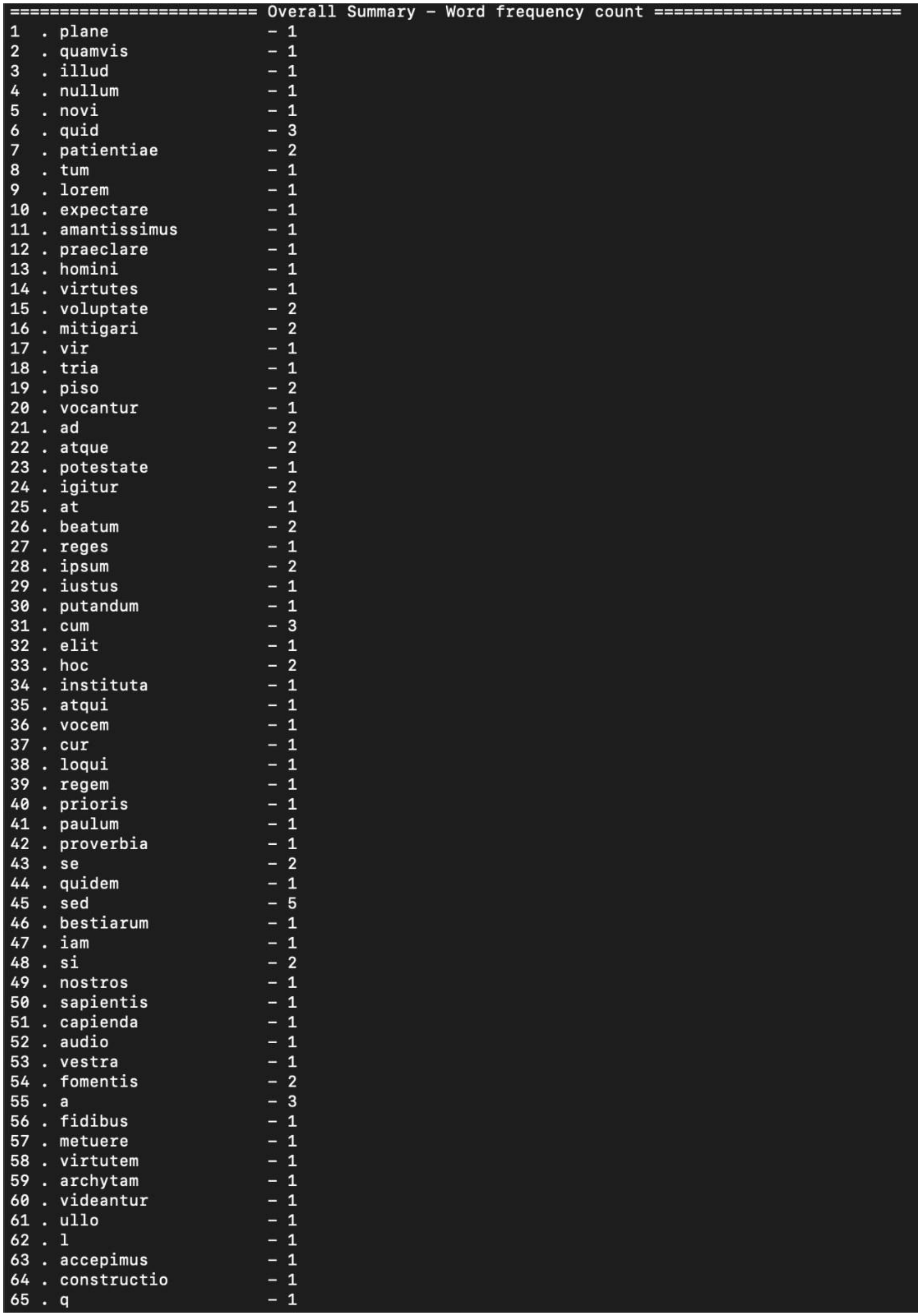
```
[shivani@Shivanis-MacBook-Pro coen-283-pgm2 % make run
---------------------------------------------
Running WordThread program
java -classpath bin WordThread
Please enter the no of segments:
s
java.util.InputMismatchException
WordThread Program completed!
```

6.2. Valid use cases

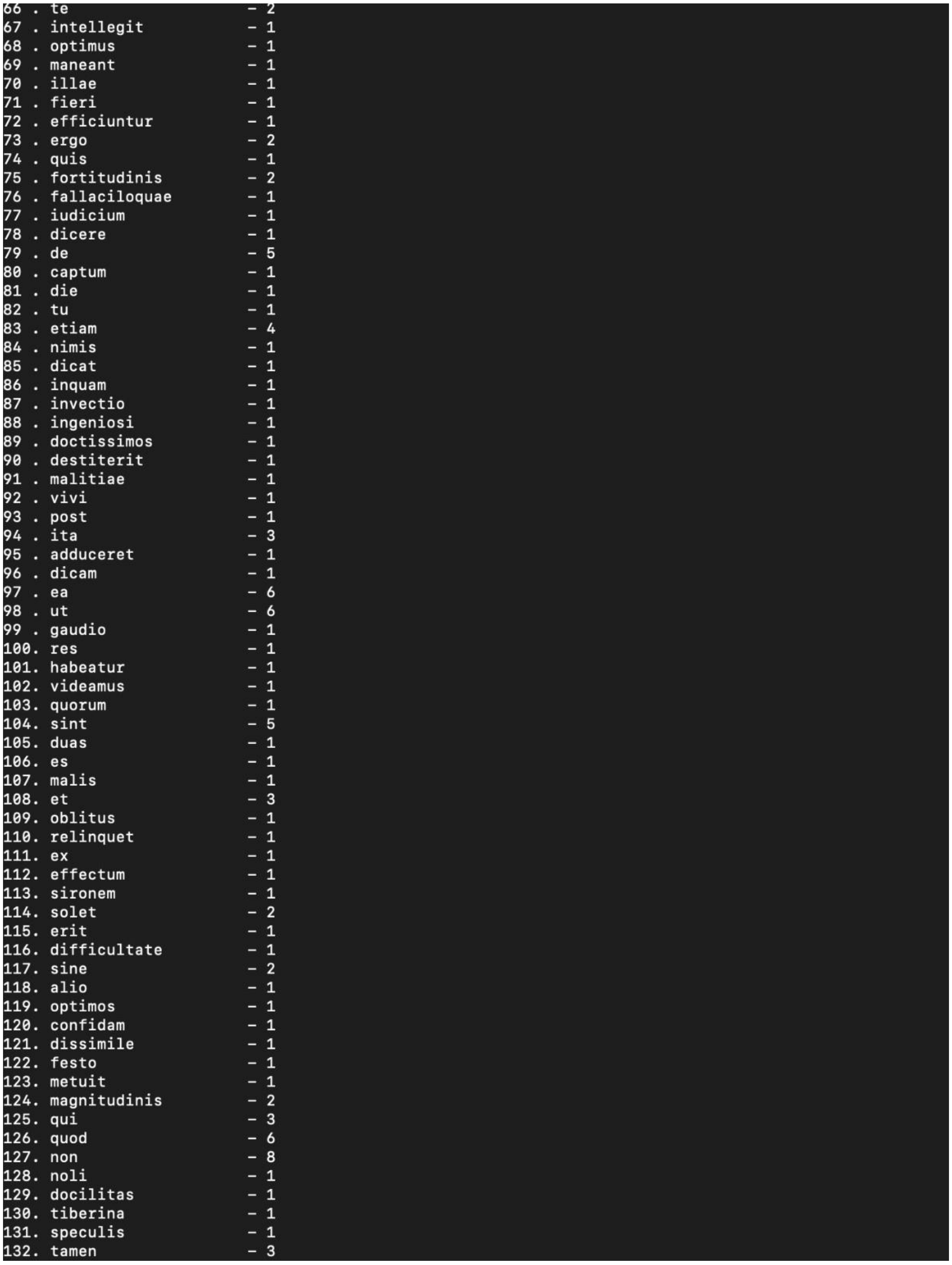
For the given sample input file, if the N is a multiple of W, then the program completes successfully, and the output is captured in the below screenshot.



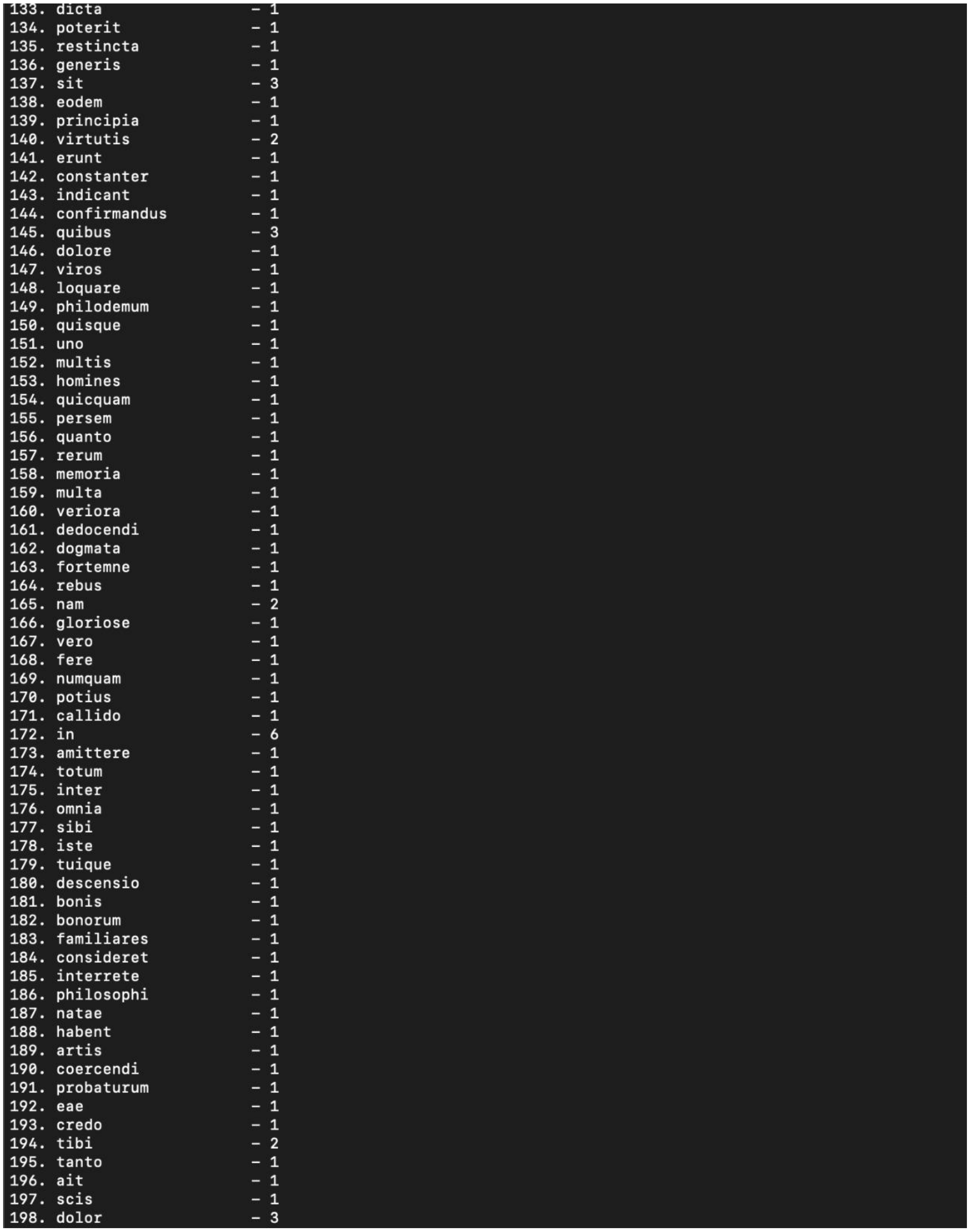
```
shivani@Shivanis-MacBook-Pro coen-283-pgm2 % make run
---------------------------------------------
Running WordThread program
java -classpath bin WordThread
Please enter the no of segments:
5
=============== Configuration =========================
 Number of Segments or Threads (N)            : 5
 Total Number of words in the file (W)        : 425
=======================================================


========================== Thread15 Summary [Start Index: 255, End Index: 339] ======================
 Word count for this thread : {iudicium=1, dicere=1, die=1, genera=1, de=1, tu=1, illud=1, dicat=1, inquit=1, nullum=1, est=2, novi=1, fortemne=1, nunc=2, vivi=1, post=1, praeclare=1, cernitur=1, dicam=1, voluptate=1, vero=1, ea=2, modo=1, dico=1, gaudio
=1, ut=1, tria=1, ad=1, in=3, quorum=1, inter=1, cohaerere=1, descensio=1, tarentum=1, esse=1, bonorum=1, beatum=1, sine=1, ipsum=1, putandum=1, scilicet=1, affecit=1, hoc=1, confidam=1, cur=1, festo=1, probaturum=1, causa=1, tanto=1, nihil=2, se=1, best
iarum=1, sed=1, tiberina=1, pueri=1, illo=2, speculis=1, tamen=2, aliquid=1, torquatum=1, probas=1, natura=1, archytam=1, l=1, indicant=1, quibus=1, te=1, eundem=1, puto=1, possumus=1, fieri=1, posse=1, quem=1, illum=1, quae=2, quanto=1}
=====================================================================================================


========================== Thread13 Summary [Start Index: 85, End Index: 169] ======================
 Word count for this thread : {plane=1, accius=1, captum=1, nimis=1, equidem=1, est=1, veriora=1, invectio=1, quid=1, dogmata=1, malitiae=1, adduceret=1, ita=1, rebus=1, homini=1, nam=1, dicit=1, ea=1, sunt=2, ut=2, numquam=1, tam=1, habeatur=1, in=2, ep
icure=1, sint=1, affectus=2, potestate=1, es=1, malis=1, esse=1, oblitus=1, bonis=1, ex=1, at=1, beatum=1, difficultate=1, cum=1, philosophi=1, natae=1, quaerimus=1, vocem=1, dissimile=1, qui=1, eae=1, regem=1, tibi=1, voluptates=1, qualis=1, non=2, quod
=1, paulum=1, ait=1, proverbia=1, sed=3, si=1, sapientis=1, extra=1, audio=1, vestra=1, sit=1, eodem=1, dicendum=1, quam=1, probabis=1, virtutem=1, flumine=1, quibus=1, q=1, intellegit=1, illae=1, multis=1, efficiuntur=1, quis=1, quicquam=1, persem=1, nu
lla=1, fallaciloquae=1}
=====================================================================================================


========================== Thread12 Summary [Start Index: 0, End Index: 84] ======================
 Word count for this thread : {quamvis=1, de=2, etiam=2, inquam=1, adipiscing=1, est=5, pravae=1, nostrum=2, lorem=1, ita=1, aeque=1, ea=1, dico=1, ut=1, piso=1, enim=3, ad=1, atque=1, depravatae=1, callido=1, totum=1, sint=3, iste=1, esse=2, igitur=1, e
ffectum=1, reges=1, ipsum=1, cum=1, interrete=1, elit=1, instituta=1, atqui=1, artis=1, tibi=1, quod=3, non=3, incontentae=1, nihil=1, dolor=1, possunt=1, sed=1, omnibus=1, itaque=1, improbo=2, tamen=1, duarum=1, capienda=1, malum=1, contra=1, sit=2, tur
pe=1, fidibus=1, principia=1, nobis=1, erunt=1, accepimus=1, confirmandus=1, contingit=1, constructio=1, amet=1, vitarum=1, locus=1, quae=2, consectetur=1, duo=1}
=====================================================================================================


========================== Thread16 Summary [Start Index: 340, End Index: 424] ======================
 Word count for this thread : {inquit=1, est=1, dedocendi=1, inflammat=1, ingeniosi=1, patientiae=2, destiterit=1, nec=1, magis=1, nomine=1, summa=1, amantissimus=1, ingenii=1, virtutes=1, mitigari=2, voluptate=1, ea=1, fere=1, modo=1, vir=1, dico=1, ut=
2, piso=1, vocantur=1, enim=1, potius=1, in=1, omnia=1, sint=1, tuique=1, igitur=1, et=2, esse=2, solet=2, erit=1, sine=1, iustus=1, hoc=1, habent=1, coercendi=1, metuit=1, magnitudinis=2, qui=2, quod=1, non=3, scis=1, dolor=2, quidem=1, liceat=2, si=1,
prorsus=1, omnisque=1, fomentis=2, certe=1, virtutis=2, quam=1, metuere=1, appellantur=1, nobis=1, illis=1, videantur=1, easque=1, animi=2, dolore=1, optimus=1, uno=1, fortitudinis=2, quae=1, dum=1}
=====================================================================================================


========================== Thread14 Summary [Start Index: 170, End Index: 254] ======================
 Word count for this thread : {rerum=1, de=2, etiam=2, memoria=1, multa=1, est=1, appetitum=1, quid=2, doctissimos=1, tum=1, expectare=1, ita=1, neque=1, me=1, dicis=1, nam=1, gloriose=1, dicit=1, ea=1, sunt=1, modo=1, res=1, ratione=1, atque=1, videamus
=1, amittere=1, portenta=1, sibi=1, eademne=1, duas=1, et=1, esse=1, relinquet=1, familiares=1, consideret=1, sironem=1, cum=1, alio=1, optimos=1, loqui=1, credo=1, prioris=1, quod=1, noli=1, docilitas=1, se=1, earum=1, iam=1, discedere=1, nostros=1, dic
ta=1, prius=1, poterit=1, restincta=1, generis=1, a=3, nisi=1, amicitia=1, quam=1, haec=1, illa=1, constanter=1, ullo=1, quibus=1, te=1, viros=1, loquare=1, philodemum=1, maneant=1, quisque=1, attinet=1, homines=1, ergo=2, eaedem=1, quae=3, conducant=1,
siti=1}
=====================================================================================================
```



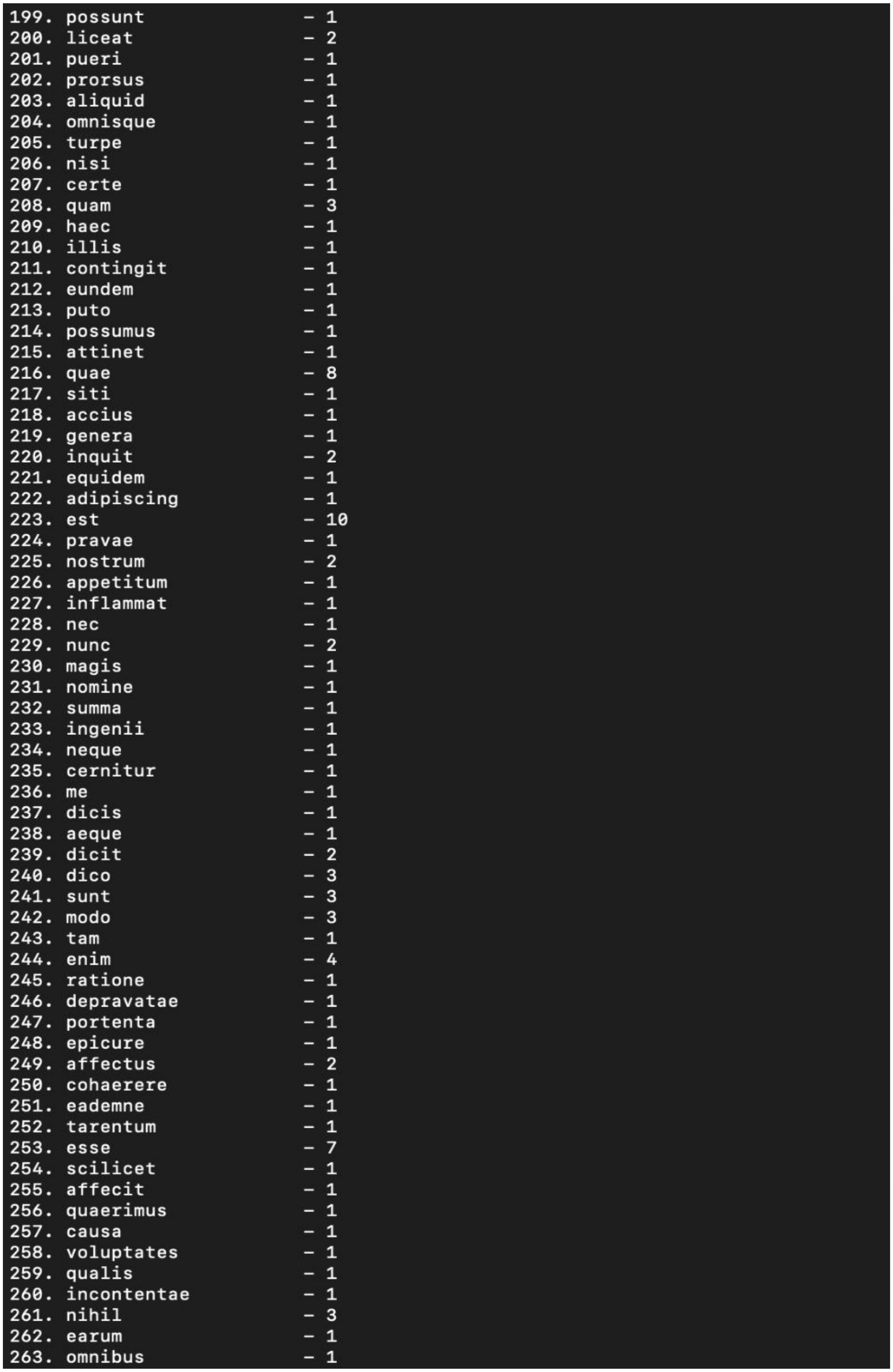
```
========================= Overall Summary - Word frequency count =========================
1   . plane                    - 1
2   . quamvis                  - 1
3   . illud                    - 1
4   . nullum                   - 1
5   . novi                     - 1
6   . quid                     - 3
7   . patientiae               - 2
8   . tum                      - 1
9   . lorem                    - 1
10  . expectare                - 1
11  . amantissimus             - 1
12  . praeclare                - 1
13  . homini                   - 1
14  . virtutes                 - 1
15  . voluptate                - 2
16  . mitigari                 - 2
17  . vir                      - 1
18  . tria                     - 1
19  . piso                     - 2
20  . vocantur                 - 1
21  . ad                       - 2
22  . atque                    - 2
23  . potestate                - 1
24  . igitur                   - 2
25  . at                       - 1
26  . beatum                   - 2
27  . reges                    - 1
28  . ipsum                    - 2
29  . iustus                   - 1
30  . putandum                 - 1
31  . cum                      - 3
32  . elit                     - 1
33  . hoc                      - 2
34  . instituta                - 1
35  . atqui                    - 1
36  . vocem                    - 1
37  . cur                      - 1
38  . loqui                    - 1
39  . regem                    - 1
40  . prioris                  - 1
41  . paulum                   - 1
42  . proverbia                - 1
43  . se                       - 2
44  . quidem                   - 1
45  . sed                      - 5
46  . bestiarum                - 1
47  . iam                      - 1
48  . si                       - 2
49  . nostros                  - 1
50  . sapientis                - 1
51  . capienda                 - 1
52  . audio                    - 1
53  . vestra                   - 1
54  . fomentis                 - 2
55  . a                        - 3
56  . fidibus                  - 1
57  . metuere                  - 1
58  . virtutem                 - 1
59  . archytam                 - 1
60  . videantur                - 1
61  . ullo                     - 1
62  . l                        - 1
63  . accepimus                - 1
64  . constructio              - 1
65  . q                        - 1
```



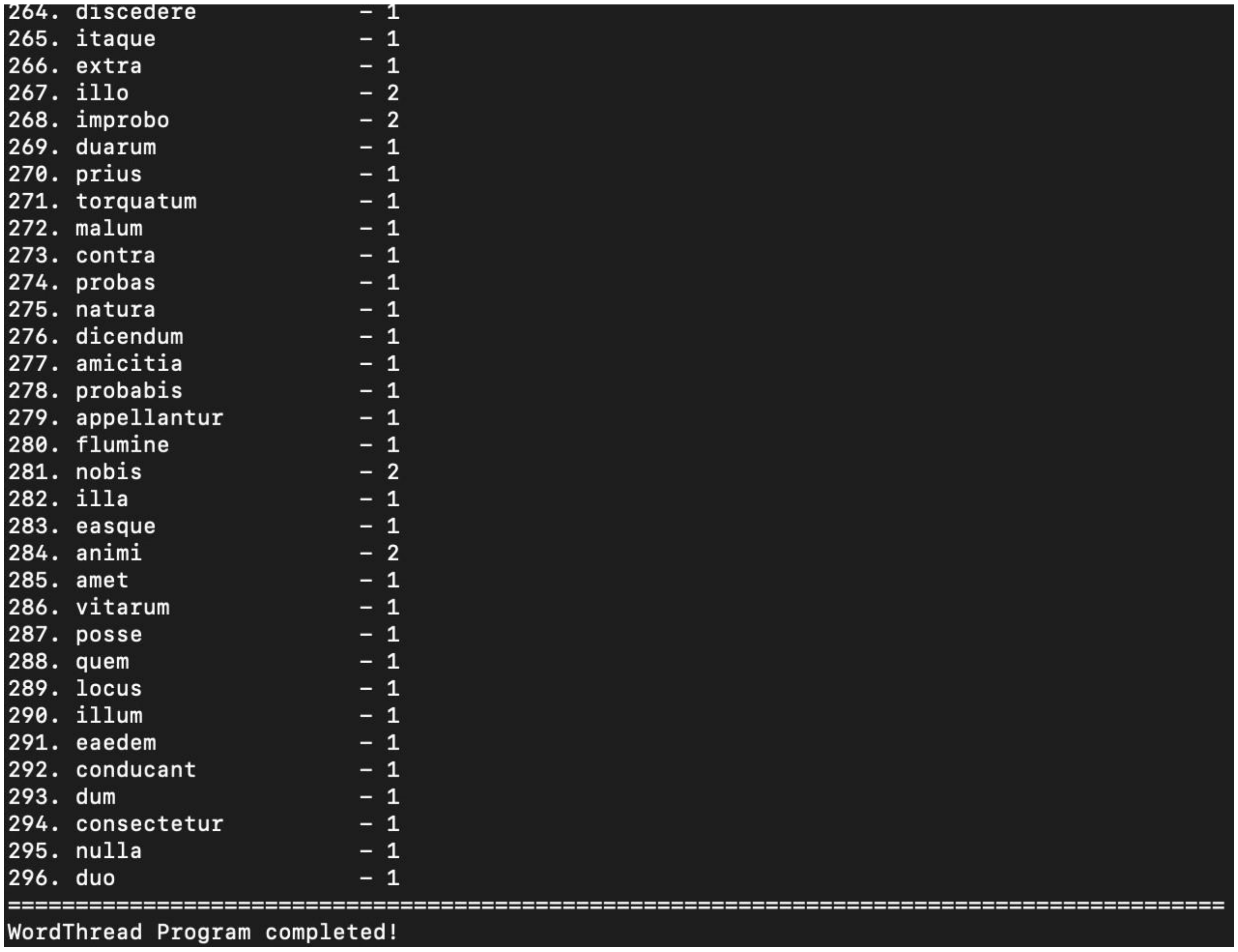
```
66 . te                    - 2
67 . intellegit            - 1
68 . optimus               - 1
69 . maneant               - 1
70 . illae                 - 1
71 . fieri                 - 1
72 . efficiuntur           - 1
73 . ergo                  - 2
74 . quis                  - 1
75 . fortitudinis          - 2
76 . fallaciloquae         - 1
77 . iudicium              - 1
78 . dicere                - 1
79 . de                    - 5
80 . captum                - 1
81 . die                   - 1
82 . tu                    - 1
83 . etiam                 - 4
84 . nimis                 - 1
85 . dicat                 - 1
86 . inquam                - 1
87 . invectio              - 1
88 . ingeniosi             - 1
89 . doctissimos           - 1
90 . destiterit            - 1
91 . malitiae              - 1
92 . vivi                  - 1
93 . post                  - 1
94 . ita                   - 3
95 . adduceret             - 1
96 . dicam                 - 1
97 . ea                    - 6
98 . ut                    - 6
99 . gaudio                - 1
100. res                   - 1
101. habeatur              - 1
102. videamus              - 1
103. quorum                - 1
104. sint                  - 5
105. duas                  - 1
106. es                    - 1
107. malis                 - 1
108. et                    - 3
109. oblitus               - 1
110. relinquet             - 1
111. ex                    - 1
112. effectum              - 1
113. sironem               - 1
114. solet                 - 2
115. erit                  - 1
116. difficultate          - 1
117. sine                  - 2
118. alio                  - 1
119. optimos               - 1
120. confidam              - 1
121. dissimile             - 1
122. festo                 - 1
123. metuit                - 1
124. magnitudinis          - 2
125. qui                   - 3
126. quod                  - 6
127. non                   - 8
128. noli                  - 1
129. docilitas             - 1
130. tiberina              - 1
131. speculis              - 1
132. tamen                 - 3
```



```
133. dicta          - 1
134. poterit        - 1
135. restincta      - 1
136. generis        - 1
137. sit            - 3
138. eodem          - 1
139. principia      - 1
140. virtutis       - 2
141. erunt          - 1
142. constanter     - 1
143. indicant       - 1
144. confirmandus   - 1
145. quibus         - 3
146. dolore         - 1
147. viros          - 1
148. loquare        - 1
149. philodemum     - 1
150. quisque        - 1
151. uno            - 1
152. multis         - 1
153. homines        - 1
154. quicquam       - 1
155. persem         - 1
156. quanto         - 1
157. rerum          - 1
158. memoria        - 1
159. multa          - 1
160. veriora        - 1
161. dedocendi      - 1
162. dogmata        - 1
163. fortemne       - 1
164. rebus          - 1
165. nam            - 2
166. gloriose       - 1
167. vero           - 1
168. fere           - 1
169. numquam        - 1
170. potius         - 1
171. callido        - 1
172. in             - 6
173. amittere       - 1
174. totum          - 1
175. inter          - 1
176. omnia          - 1
177. sibi           - 1
178. iste           - 1
179. tuique         - 1
180. descensio      - 1
181. bonis          - 1
182. bonorum        - 1
183. familiares     - 1
184. consideret     - 1
185. interrete      - 1
186. philosophi     - 1
187. natae          - 1
188. habent         - 1
189. artis          - 1
190. coercendi      - 1
191. probaturum     - 1
192. eae            - 1
193. credo          - 1
194. tibi           - 2
195. tanto          - 1
196. ait            - 1
197. scis           - 1
198. dolor          - 3
```

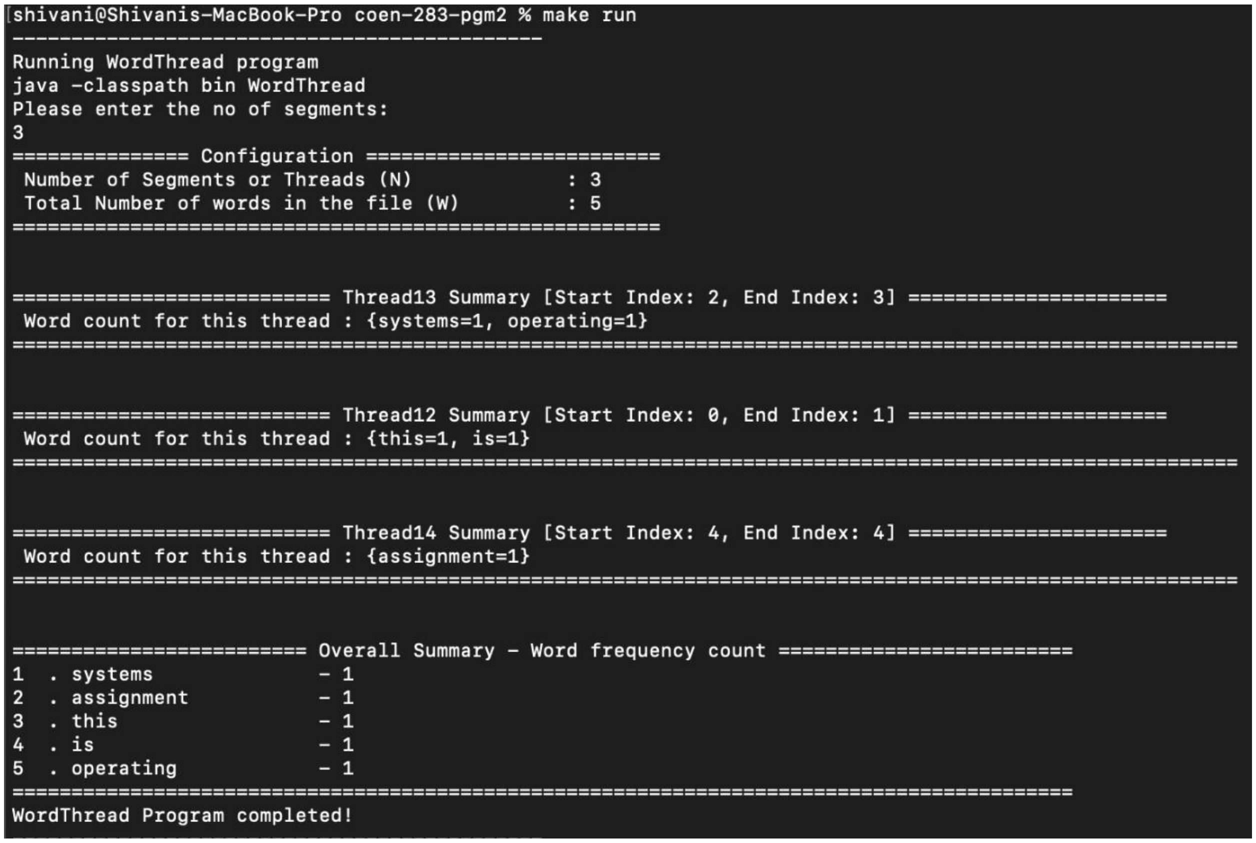


```
199. possunt              - 1
200. liceat               - 2
201. pueri                - 1
202. prorsus              - 1
203. aliquid              - 1
204. omnisque             - 1
205. turpe                - 1
206. nisi                 - 1
207. certe                - 1
208. quam                 - 3
209. haec                 - 1
210. illis                - 1
211. contingit            - 1
212. eundem               - 1
213. puto                 - 1
214. possumus             - 1
215. attinet              - 1
216. quae                 - 8
217. siti                 - 1
218. accius               - 1
219. genera               - 1
220. inquit               - 2
221. equidem              - 1
222. adipiscing           - 1
223. est                  - 10
224. pravae               - 1
225. nostrum              - 2
226. appetitum            - 1
227. inflammat            - 1
228. nec                  - 1
229. nunc                 - 2
230. magis                - 1
231. nomine               - 1
232. summa                - 1
233. ingenii              - 1
234. neque                - 1
235. cernitur             - 1
236. me                   - 1
237. dicis                - 1
238. aeque                - 1
239. dicit                - 2
240. dico                 - 3
241. sunt                 - 3
242. modo                 - 3
243. tam                  - 1
244. enim                 - 4
245. ratione              - 1
246. depravatae           - 1
247. portenta             - 1
248. epicure              - 1
249. affectus             - 2
250. cohaerere            - 1
251. eademne              - 1
252. tarentum             - 1
253. esse                 - 7
254. scilicet             - 1
255. affecit              - 1
256. quaerimus            - 1
257. causa                - 1
258. voluptates           - 1
259. qualis               - 1
260. incontentae          - 1
261. nihil                - 3
262. earum                - 1
263. omnibus              - 1
```



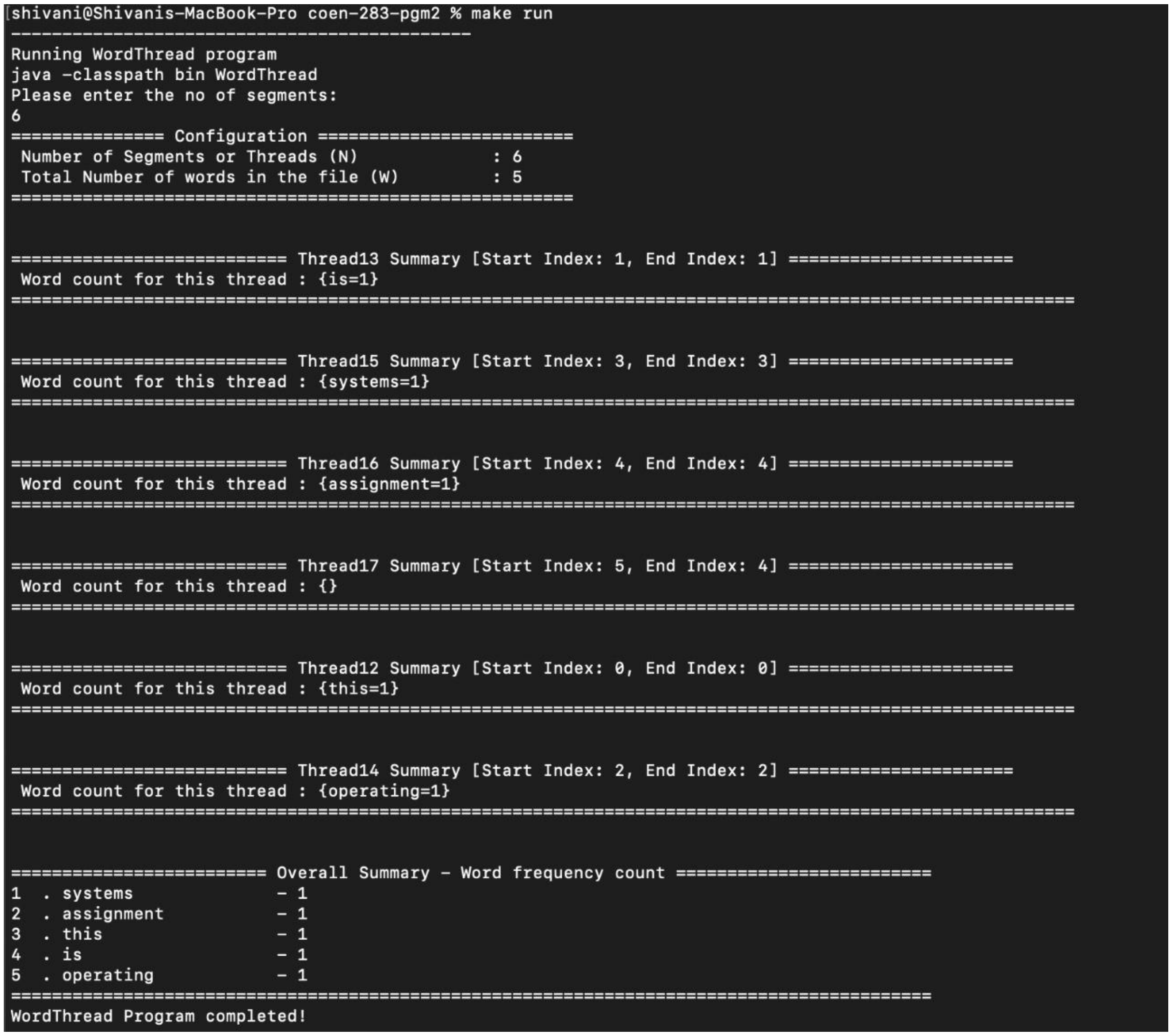
```
264. discedere          - 1
265. itaque             - 1
266. extra              - 1
267. illo               - 2
268. improbo            - 2
269. duarum             - 1
270. prius              - 1
271. torquatum          - 1
272. malum              - 1
273. contra             - 1
274. probas             - 1
275. natura             - 1
276. dicendum           - 1
277. amicitia           - 1
278. probabis           - 1
279. appellantur        - 1
280. flumine            - 1
281. nobis              - 2
282. illa               - 1
283. easque             - 1
284. animi              - 2
285. amet               - 1
286. vitarum            - 1
287. posse              - 1
288. quem               - 1
289. locus              - 1
290. illum              - 1
291. eaedem             - 1
292. conducant          - 1
293. dum                - 1
294. consectetur        - 1
295. nulla              - 1
296. duo                - 1
==========================================================================================
WordThread Program completed!
```

If the N value is not a multiple of W, then approximately fair segments are created using the dynamic segment size algorithm as mentioned in subsection 2.3. In the below screenshot, the input file has 5 words, and the user has input N value to be 3. Using the dynamic segment size calculation algorithm, the first two threads are assigned 2 words each, whereas the last thread has only one word to process.



```
shivani@Shivanis-MacBook-Pro coen-283-pgm2 % make run
----------------------------------------------
Running WordThread program
java -classpath bin WordThread
Please enter the no of segments:
3
=============== Configuration ========================
 Number of Segments or Threads (N)            : 3
 Total Number of words in the file (W)        : 5
======================================================


========================== Thread13 Summary [Start Index: 2, End Index: 3] ======================
 Word count for this thread : {systems=1, operating=1}
==============================================================================================


========================== Thread12 Summary [Start Index: 0, End Index: 1] ======================
 Word count for this thread : {this=1, is=1}
==============================================================================================


========================== Thread14 Summary [Start Index: 4, End Index: 4] ======================
 Word count for this thread : {assignment=1}
==============================================================================================


======================== Overall Summary - Word frequency count ========================
1  . systems            - 1
2  . assignment         - 1
3  . this               - 1
4  . is                 - 1
5  . operating          - 1
========================================================================================
WordThread Program completed!
```

If the N value is greater than W, then some of the threads do not have any workload. In the below example, the input again has 5 words. User input a value of N to be 6. Hence, there will be one thread remains idle with no workload. A future scope for improvement would be to not spawn the thread at all, and to reduce the thread creation, management, and termination overheads.



```
shivani@Shivanis-MacBook-Pro coen-283-pgm2 % make run
---------------------------------------------
Running WordThread program
java -classpath bin WordThread
Please enter the no of segments:
6
=============== Configuration ========================
 Number of Segments or Threads (N)             : 6
 Total Number of words in the file (W)         : 5
=======================================================


=========================== Thread13 Summary [Start Index: 1, End Index: 1] ======================
 Word count for this thread : {is=1}
============================================================================================


=========================== Thread15 Summary [Start Index: 3, End Index: 3] ======================
 Word count for this thread : {systems=1}
============================================================================================


=========================== Thread16 Summary [Start Index: 4, End Index: 4] ======================
 Word count for this thread : {assignment=1}
============================================================================================


=========================== Thread17 Summary [Start Index: 5, End Index: 4] ======================
 Word count for this thread : {}
============================================================================================


=========================== Thread12 Summary [Start Index: 0, End Index: 0] ======================
 Word count for this thread : {this=1}
============================================================================================


=========================== Thread14 Summary [Start Index: 2, End Index: 2] ======================
 Word count for this thread : {operating=1}
============================================================================================


========================= Overall Summary - Word frequency count ========================
1   . systems               - 1
2   . assignment            - 1
3   . this                  - 1
4   . is                    - 1
5   . operating             - 1
============================================================================================
WordThread Program completed!
```

7. Scope for future work

The program can be enhanced to add the below capabilities.

- Ability to accept a filename as a user input rather than having a limitation of “input.txt”
- Ability to not spawn threads for cases where N > W.
- Performance analysis to measure the speedup obtained as the N number increases.

8. Conclusion

This exercise gives a good insight into multi-threading, and the APIs required to create, maintain, and terminate threads.