```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Perigrafi
FUNCTION main
A.1 Declare variables
A.2 Request the first line of words and allocate necessary space
A.3 Ask the user for their choice
A.4 Perform the corresponding action
A.5 Terminate the program

FUNCTION allocateMemory
B.1 Declare variables
B.2 Get the number of words
B.3 Assign the corresponding space

FUNCTION getWords
C.1 Declare variables
C.2 Get the sequence of words

FUNCTION printWords
D.1 Declare variables
D.2 Display the sequence of words

FUNCTION getWordsStats
E.1 Declare variables
E.2 Initialize variable values
E.3 Compare words
E.4 Display the longest, shortest, and average word

FUNCTION getCharachterStats
F.1 Declare variables
F.2 Get a character
F.3 Initialize variable values
F.4 Find the number of occurrences
F.5 Find maximum and minimum occurrences and store the word where they occur
F.6 Find the average occurrences
F.7 Display maximum, minimum, average occurrences, and the words with maximum and minimum occurrences

FUNCTION freeMemory
G.1 Declare variables
G.2 Free up space
*/

//Set prototypes
void allocateMemory(int *size, char *** words);

void getWords(int size, char ** words);

void printWords(int size, char ** words);

void getWordStats(int size, char ** words);

void getCharacterStats(int size, char ** words);

void freeMemory(int size, char ** words);

//Set enumaration
enum menu {
    TERMINATE,
    GET_WORDS,
    SHOW_WORDS,
    WORD_STATS,
    CHARACTER_STATS
};

int main() {
    //Set variables
    int word, size;
    char ** words;
    int choice = -1;

```

```
74:     //Get the first word
75:     allocateMemory(&size, &words);
76:     getWords(size, words);
77:
78:     system("cls");
79:
80:     while(choice != TERMINATE) {
81:         //Ask for choice
82:         printf("Choose action: \nTERMINATE: 0\nENTER WORDS: 1\nPRINT WORDS: 2\nGET WORDS STATS: 3\nGET CHARACTER STATS
83:         scanf("%d", &choice);
84:
85:         //Clear console screeen
86:         system("cls");
87:
88:         switch(choice) {
89:             case TERMINATE:
90:                 freeMemory(size, words);
91:                 printf("Exiting program...!");
92:                 return 0;
93:                 break;
94:             case GET_WORDS:
95:                 //Free previous allocated memory
96:                 freeMemory(size, words);
97:                 //Get size and give memory
98:                 allocateMemory(&size, &words);
99:                 //Get words function
100:                getWords(size, words);
101:                //System to clear screen
102:                system("pause");
103:                system("cls");
104:                break;
105:            case SHOW_WORDS:
106:                //Print the word list
107:                printWords(size, words);
108:                //System to clear screen
109:                system("pause");
110:                system("cls");
111:                break;
112:            case WORD_STATS:
113:                //Get the words status
114:                getWordStats(size, words);
115:                //System to clear screen
116:                system("pause");
117:                system("cls");
118:                break;
119:            case CHARACTER_STATS:
120:                //Get character status
121:                getCharacterStats(size, words);
122:                //System to clear screen
123:                system("pause");
124:                system("cls");
125:                break;
126:            default:
127:                break;
128:        }
129:    }
130:
131:    return 0;
132: }
133: void allocateMemory(int *size, char *** words) {
134:    //Set variables
135:    int i;
136:
137:    //Repeat till under 30 words
138:    do {
139:        //Ask number of words
140:        printf("Enter how many words you want(up to 30): ");
141:        scanf("%d", size);
142:    }
143:    while(*size > 30 || *size == 0);
144:
145:    //Check to see the size
146:    if (*size > 30) {
```

```
147:         printf("Word limit error!\nExiting program.");
148:         exit(0);
149:     }
150:
151:     //Allocate the memory u need to the arrays
152:     *words = (char **)malloc(*size * sizeof(char*));
153:     for (i = 0; i < *size; i++) {
154:         (*words)[i] = (char*)malloc(30 * sizeof(char)); //Using 30 for word length
155:     }
156: }
157:
158: void getWords(int size, char ** words) {
159:     //Set variables
160:     int word;
161:
162:     for (word = 0; word < size; word++) {
163:         //Ask word
164:         printf("\nEnter word: ");
165:         //Get word and start storing it from the end of the previous word
166:         scanf("%s", words[word]);
167:     }
168: }
169:
170: void printWords(int size, char ** words) {
171:     //Set variables
172:     int word;
173:     //Print the words
174:     for (word = 0; word < size; word++) {
175:         printf("%s ", words[word]);
176:     }
177:     //Start new line
178:     printf("\n");
179: }
180:
181: void getWordStats(int size, char ** words) {
182:     //Set variables;
183:     int word, min, max, wordLen;
184:     float average;
185:
186:     //Set starting values
187:     max = strlen(words[0]);
188:     min = strlen(words[0]);
189:     average = strlen(words[0]);
190:
191:     //Itirate through each words to get size
192:     for (word = 1; word < size; word++) {
193:         //Get the word's length
194:         wordLen = strlen(words[word]);
195:         //Compare to current max and min
196:         if (wordLen > max) max = wordLen;
197:         if (wordLen < min) min = wordLen;
198:         //Add to average
199:         average += wordLen;
200:         }
201:     //Divide by size
202:     average = average / size;
203:     //Print status
204:     printf("Max Word Length - %d\nMin Word Length- %d\nAverage Word Length - %.2f\n", max, min, average);
205: }
206:
207: void getCharacterStats(int size, char ** words) {
208:     //Set variables
209:     int word, letter, charCount, charMax, charMin, charTotal, maxWord, minWord;
210:     float charAverage;
211:     char character;
212:
213:     //Charachter check
214:     printf("Enter charachter: ");
215:     scanf(" %c", &character);
216:
217:     //Set starting values
218:     charMax = 0;
219:     charMin = 900; //30 words * 30 charachters maximum appearances
```

```
220:     charAverage = 0;
221:     charTotal = 0;
222:
223:     //Itirate through words
224:     for (word = 0; word < size; word++) {
225:         //Itirate through letters
226:         charCount  = 0;
227:         for (letter = 0; letter < strlen(words[word]);letter++) {
228:             if (words[word][letter] == character) {
229:                 charCount++;
230:             }
231:         }
232:         //Add the current character count to the total
233:         charTotal += charCount;
234:         //Check if currentCount is bigger than max
235:         if (charMax < charCount) {
236:             charMax = charCount;
237:             //Store the current max charachter word position
238:             maxWord = word;
239:         }
240:         if (charMin > charCount) {
241:             charMin = charCount;
242:             //Store the current min charachter word position
243:             minWord = word;
244:         }
245:     }
246:     //Get average
247:     charAverage = (float)charTotal / (float)size;
248:     //Print Stats
249:     printf("TotalChar - %d\nAverageChar - %.2f\nMaxChar - %d\nMinChar - %d\nMaxWord - %s\nMinWord - %s\n", charTotal,
250: }
251:
252: void freeMemory(int size, char ** words) {
253:     int word;
254:
255:     //Free the allocated memory once program is done
256:     for (word = 0; word < size; word++) {
257:         free(words[word]);
258:     }
259:     free(words);
260: }
```