**FORMAN CHRISTIAN COLLEGE (A CHARTERED UNIVERSITY)**

****

**Course Title and Course Code**

**Session**

**Lab Number 1**

**Title of Lab/Assignment**

**Kiran Qaiser 251683919**

**Hafsah Shahbaz 251684784**

**You should attach the lab / assignment handout as second page of this report.**

**From third page onwards following headings should be included:**

* **Introduction**
  + **Should carry information of all major library functions used in your program**
* **Your logic / algorithm in simple English. Bullet points are appreciated.**
* **Your code**
* **Screen shots of at least three outputs of your code with appropriate inputs followed by your comments about the input and output relationship.**
* **References**

**Introduction**

The given C program reads and analyses a text file and displays line numbers and character counts. It makes use of standard C library methods for handling files and manipulating characters.

Principal Uses of the Library:

fopen(): Utilizing the command-line parameter, it opens a file for viewing.

perror(): Produces informative error messages in the event that the file opening process fails.

fgetc(): Reads characters from the file that has been opened.

printf(): Output is prepared and printed to the console.

fclose():The file is closed using the function.

**Logic:**

1. Command-line Argument Check:

Check that the appropriate number of command-line arguments (filename only) is provided.

If not, display an error message indicating the correct usage and exit the program.

1. File Opening:

Open the given file in read-only mode.

If the attempt to open the file is unsuccessful, output an error message (such as "file not

found") and end the application.

1. Initialization:

* char\_count : To track the number of characters in a line, initialize the following variables.
* line\_count: To preserve the current line count.
* is\_empty\_line: Check if the current line is empty.

1. **Reading File Character by Character:**

**Continue iterating over every character in the file until the end (EOF).**

**For each character:**

* **The character should be printed on the console.**
* **char\_count is increased while keeping out tabs, spaces, and newline characters.**
* **Depending on the kind of character, update is\_empty\_line.**

1. Processing Newline Character:

When a newline character appears, output the line number and the total number of

characters. In that line if is\_empty\_line is false.

Print a newline character if is\_empty\_line is true.

1. Finalization:

Following the loop: Verify that the final line is not empty.

If this is the case, output the final line's total character count.

Close the file to release system resources.

1. Error Handling:

Handle errors correctly at every stage of the process:

When a file opens unsuccessfully or with incorrect command-line options, provide informative

error messages.

To ensure a smooth shutdown, if serious problems arise when reading a file, exit the

program.

**CODE:**

#include <stdio.h>

#include <stdlib.h>

int main(int argc, char \*argv[]) {

if (argc != 2) {

printf("Usage: %s <filename>\n", argv[0]);

return 1;

}

FILE \*file = fopen(argv[1], "r");

if (file == NULL) {

perror("Error opening file");

return 1;

}

char ch;

int line\_count = 1;

int char\_count = 0;

int is\_empty\_line = 1;

printf("%d: ", line\_count);

while ((ch = fgetc(file)) != EOF) {

if (ch == '\n') {

if (!is\_empty\_line) {

printf(" --- %d\n", char\_count);

char\_count = 0;

} else {

printf("\n");

}

line\_count++;

printf("%d: ", line\_count);

is\_empty\_line = 1;

} else {

printf("%c", ch);

char\_count++;

if (ch != ' ' && ch != '\t' && ch != '\n') {

is\_empty\_line = 0;

}

}

}

if (!is\_empty\_line) {

printf(" --- %d\n", char\_count);}

fclose(file);

return 0;

}

**Comments:**

The program efficiently reads the input file, counting characters precisely and arranging the data in a

structured manner.

The output has an easy-to-read structure and presents the contents of the input file line by line.

