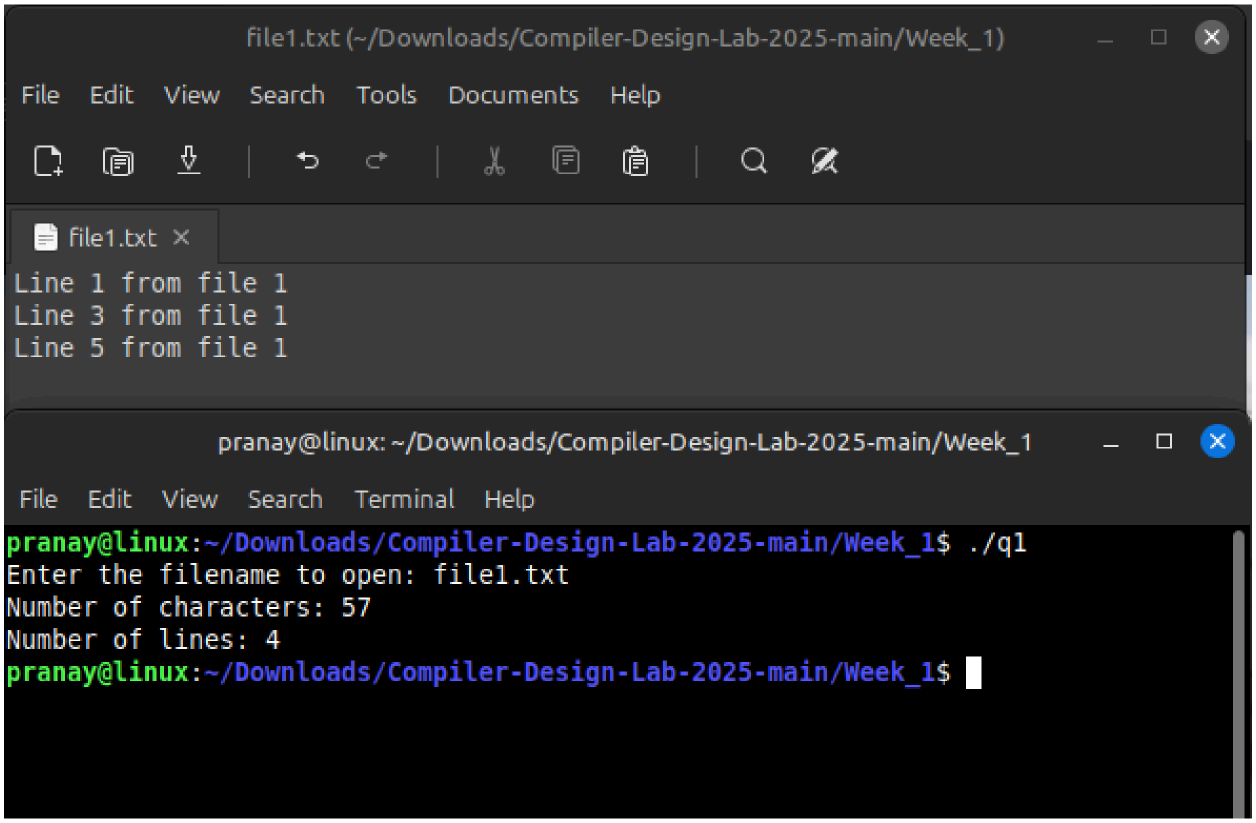
CD LAB - WEEK 1

Pranay Goel
220905524
CSE B

Write a ‘C’ program to

1. count the number of lines and characters in a file.

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

int main() {
    FILE *file;
    char filename[100];
    char c;
    int charCount = 0, lineCount = 0;

    printf("Enter the filename to open: ");
    scanf("%s", filename);

    file = fopen(filename, "r");
    if (file == NULL) {
        printf("Cannot open file %s for reading\n", filename);
        exit(1);
    }

    while ((c = fgetc(file)) != EOF) {
        charCount++;
        if (c == '\n') {
            lineCount++;
        }
    }

    // For when the file doesn't end with a newline
    if (charCount > 0 && c != '\n') {
        lineCount++;
    }

    printf("Number of characters: %d\n", charCount);
    printf("Number of lines: %d\n", lineCount);

    fclose(file);

    return 0;
}
```


file1.txt (~/Downloads/Compiler-Design-Lab-2025-main/Week_1)
File Edit View Search Tools Documents Help
file1.txt

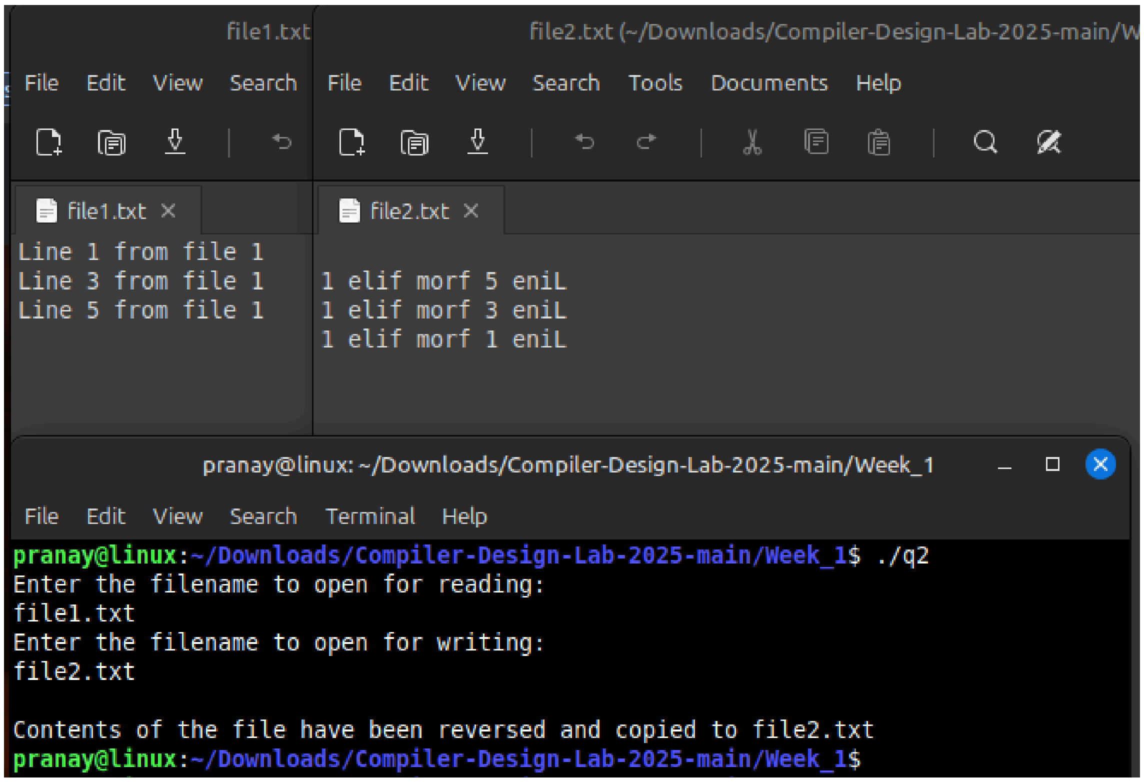
```
Line 1 from file 1
Line 3 from file 1
Line 5 from file 1
```

pranay@linux: ~/Downloads/Compiler-Design-Lab-2025-main/Week_1
File Edit View Search Terminal Help

```
pranay@linux:~/Downloads/Compiler-Design-Lab-2025-main/Week_1$ ./q1
Enter the filename to open: file1.txt
Number of characters: 57
Number of lines: 4
pranay@linux:~/Downloads/Compiler-Design-Lab-2025-main/Week_1$ 
```

2. reverse the file contents and store in another file. Also display the size of the file using the file handling function.

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

int main()
{
    FILE *fptr1, *fptr2;
    char sourceFilename[100], destFilename[100];
    long fileSize;
    char c;

    printf("Enter the filename to open for reading: \n");
    scanf("%s", sourceFilename);
    fptr1 = fopen(sourceFilename, "r");
    if (fptr1 == NULL)
    {
        printf("Cannot open file %s for reading\n", sourceFilename);
        exit(1);
    }
    printf("Enter the filename to open for writing: \n");
    scanf("%s", destFilename);
    fptr2 = fopen(destFilename, "w");
    if (fptr2 == NULL)
    {
        printf("Cannot open file %s for writing\n", destFilename);
        fclose(fptr1);
        exit(1);
    }

    fseek(fptr1, 0, SEEK_END);
    fileSize = ftell(fptr1);

    while (fileSize > 0)
    {
        fseek(fptr1, --fileSize, SEEK_SET);
        c = fgetc(fptr1);
        fputc(c, fptr2);
    }

    printf("\nContents of the file have been reversed and copied to %s\n", destFilename);
    fclose(fptr1);
    fclose(fptr2);
}
```



file1.txt

```
Line 1 from file 1
Line 3 from file 1
Line 5 from file 1
```

file2.txt (~/Downloads/Compiler-Design-Lab-2025-main/W

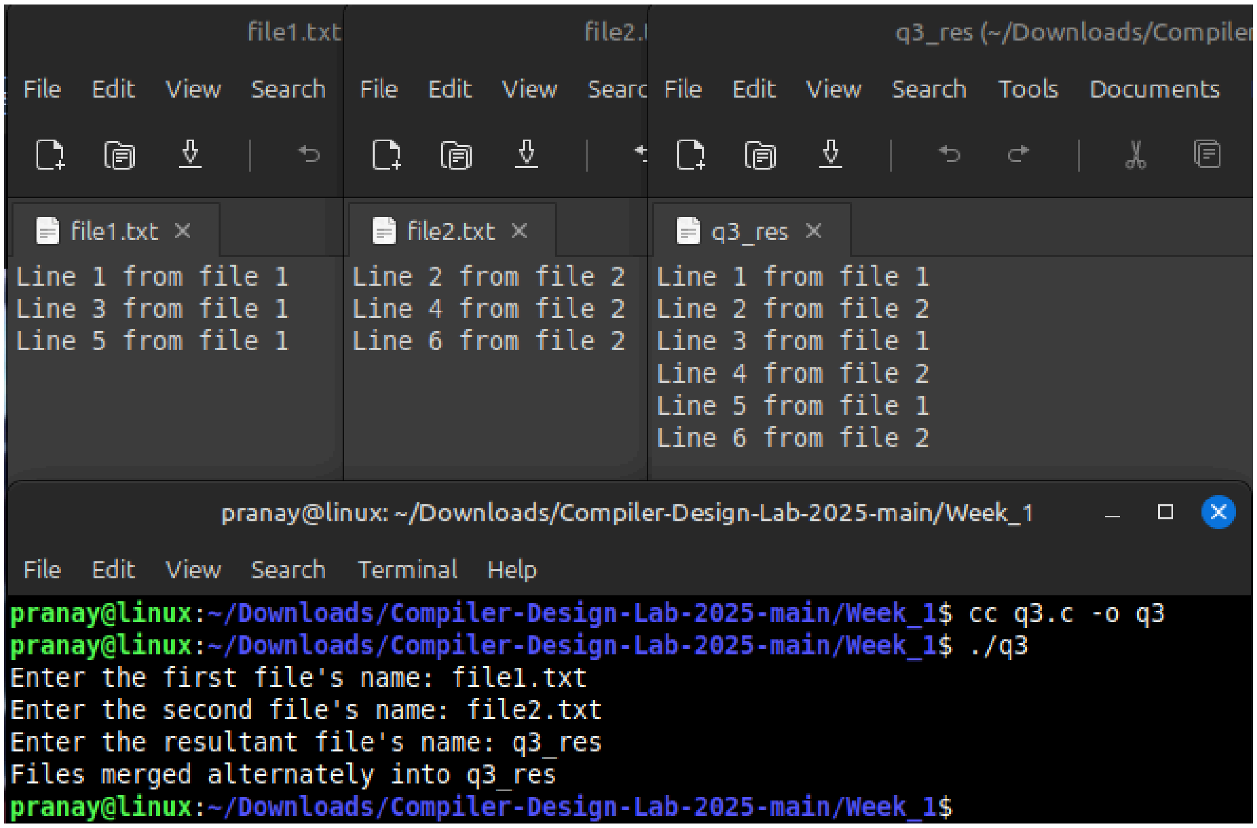
```

1 elif morf 5 eniL
1 elif morf 3 eniL
1 elif morf 1 eniL
```

pranay@linux: ~/Downloads/Compiler-Design-Lab-2025-main/Week_1

```
pranay@linux:~/Downloads/Compiler-Design-Lab-2025-main/Week_1$ ./q2
Enter the filename to open for reading:
file1.txt
Enter the filename to open for writing:
file2.txt

Contents of the file have been reversed and copied to file2.txt
pranay@linux:~/Downloads/Compiler-Design-Lab-2025-main/Week_1$
```

3. That merges lines alternatively from 2 files and stores it in a resultant file.

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

int main() {
    FILE *f1ptr, *f2ptr, *resultptr;
    char f1[100], f2[100], result[100];
    int c;

    printf("Enter the first file's name: ");
    scanf("%s", f1);
    f1ptr = fopen(f1, "r");
    if (f1ptr == NULL) {
        printf("Cannot open file %s for reading\n", f1);
        exit(1);
    }

    printf("Enter the second file's name: ");
    scanf("%s", f2);
    f2ptr = fopen(f2, "r");
    if (f2ptr == NULL) {
        printf("Cannot open file %s for reading\n", f2);
        exit(1);
    }

    printf("Enter the resultant file's name: ");
    scanf("%s", result);
    resultptr = fopen(result, "a");
    if (resultptr == NULL) {
        printf("Cannot open file %s for writing\n", result);
        exit(1);
    }

    while (1) {
        while ((c = getc(f1ptr)) != EOF) {
            putc(c, resultptr);
            if (c == '\n') break;
        }

        while ((c = getc(f2ptr)) != EOF) {
            putc(c, resultptr);
            if (c == '\n') break;
        }
        if (feof(f1ptr) && feof(f2ptr)) break;
    }

    printf("Files merged alternately into %s\n", result);
    fclose(f1ptr);
    fclose(f2ptr);
    fclose(resultptr);

    return 0;
}
```


file1.txt
file2.
q3_res (~/Downloads/Compiler
File Edit View Search
File Edit View Searc
File Edit View Search Tools Documents
file1.txt
file2.txt
q3_res
pranay@linux: ~/Downloads/Compiler-Design-Lab-2025-main/Week_1
File Edit View Search Terminal Help

```
Line 1 from file 1
Line 3 from file 1
Line 5 from file 1
```

```
Line 2 from file 2
Line 4 from file 2
Line 6 from file 2
```

```
Line 1 from file 1
Line 2 from file 2
Line 3 from file 1
Line 4 from file 2
Line 5 from file 1
Line 6 from file 2
```

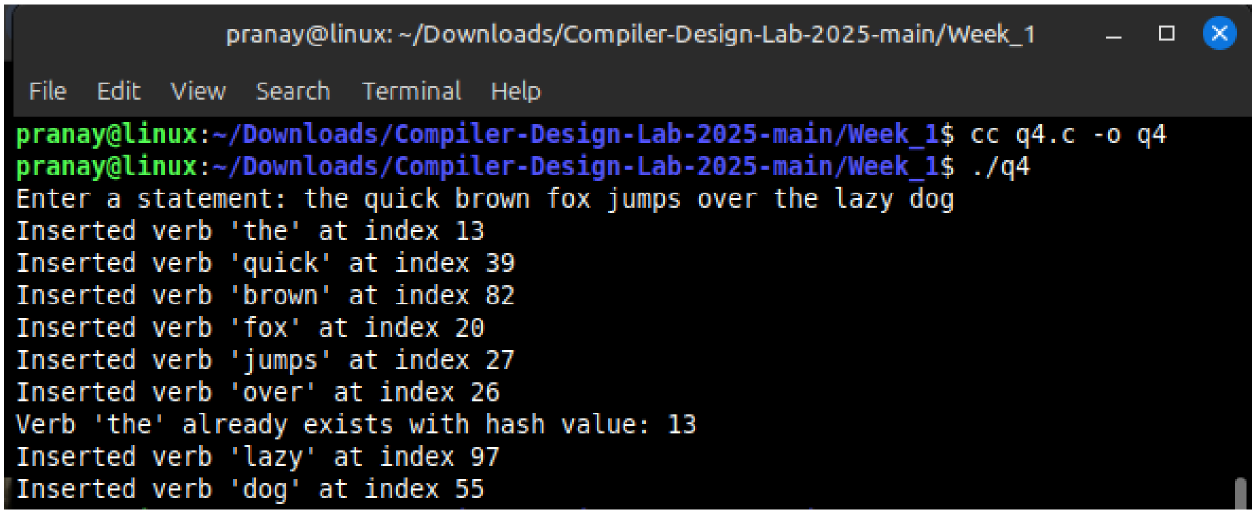
```
pranay@linux:~/Downloads/Compiler-Design-Lab-2025-main/Week_1$ cc q3.c -o q3
pranay@linux:~/Downloads/Compiler-Design-Lab-2025-main/Week_1$ ./q3
Enter the first file's name: file1.txt
Enter the second file's name: file2.txt
Enter the resultant file's name: q3_res
Files merged alternately into q3_res
pranay@linux:~/Downloads/Compiler-Design-Lab-2025-main/Week_1$
```

4. That accepts an input statement, identifies the verbs present in them and performs the following functions:

a. INSERT: Used to insert a verb into the hash table.
   Syntax: insert (char *str)

b. SEARCH: Used to search for a key(verb) in the hash table. This function is called by the INSERT function. If the symbol table already contains an entry for the verb to be inserted, then it returns the hash value of the respective verb. If a verb is not found, the function returns -1.
   Syntax: int search (key)

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

#define TABLE_SIZE 8
#define MAX_VERB_LENGTH 20

// Verb structure for hash table
typedef struct {
    char verb[MAX_VERB_LENGTH];
    int count;
} VerbEntry;

// Global hash table and verb array
VerbEntry verbTable[TABLE_SIZE];
char* commonVerbs[] = {"is", "are", "was", "were", "have", "has", "do", "does"};

// Initialize hash table
void initHashTable() {
    for (int i = 0; i < TABLE_SIZE; i++) {
        strcpy(verbTable[i].verb, "");
        verbTable[i].count = 0;
    }
}

// Hash function - simple modulo
int hashFunction(char* verb) {
    int total = 0;
    for (int i = 0; verb[i] != '\0'; i++) {
        total += verb[i];
    }
    return total % TABLE_SIZE;
}

// Search function
int search(char* verb) {
    int index = hashFunction(verb);

    // Linear probing
    for (int i = 0; i < TABLE_SIZE; i++) {
        int current = (index + i) % TABLE_SIZE;

        if (strcmp(verbTable[current].verb, verb) == 0) {
            return current;  // Verb found
        }

        if (strcmp(verbTable[current].verb, "") == 0) {
            return -1;  // Empty slot, verb not found
        }
    }

    return -1;
}

// Insert function
void insert(char* verb) {
    // Check if verb already exists
    int existingIndex = search(verb);
    if (existingIndex != -1) {
        verbTable[existingIndex].count++;
        return;
    }

    // Find insertion point
    int index = hashFunction(verb);
    for (int i = 0; i < TABLE_SIZE; i++) {
        int current = (index + i) % TABLE_SIZE;

        if (strcmp(verbTable[current].verb, "") == 0) {
            strcpy(verbTable[current].verb, verb);
            verbTable[current].count = 1;
            return;
        }
    }
}

// Read file and process verbs
void processFile(char* filename) {
    FILE* file = fopen(filename, "r");
    if (!file) {
        printf("Cannot open file\n");
        return;
    }

    char currentWord[MAX_VERB_LENGTH] = {0};
    int wordIndex = 0;
    int ch;
    while ((ch = getc(file)) != EOF) {
        // Check if character is alphabetic
        if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z')) {
            currentWord[wordIndex++] = ch;
        } else {
            // End of word
            if (wordIndex > 0) {
                currentWord[wordIndex] = '\0';

                // Check if current word is a common verb
                for (int i = 0; i < 8; i++) {
                    if (strcmp(currentWord, commonVerbs[i]) == 0) {
                        insert(currentWord);
                        break;
                    }
                }

                // Reset word
                wordIndex = 0;
                memset(currentWord, 0, sizeof(currentWord));
            }
        }
    }

    fclose(file);
}

// Print hash table
void printHashTable() {
    printf("Verb Hash Table:\n");
    for (int i = 0; i < TABLE_SIZE; i++) {
        if (strcmp(verbTable[i].verb, "") != 0) {
            printf("%s: %d occurrences\n", verbTable[i].verb, verbTable[i].count);
        }
    }
}

int main() {
    initHashTable();
    processFile("test4");  // Replace with your input file
    printHashTable();

    return 0;
}
```


pranay@linux: ~/Downloads/Compiler-Design-Lab-2025-main/Week_1
File Edit View Search Terminal Help

```
pranay@linux:~/Downloads/Compiler-Design-Lab-2025-main/Week_1$ cc q4.c -o q4
pranay@linux:~/Downloads/Compiler-Design-Lab-2025-main/Week_1$ ./q4
Enter a statement: the quick brown fox jumps over the lazy dog
Inserted verb 'the' at index 13
Inserted verb 'quick' at index 39
Inserted verb 'brown' at index 82
Inserted verb 'fox' at index 20
Inserted verb 'jumps' at index 27
Inserted verb 'over' at index 26
Verb 'the' already exists with hash value: 13
Inserted verb 'lazy' at index 97
Inserted verb 'dog' at index 55
```