CD LAB-1

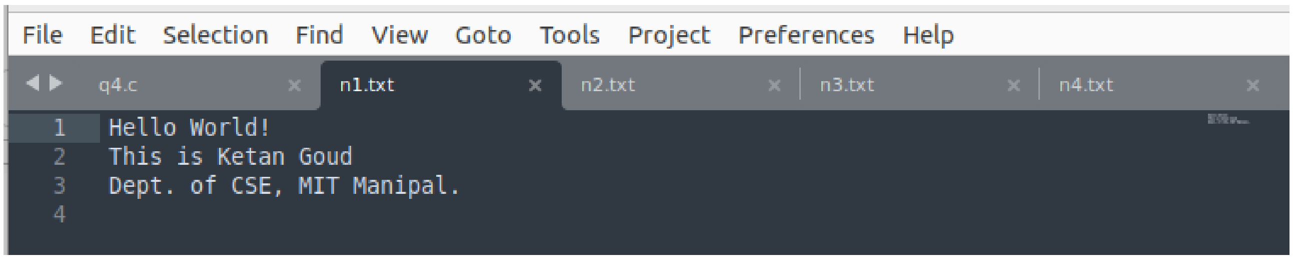
Name: Mogula Ketan Goud
Roll No: 39
Section & Batch: CSE D – D2
Reg No: 220905260

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

```
#include <stdio.h>
#include <stdlib.h> // For exit()
int main()
{
        int lines=0, chars=0;
        FILE *fptr1, *fptr2;
        char filename[100], c;
        printf("Enter the filename to open for reading: \n");
        scanf("%s", filename);
        fptr1 = fopen(filename, "r");
        if (fptr1 == NULL)
        {
                printf("Cannot open file %s \n", filename);
                exit(0);
        }
        c = fgetc(fptr1);
        while (c != EOF)
        {
                c = fgetc(fptr1);
                if(c=='\n')
                        lines++;
                else
                        chars++;
        }
        printf("number of characters are %d\n",chars);
        printf("number of lines are %d\n",lines);
        fclose(fptr1);
        return 0;
}
```

Sample Input and Output:

```
Enter the filename to open for reading:
n1.txt
number of characters are 56
number of lines are 3
```


File Edit Selection Find View Goto Tools Project Preferences Help
q4.c
n1.txt
n2.txt
n3.txt
n4.txt

```
Hello World!
This is Ketan Goud
Dept. of CSE, MIT Manipal.
```

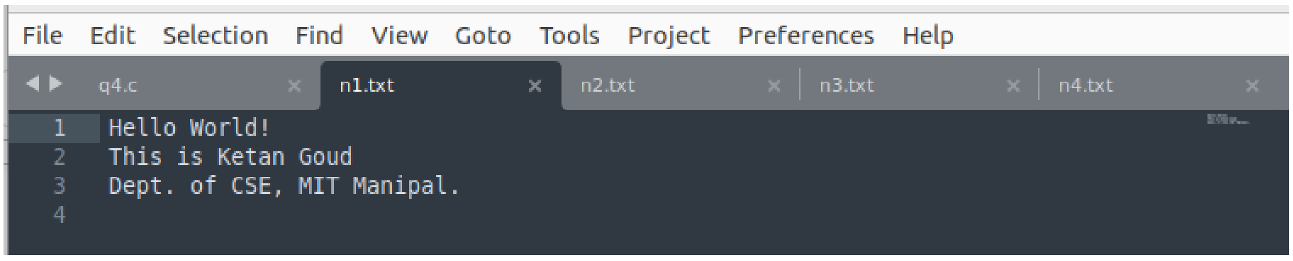
2. To reverse the file contents and store in another file. Also display the size of file using file handling function.

```
#include <stdio.h>
#include <stdlib.h> // For exit()
int main()
{
	int lines=0, chars=0;
	FILE *fptr1, *fptr2;
	char filename[100], c;
	printf("Enter the filename to open for reading: \n");
	scanf("%s", filename);
	fptr1 = fopen(filename, "r");
	if (fptr1 == NULL)
	{
		printf("Cannot open file %s \n", filename);
		exit(0);
	}
	printf("Enter the filename to open for writing: \n");
	scanf("%s", filename);
	fptr2 = fopen(filename, "w+"); // Open another file for writing
	fseek(fptr1,0L,SEEK_END);
	int res=ftell(fptr1);
	printf("Length of file is %d\n",res);
	for(int i=res-1;i>=0;i--)
	{
		fseek(fptr1,i,SEEK_SET);
		c=fgetc(fptr1);
		fputc(c,fptr2);
	}
	fclose(fptr1);
	return 0;
}
```

Sample Input and Output:

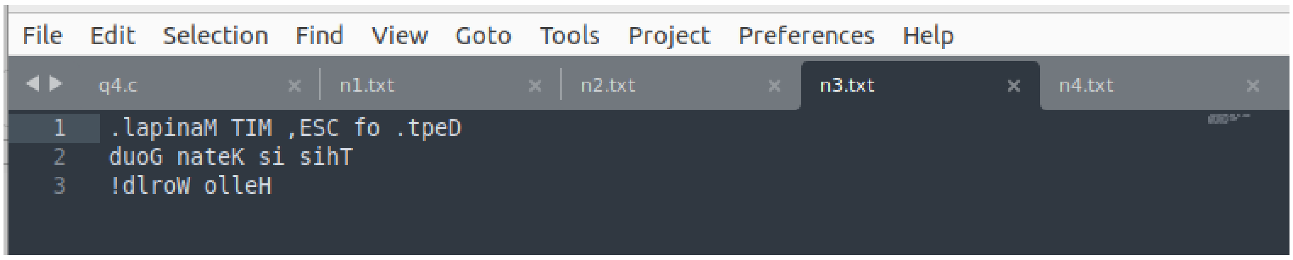
```
Enter the filename to open for reading:
n1.txt
Enter the filename to open for writing:
n3.txt
Length of file is 59
```

n1.txt:



```
Hello World!
This is Ketan Goud
Dept. of CSE, MIT Manipal.
```

n3.txt


File Edit Selection Find View Goto Tools Project Preferences Help
q4.c
n1.txt
n2.txt
n3.txt
n4.txt

```
.lapinaM TIM ,ESC fo .tpeD
duoG nateK si sihT
!dlroW olleH
```

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

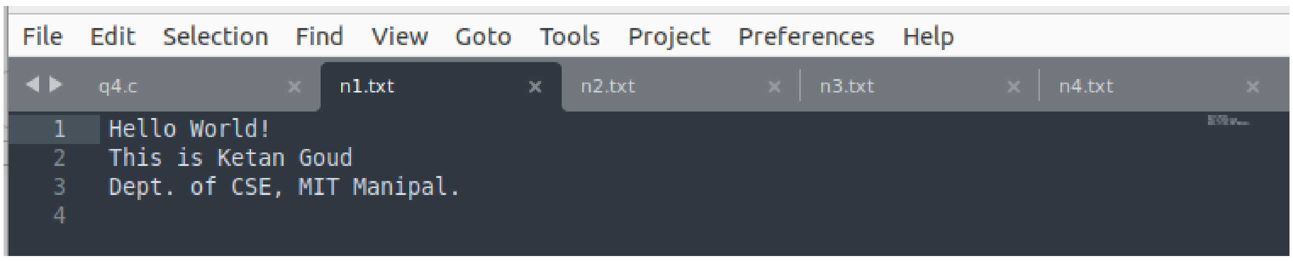
```
#include <stdio.h>
#include <stdlib.h> // For exit()
int main()
{
	int lines=0, chars=0;
	FILE *fptr1, *fptr2, *fptr3;
	char filename[100], c1,c2;
	printf("Enter the filename to open for reading: \n");
	scanf("%s", filename);
	fptr1 = fopen(filename, "r");
	if (fptr1 == NULL)
	{
		printf("Cannot open file %s \n", filename);
		exit(0);
	}
	printf("Enter the filename to open for reading: \n");
	scanf("%s", filename);
	fptr2 = fopen(filename, "r");
	if (fptr2 == NULL)
	{
		printf("Cannot open file %s \n", filename);
		exit(0);
	}
	printf("Enter the filename to open for writing: \n");
	scanf("%s", filename);
	fptr3 = fopen(filename, "w+");
	while(1)
	{
		if(c1!=EOF)
		{
			c1=fgetc(fptr1);
			while(c1!='\n')
			{
				if(c1==EOF)
					break;
				fputc(c1,fptr3);
				c1=fgetc(fptr1);
			}
			if(c1=='\n')
				fputc('\n',fptr3);
```

```
                }
                if(c2!=EOF)
                {
                        c2=fgetc(fptr2);
                        while(c2!='\n')
                        {
                                if(c2==EOF)
                                        break;
                                fputc(c2,fptr3);
                                c2=fgetc(fptr2);
                        }
                        if(c2=='\n')
                                fputc('\n',fptr3);
                }
                if(c1==EOF && c2==EOF)
                        break;
        }
        printf("merged files alternatively\n");
        fclose(fptr1);
        fclose(fptr2);
        fclose(fptr3);
        return 0;
}
```

Sample Input and Output:

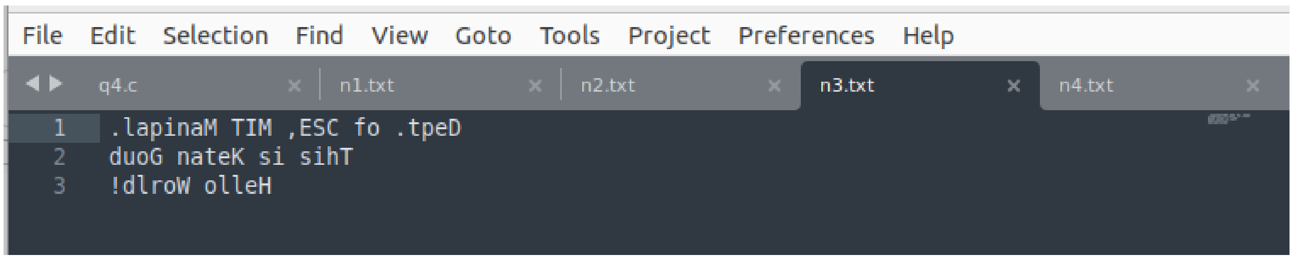
```
Enter the filename to open for reading:
n1.txt
Enter the filename to open for reading:
n3.txt
Enter the filename to open for writing:
n4.txt
merged files alternatively
```

n1.txt:


File Edit Selection Find View Goto Tools Project Preferences Help
q4.c
n1.txt
n2.txt
n3.txt
n4.txt

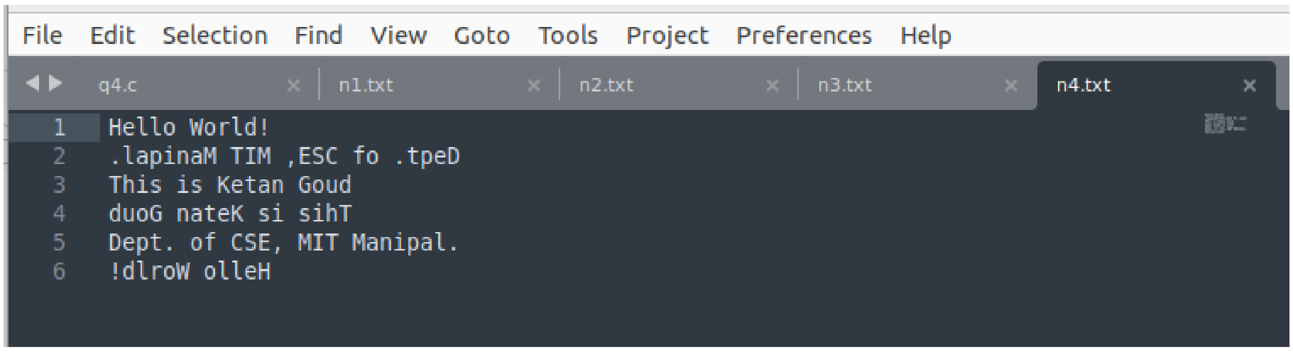
```
Hello World!
This is Ketan Goud
Dept. of CSE, MIT Manipal.

```

n3.txt:


File Edit Selection Find View Goto Tools Project Preferences Help
q4.c
n1.txt
n2.txt
n3.txt
n4.txt

```
.lapinaM TIM ,ESC fo .tpeD
duoG nateK si sihT
!dlroW olleH
```

n4.txt:


File Edit Selection Find View Goto Tools Project Preferences Help
q4.c
n1.txt
n2.txt
n3.txt
n4.txt

```
Hello World!
.lapinaM TIM ,ESC fo .tpeD
This is Ketan Goud
duoG nateK si sihT
Dept. of CSE, MIT Manipal.
!dlroW olleH
```

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 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 <ctype.h>

#define TABLE_SIZE 100

typedef struct
{
   char *verb;
   int used;
} HashEntry;

HashEntry Table[TABLE_SIZE];

unsigned int hash(char *str)
{
   unsigned int hashVal = 0;
   while (*str != '\0')
   {
      hashVal = hashVal*31 + *str++; // hashVal = hashVal*31 + current_char
   }
   return hashVal % TABLE_SIZE;
}

void insert(char *verb)
{
   int index = search(verb);
   if (index != -1)
   {
      printf("Verb '%s' already exists at index %d.\n", verb, index);
```

```
    }
    else
    {
        index = hash(verb);
        while (Table[index].used)
        {
            index = (index + 1) % TABLE_SIZE;
        }
        Table[index].verb = verb;
        Table[index].used = 1;
        printf("Inserted verb '%s' at index %d.\n", verb, index);
    }
}

int search(char *key)
{
    int index = hash(key);
    int initialIndex = index;
    while (Table[index].used)
    {
        if (strcmp(Table[index].verb, key) == 0)
        {
            return index;
        }
        index = (index + 1) % TABLE_SIZE;
        if (index == initialIndex)
        {
            break;
        }
    }
    return -1;
}

int isVerb(char *word)
{
    const char *verbs[] = {"run", "eat", "go", "have", "make", "take", "see", "do", "say", "come"};
    for (int i = 0; i < 10; i++)
    {
        if (strcmp(word, verbs[i]) == 0)
        {
            return 1;
        }
    }
    return 0;
}

void processStatement(char *statement)
{
    char word[100];
    int i = 0;
    for (int j = 0; j <= strlen(statement); j++)
    {
        if (isalpha(statement[j]))
        {
            word[i++] = tolower(statement[j]);
        }
        else
        {
            word[i] = '\0';
            if (i > 0 && isVerb(word))
            {
                insert(word);
            }
            i = 0;
        }
    }
}

int main()
{
    for (int i = 0; i < TABLE_SIZE; i++)
    {
        Table[i].used = 0;
    }
    char statement[500];
    printf("Enter a statement: ");
    fgets(statement, sizeof(statement), stdin);
    processStatement(statement);
    return 0;
}
```

Sample Input and Output:

```
Enter a statement: I run and eat an apple while I see the sunset
Inserted verb 'run' at index 91.
Inserted verb 'eat' at index 84.
Inserted verb 'see' at index 47.
```