BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI (RAJ.)
CS F111 Computer Programming
LAB SESSION #13
(File Handling)

File handling is about dealing with files in the program. A file has to ***opened*** before data can be read from or written to it. When a file is opened, the operating system associates a ***stream*** with the file. A **common *buffer*** and a ***file position indicator*** are maintained in the memory for a function to know how much of the file has already been read. The stream is disconnected when the file is closed.

To open a file:

```
FILE *fp;                       // Defines file pointer
fp = fopen("foo.txt", "r");     //Only reading is permitted
```

If the call is successful, fopen returns a pointer to a structure typedef'd to FILE. The pointer variable, fp, assigned by fopen acts as a file handle which will be used subsequently by all functions that access the file.

File opening modes:

- **r** – Reading a file
- **w** – Writing a file
- **a** – Appending to the end of an existing file

When used with "**w**" or "**a**", **fopen** creates a file if it does not find one. **fopen** will fail if the file does not have the necessary permissions (r, w and a) or if the file does not exist in case of "**r**".

Extended file opening modes

Mode	File opened for
r	Reading only
r+	Both reading and writing
w	Writing only
w+	Both reading and writing
a	Appending only
a+	Reading entire file but only appending permitted

File read/write functions

The standard library offers a number of functions for performing read/write operations on files. All of these functions are found in the standard library **stdio.h**.

- Character-oriented functions (**fgetc** and **fputc**)
- Line-oriented functions (**fgets** and **fputs**)
- Formatted functions (**fscanf** and **fprintf**)

Operating systems have a limit on the number of files that can be opened by a program. Files must be closed with the **fclose** function **fclose(fp).** Closing a file frees the file pointer and associated buffers.

Example program to create a file and write text into it:

```
int main(){
        FILE *fp; char buf1[80], buf2[80];
        fputs("Enter a line of text: \n", stdout);
        fgets(buf1, 79, stdin);
        fp = fopen("foo", "w");
        fputs(buf1, fp);
        fclose(fp);
        fp = fopen("foo", "r");
        fgets(buf2, 79, fp);
        fputs(buf2, stdout);
        fclose(fp);
        return 0;
}
```

Example program to manipulate file offset pointer:

```
int main(){
        FILE *fp; int c, x;
        char buf[80] = "A1234 abcd"; char stg[80];
        fp = fopen("foo", "w+");
        fputs(buf, fp);
        rewind(fp);
        c = fgetc(fp);
        fputc(c, stdout);
        printf("\n");
        fscanf(fp, "%d", &x);
        fprintf(stdout, "%d", x);
        printf("\n");
        fgets(stg, 4, fp);
        fputs(stg, stdout);
        printf("\n");
        return 0;
}
```

Example program to display the content on a file using command line arguments:

```
#include <stdio.h>
 int main(int argc, char *argv[]){
        FILE *fp; char ch;
        fp = fopen(argv[1], "r");
        if (fp == NULL){
                printf("Error");
                return(0);}
        ch = fgetc(fp);
        while (ch != EOF){
                printf("%c", ch);
                ch = fgetc(fp);}
        fclose(fp);}}
```

Questions

1. Write a program to accept an English sentence and an English word from the user (using scanf() only), and then to check whether the word exists in the sentence or not. If it occurs, the program should print out how many times the word occurs in the sentence and then remove all the occurrences of the word from the sentence. For taking the input, you have to use input redirection, that is, you have to create a file named input.txt and then take the input from input.txt instead of the keyboard. Eg. **./a.out < input.txt.** Do not use any string processing functions.

   A sample input.txt could be as shown below

hi hello how are you hello hi hi

   input.txt

   No. of occurrences of hi is 2

   output string is: hello how are you hello

2. Given a file - **numbers.txt**, which contains one integer number per line. The first line of the above file contains the total number of integer numbers in the file. Write a program to read all the numbers from the above file using file operations into a dynamically created array. Then compute their average and display.

3. Given a file – **twoDPoints.csv** that contains two-dimensional points (**(x,y) pairs**). Every line in this file is a **x,y** pair (separated by comma). The first line contains the number of points in the file. You need to store all the points of this file into a 2-D array (say pointsArray). Then for each two-dimensional point stored in the pointsArray, you need to find the closest other point within the same array and store them in another array known as closePointsArray, which is also a 2-D array. Finally, print all the two-dimensional points stored in the closePointsArray.