UNIT – 5

PART – A :

1.) What are file I/O Functions?

A.) C provides a number of functions that helps to perform basic file operations**. Some of the file I/O Functions are :**

i.) fopen() - create a new file or open a existing file
ii.) fclose() - closes a file
iii.) getc() - reads a character from a file
iv.) putc() - writes a character to a file
v.) fscanf() - reads a set of data from a file
vi.) fprintf() - writes a set of data to a file, etc.

2.) List the file I/O and positioning functions?

A.) File I/O Finctions are :-

i.) fopen() - create a new file or open a existing file
ii.) fclose() - closes a file
iii.) getc() - reads a character from a file
iv.) putc() - writes a character to a file

Positioning Functions :-

In C programming, position functions reposition the filepointer in a file.

List of Positioning Function are:-

a.) fseek() - it moves the file pointer cursor to a specific position.
b.) rewind() – it moves the cursor from any position to the starting position.
c.) ftell() – it tells the position of the file pointer on that file with respect to starting of the file.

3.) Compare text file with binary file?

A.) **Text File :-**

i.) Text Files can store only text in a file.
ii.) The text file can be opened using any text editor.
iii.) Text files are easily readable and understandable.
iv.) They are less secured.

Binary File:-

i.) Binary files can store different types of data(image, audio, text) in a single file.
ii.) These files can be opened with only specific applications.
iii.) These files can be easily understood.
iv.) They are more secured.

4.) Give the syntax for fseek() ?

A.) **fseek()** syntax :- fseek(filepointer, offset, position/origin)

5.) What is errno ?

A.) **errno** is a preprocessor macro used for error indication in files.
The **errno.h** header file also defines a list of macros indicating different error codes, which will expand to integer constant expressions with type int.

Part – B :

1.) Explain file positioning functions ?
A.) In C programming, position functions reposition the filepointer in a file.

List of Positioning Function are:-

a.) **fseek()** – This function moves the file pointer cursor to a specific position.

The syntax for fseek() :- fseek(filepointer, offset, position/origin)

The value of position/origin must be one of the constants SEEK_SET, SEEK_CUR, or SEEK_END, to indicate whether the offset is relative to the beginning of the file, the current file position, or the end of the file, respectively.

b.) **rewind()** – This function moves the cursor from any position to the starting position.

The syntax for rewind() :- void rewind(FILE *Pointer)

c.) ftell() – This function tells the position of the file pointer on that file with respect to starting of the file.
The syntax for ftell() :- long ftell(FILE *pointer)

2.) Define file? Explain file operations?
A.) A collection of data which is stored on a secondary device like a hard disk is known as a **file**.
A file is generally used as real-life applications that contain a large amount of data.

There are different operations that can be carried out on a file. These are below, in this section, we will discuss one by one.

- Creation of a new file
- Opening an existing file
- Reading from a file
- Writing to a file
- Moving to a specific location in a file (seeking)
- Closing a file

Opening a File :

Before we can read (or write) information from (to) a file on a disk we must open the file. To open the file we have called the function **fopen().**

The **fopen()** function opens a stream for use and links a file with that stream. Then it returns the file pointer associated with that file.

mode: The string pointed to by mode determines how the file will be opened.Below table shows the legal values for mode.

Mode	Meaning
r	Open a text file for reading
w	Create a text file for writing. If it does not exist, then a new file is created. Here your program will start writing content from the beginning of the file.

a	Append to a text file. If it does not exist, then a new file is created. Here your program will start appending content in the existing file content.
r+	Open a text file for read/write
w+	Create a text file for read/write. It first truncates the file to zero length if it exists, otherwise creates a file if it does not exist.
a+	Append or create a text file for read/write. It creates the file if it does not exist. The reading will start from the beginning but writing can only be appended.

Reading and Writing to a File :-

For reading and writing to a text file, we use the functions fprintf() and fscanf().

They are just the file versions of printf() and scanf(). The only difference is that fprintf() and fscanf() expects a pointer to the structure FILE.

Moving to a specified location in C (Seeking) :-

fseek() seeks the cursor to the given record in the file.

Syntax of fseek() - fseek(filepointer, offset, position/origin)

The value of position/origin must be one of the constants SEEK_SET, SEEK_CUR, or SEEK_END, to indicate whether the offset is relative to the beginning of the file, the current file position, or the end of the file, respectively.

Closing a File :-

When we have finished reading from the file, we need to close it. there is a limit to the number of files you can have open at any one time, you may have to close one file before opening another. This is done using the function fclose().The fclose( ) function closes a stream that was opened by a call to fopen(). It writes any data still remaining in the disk buffer to the file and does a formal operating system level close on the file.

The syntax for fclose() - int fclose(FILE *fp);

fclose() function returns zero on success, or EOF if there is an error in closing the file.

3.) Write a C program to reverse the file content ?

A.) Program to reverse the file content :-

```
o  #include<stdio.h>
o  #include<conio.h>
o  int main()
o  {
o  FILE *fp;
o  char ch, fname[30], newch[500];
o  int i=0, j, COUNT=0;
o  printf("Enter the filename with extension: ");
o  gets(fname);
o  fp = fopen(fname, "r");
o  if(!fp)
o  {
       ▪  printf("Error in opening the file...\nExiting...");
       ▪  getch();
       ▪  return 0;
o  }
o  printf("\nThe original content is:\n\n");
o  ch = getc(fp);
o  while(ch != EOF)
o  {
```

```
                COUNT++;
                putchar(ch);
                newch[i] = ch;
                i++;
                ch = getc(fp);
        }
        printf("\n\n\n");
        printf("The content in reverse order is:\n\n");
        for(j=(COUNT-1); j>=0; j--)
        {
                ch = newch[j];
                printf("%c", ch);
        }
        printf("\n");
        getch();
        return 0;
        }
```

Output :-

```
Enter the filename with extension: code.txt

The original content is : Hello Computer.

The content in Reverse order is: Computer Hello
```

4.) Write a C program to copy one file content to another file ?
A.) Program to copy one file content to another file:

```
#include <stdio.h>
#include <stdlib.h> // For exit()
int main(){
  FILE *fptr1, *fptr2;
  char filename[100], c;
  printf("Enter the filename to open for reading \n");
  scanf("%s",filename);
  // Open one file for reading
  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);
  // Open another file for writing
  fptr2 = fopen(filename, "w");
  if (fptr2 == NULL){
    printf("Cannot open file %s \n", filename);
    exit(0);
  }
  // Read contents from file
  c = fgetc(fptr1);
  while (c != EOF){
    fputc(c, fptr2);
    c = fgetc(fptr1);
  }
  printf("\nContents copied to %s", filename);
  fclose(fptr1);
  fclose(fptr2);
  return 0;
```

Output:

```
Enter the filename to open for reading

a.txt

Enter the filename to open for writing

b.txt

Contents copied to b.txt
```

5.) What are the different Input/Output operations in Files?

A.) There are different operations that can be carried out on a file. These are below, in this section, we will discuss one by one.

- Creation of a new file
- Opening an existing file
- Reading from a file
- Writing to a file
- Moving to a specific location in a file (seeking)
- Closing a file

Opening a File :

Before we can read (or write) information from (to) a file on a disk we must open the file. To open the file we have called the function **fopen().**

The **fopen()** function opens a stream for use and links a file with that stream. Then it returns the file pointer associated with that file.

mode: The string pointed to by mode determines how the file will be opened.Below table shows the legal values for mode.

Mode	Meaning
r	Open a text file for reading
w	Create a text file for writing. If it does not exist, then a new file is created. Here your program will start writing content from the beginning of the file.
a	Append to a text file. If it does not exist, then a new file is created. Here your program will start appending content in the existing file content.
r+	Open a text file for read/write
w+	Create a text file for read/write. It first truncates the file to zero length if it exists, otherwise creates a file if it does not exist.
a+	Append or create a text file for read/write. It creates the file if it does not exist. The reading will start from the beginning but writing can only be appended.

Reading and Writing to a File :-

For reading and writing to a text file, we use the functions fprintf() and fscanf().

They are just the file versions of printf() and scanf(). The only difference is that fprintf() and fscanf() expects a pointer to the structure FILE.

Moving to a specified location in C (Seeking) :-

fseek() seeks the cursor to the given record in the file.

Syntax of fseek() - fseek(filepointer, offset, position/origin)

The value of position/origin must be one of the constants SEEK_SET, SEEK_CUR, or SEEK_END, to indicate whether the offset is relative to the beginning of the file, the current file position, or the end of the file, respectively.

Closing a File :-

When we have finished reading from the file, we need to close it. there is a limit to the number of files you can have open at any one time, you may have to close one file before opening another. This is done using the function fclose().The fclose( ) function closes a stream that was opened by a call to fopen(). It writes any data still remaining in the disk buffer to the file and does a formal operating system level close on the file.

The syntax for fclose() - int fclose(FILE *fp);

fclose() function returns zero on success, or EOF if there is an error in closing the file.

6.) Write a C program to display the contents of files using command line arguments ?
A.) Program to display the contents of files using command line arguments:

```
#include<stdio.h>
main()
{
FILE *fp;
char ch;
fp = fopen("E:\\abc.txt’, “r’);
while( (ch = fgetc(fp)) != EOF)
printf("%c",ch);
fclose(fp);
}
```

Output:

```
Hai how r u.
```

7.) Explain Error Handling Functions in Files?
A.) Generally when we want to perform any I/O operations on files then we may get some errors and to handle those errors we have to use some **Error Handling Functions.**

Some of the error handling functions are :-

i.) **feof()** Function :- This function is used to check whether the character is the “End of the file character” or not.
Syntax of **feof()** :- feof(FILE*pointer)
If the character is EOF(End of File) charcter then it returns “TRUE” or else it returns “FALSE”.

ii.) **ferror()** function :- The ferror() function is used to check for the file error on given stream.
A return value of zero indicates no error has occurred, whereas a non-zero value indicates an error occurred.
Syntax for ferror() :- int ferror(FILE *pointer)

iii.) **perror()** function :- It is a standard library function which prints the error messages, specified by compiler while performing read operation or a write operation in a file.

perror() stands for print error.

Syntax for perror() :- void perror(const char* str).