Assignment 3

1. Write a C program to compare two files and display whether the files are identical or different.

```
#include <stdio.h>

#include <stdlib.h>

int main() {

    FILE *f1, *f2;

    char file1[100], file2[100];

    int ch1, ch2;

    int different = 0;

    printf("Enter first file name: ");

    scanf("%s", file1);

    printf("Enter second file name: ");

    scanf("%s", file2);

    f1 = fopen(file1, "r");

    f2 = fopen(file2, "r");

    if (f1 == NULL || f2 == NULL) {

        printf("Error: Unable to open one or both files.\n");

        return 1;

    }

    // Compare each character

    while (1) {

        ch1 = fgetc(f1);

        ch2 = fgetc(f2);
        if (ch1 == EOF && ch2 == EOF)
            break;
        if (ch1 != ch2) {
            different = 1;
            break;
        }
    }
    if (different == 0)
        printf("Files are IDENTICAL.\n");
    else
        printf("Files are DIFFERENT.\n");
    fclose(f1);
    fclose(f2);
    return 0;
}
```

2. Explain different file handling functions with examples.

File handling in C allows you to create, read, write, append, and modify files stored on disk. C uses the <stdio.h> library which provides various built-in file functions.

Common File Handling Functions

Function	Purpose
fopen()	Open a file in a specific mode
fclose()	Close an opened file
fprintf()	Write formatted data to file
fscanf()	Read formatted data from file
fgets()	Read a string from file

Function	Purpose
fputs()	Write a string to file
fgetc()	Read a character
fputc()	Write a character
fread() / fwrite()	Read/Write binary data
feof()	Checks end of file
fseek()	Move file pointer
ftell()	Tells current file position
rewind()	Move pointer to beginning

Examples

(i) fopen()

```
FILE *fptr;

fptr = fopen("data.txt", "r");
```

(ii) fprintf()

```
fprintf(fptr, "Hello World");
```

(iii) fgets()

```
char str[100];

fgets(str, 100, fptr);
```

(iv) fgetc()

```
char c = fgetc(fptr);
```

(v) fseek()

```
fseek(fptr, 0, SEEK_END);
```

3. Explain different modes to open a file with examples.

Files in C are opened using the fopen(filename, mode) function.
The mode decides what you want to do with the file.

File Modes

Mode	Meaning
"r"	Read mode (file must exist)
"w"	Write mode (creates new or overwrites existing file)
"a"	Append mode (writes at end of file)
"r+"	Read + Write (file must exist)
"w+"	Read + Write (overwrites existing)
"a+"	Read + Append (creates file if not exists)

Examples of Modes

(i) Create a file

```
FILE *f = fopen("file.txt", "w");

fclose(f);
```

(ii) Write to file

```
FILE *f = fopen("file.txt", "w");

fprintf(f, "Some text here");

fclose(f);
```

(iii) Append to file

```
FILE *f = fopen("file.txt", "a");

fprintf(f, "\nHello again!");

fclose(f);
```

(iv) Read from file

```
FILE *f = fopen("file.txt", "r");

char data[100];
```

```
fgets(data, 100, f);

printf("%s", data);

fclose(f);
```

4. Difference between Structure and Union (in detail)

Feature	Structure	Union
Memory Allocation	Each member has separate memory	All members share same memory
Size	Sum of all members	Size of largest member
Use of Members	All members can be used at same time	Only one member valid at a time
Initialization	All members can be initialized	Only first member can be initialized
Usage	For storing different data together	For saving memory, variant data types

Example of Structure

```
struct Student {

   int roll;

   float marks;

   char name[20];

};
```

Example of Union

```
union Test {

   int x;

   float y;

};
```

5. Explain pointer in detail with example.

A pointer is a variable that stores the memory address of another variable.

Declaration

```
int *p;   // pointer to int
```

Example

```
int x = 10;

int *p = &x;

printf("%d", *p);   // prints value stored at x

printf("%p", p);    // prints address of x
```

Pointer Arithmetic

Because arrays store data contiguously, pointer arithmetic works:

```
int arr[3] = {10, 20, 30};

int *p = arr;

printf("%d", *(p+1));  // 20
```

6. What do you understand by dynamic memory?

Dynamic memory refers to memory allocated during program execution (run time) instead of compile time.

Used when size is not known in advance.

C provides four dynamic memory functions from <stdlib.h>:

(1) malloc()

Allocates memory but does NOT initialize.

```
int *p = malloc(5 * sizeof(int));
```

(2) calloc()

Allocates memory AND initializes to 0.

int *p = calloc(5, sizeof(int));

(3) realloc()

Changes size of previously allocated memory.

p = realloc(p, 10 * sizeof(int));

(4) free()

Releases memory.

free(p);

Advantages

- Memory used only when required
- Reduces wastage
- Helps in creating large structures (linked lists, trees, graphs)