1 Explain about call by value and call by reference with suitable examples.

(a) <u>Call by value</u>

In call by value, the value of actual parameters is copied into the formal parameters. In other words, we can say that the value of the variable is used in the function call in this method.

Eg-

```
#include <stdio.h>
 void swap(int, int);
 int main()
 {
    int a=10, b=20;
    printf("Befor swap a=%d, b=%d", a,b);
    swap(a,b);
    printf("After swap in main a=%d, b=%d", a,b);
 }
 void swap(int a, int y)
 {
    int temp;
    temp=a;
     a=b;
     b=temp;
    printf("After swap in function a=%d, b=%d", a,b);
 }
```

(b) Call by reference

The address of the variable is passed into the function call as a actual parameter. The value of the actual parameter can be modified by changing the formal parameter since the address of the actual parameter is passed.

Example- swapping

```
#include <stdio.h>
void swap(int *, int *);
int main()
{
    int a=10, b=20;
    swap(&a, &b);
    printf("a=%d, b=%d", a, b);
    return 0;
}

void swap(int *a, int *y)
{
    int t;
    t=*a;
    *a=*y;
    *y=t;
    printf("x=%d, y=%d", *a, *y);
}
```

2 Write a C programme for Multiplication of two matrixes.

Hello > CSE_108 > Arrays > palindrome.c > main()

```c
#include <stdio.h>

int main()
{
  int r, c, i, j,k,x,y;
  printf("Enter the number of rows and column for 1st matrix: ");
  scanf("%d%d", &r, &c);
  printf("Enter the number of rows and column for 2nd matrix: ");
  scanf("%d%d", &x, &y);
  int a[r][c], b[r][c], mul[r][c];

  printf("\nEnter elements of 1st matrix:\n");
    for (i = 0; i < r; i++)
    {
        for (j = 0; j < c; j++)
        {
            printf("Enter element a%d%d: ", i + 1, j + 1);
            scanf("%d", &a[i][j]);
        }
    }

    printf("Enter elements of 2nd matrix:\n");
    for (i = 0; i < x; i++)
    {
        for (j = 0; j < y; j++)
        {
            printf("Enter element b%d%d: ", i + 1, j + 1);
            scanf("%d", &b[i][j]);
        }
    }

    if (c==x && r==y)
    for (i = 0; i < r; i++)
    {
        for (j = 0; j < c; j++)
        {
            mul[i][j] = 0;
            for (k = 0 ; k<c; k++)
            {
                mul[i][j] += a[i][k] * b[k][j];
            }
        }
    }

    printf("\nMultiplication of two matrices: \n");
    for (i = 0; i < r; i++)
    {
        for (j = 0; j < y; j++)
        {
            printf("%d\t", mul[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

cd "/Users/manjilnepal/Documents/Manjil/Hello/CSE_108/Arrays/" && g
(base) manjilnepal@Manjils-MacBook-Air Manjil % cd "/Users/manjilne
E_108/Arrays/"palindrome
Enter the number of rows and column for 1st matrix: 2 3
Enter the number of rows and column for 2nd matrix: 3 2

Enter elements of 1st matrix:
Enter element a11: 1
Enter element a12: 2
Enter element a13: 3
Enter element a21: 1
Enter element a22: 2
Enter element a23: 3
Enter elements of 2nd matrix:
Enter element b11: 1
Enter element b12: 1
Enter element b21: 1
Enter element b22: 1
Enter element b31: 1
Enter element b32: 1

Multiplication of two matrices:
6       6
6       6
(base) manjilnepal@Manjils-MacBook-Air Arrays %
```

3 Write a C programme to implement Fibonacci series using recursion.

Hello > CSE Assignment_4 > C recurvise_fibonacci.c > main(void)

```c
#include<stdio.h>

int fibonacci(int num);
int main(void)
{
    int terms;
    printf("Enter terms: ");
    scanf("%d", &terms);
    for(int n = 0; n < terms; n++)
    {
        printf("%d\t", fibonacci(n));
    }
    printf("\n");
    return 0;
}
int fibonacci(int num)
{
    if(num == 0 || num == 1)
    {
        return num;
    }
    else
    {
        return fibonacci(num-1) + fibonacci(num-2);
    }
}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

(base) manjilnepal@Manjils-MacBook-Air CSE Assignment_4 % cd "/Users/manjilnepal/Documents/My C Code
jilnepal/Documents/My C Codes/Hello/CSE Assignment_4/"recurvise_fibonacci
Enter terms: 10
0       1       1       2       3       5       8       13      21      34
(base) manjilnepal@Manjils-MacBook-Air CSE Assignment_4 %
```

4 Explain different built in String handling functions with suitable examples?

4. Describe and explain different built in string handling functions with suitable examples

⇒ The different types of string handling functions are:-

(a) strlen()

This function counts the number of characters present in a string.

Syntax: strlen(string_name);

Eg:
```
#include <stringh.h>
int main()
{
    char a[] = "Hello";
    int length;
    length = strlen(a);
    printf("%d\n", length);
    return 0;
}
```

(b) strcpy()

This function copies the contents of one string into another.

Eg-
```
#include <string.h>
int main()
{
    char a[] = "Hello";
    char b[10];
    strcpy(b, a);
}
```

(c) strcmp()
This function compares two strings to find out whether they are same or different.
Eg- #include <string.h>

```
int main()
{
    char a[] = "Hello";
    char b[] = "hello";
    strcmp(a, b);
}
```

If the two strings are equal, return value is 0.
If a is less than b, return value is less than 0.
If a is greater than b, return value is greater than 0.

(d) strcat()
This function appends one string at the end of another.
Eg- #include <string.h>

```
int main()
{
    char a[10] = "Hello";
    char b[] = "Hi";
    strcat(a, b);
    printf("%s", a);
    return 0;
}
```

Output:
HelloHi

(e) strrev ()

It is used to show the reverse of a string.

Eg-

```
#include <string.h>
int main ()
{
    char a[] = "Hello";
    printf ("%s", strrev(a));
    return 0;
}
```

Output:

olleH

5 Write a C programmes to sort the given set of strings.

Hello > CSE_108 > Strings > sort_names.c > main()

```c
// WAP to sort names of 5 persons in alphabetical order
#include <stdio.h>
#include <string.h>

int main()
{
    int i, j;
    char names[5][10];
    char temp[10];

    for (i = 0; i < 5; i++)
    {
        printf("Enter name: ");
        scanf("%s", names[i]);
    }

    for (i = 0; i < 5; i++)
    {
        for (j = i+1; j<5; j++)
        {
            if (strcmp(names[i], names[j]) > 0)
            {
                strcpy(temp, names[i]);
                strcpy(names[i], names[j]);
                strcpy(names[j], temp);
            }
        }
    }
    for (i = 0; i < 5; i++)
    {
        printf("%s\n", names[i]);
    }
    return 0;

}
```

OUTPUT:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
cd "/Users/manjilnepal/Documents/My C Codes/Hello/CSE_108/S
(base) manjilnepal@Manjils-MacBook-Air My C Codes % cd "/Us
odes/Hello/CSE_108/Strings/"sort_names
Enter name: Manjil
Enter name: Hari
Enter name: Shyam
Enter name: Hawking
Enter name: Albert
Albert
Hari
Hawking
Manjil
Shyam
(base) manjilnepal@Manjils-MacBook-Air Strings %
```

6 What do u mean by a function? Give the structure of user defined function and explain about the arguments and return values.

Assignment

6. What do you mean by a function? Give the structure of user defined function and explain about the arguments and return values.

⇒ A function is a self contained block of statements that performs a coherent task of some kind.

* Structure of a user defined function:

```
return_type function_name (type1 arg1, type2 arg2, .....)
{
    // statements
    ..........
}
```

Example:

```
int sum(int a, int b)
{
    int s = a+b;
    return s;
}
```

* Argument and Return values

→ Values that are passed in the function call are called arguments

→ The return values is the data type of the value that the function returns.

eg-

```
float divide (float a, float b)
{
    float r = a/b;
    return r;
}
```

return type

arguments

7 Write a programme to read, calculate average and print student marks using array of structures.

Hello > CSE_108 > Structure > C student_report_card.c > N

```c
/* Write a programme to read, calculate average and print student marks using array of structures. */
#include <stdio.h>
#define N 50
typedef struct student
{
    int student_no;
    char student_name[20];
    int Physics, Chemistry, Math;
    float Avg;

}report;
int main()
{
    report s[N];
    int n, i;
    printf("Enter the number of students: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++)
    {
        printf("\nEnter Student's number: ");
        scanf("%d", &s[i].student_no);
        printf("Enter Student's name: ");
        scanf("%s", s[i].student_name);
        printf("Enter Physics marks: ");
        scanf("%d", &s[i].Physics);
        printf("Enter Chemistry marks: ");
        scanf("%d", &s[i].Chemistry);
        printf("Enter Math marks: ");
        scanf("%d", &s[i].Math);
        s[i].Avg = (s[i].Physics + s[i].Chemistry + s[i].Math)/3.0;
        printf("\n");
    }
    printf("*************Students Report Card*************\n\n");
    for (i = 0; i < n; i++)
    {
        printf("\tStudent's Number:  %d\n", s[i].student_no);
        printf("\tStudent's Name: %s\n", s[i].student_name);
        printf("\tPhysics marks: %d\n", s[i].Physics);
        printf("\tChemistry marks: %d\n", s[i].Chemistry);
        printf("\tMath marks: %d\n", s[i].Math);
        printf("\tAverage marks: %0.2f\n", s[i].Avg);
        printf("\n");
    }
    return 0;
}
```

OUTPUT:

```
cd "/Users/manjilnepal/Documents/My C Codes/Hello/CSE_108/Structure/" && gcc stu
udent_report_card
(base) manjilnepal@Manjils-MacBook-Air Structure % cd "/Users/manjilnepal/Docume
al/Documents/My C Codes/Hello/CSE_108/Structure/"student_report_card
Enter the number of students: 3

Enter Student's number: 1
Enter Student's name: Stephen_Hawking
Enter Physics marks: 90
Enter Chemistry marks: 99
Enter Math marks: 99


Enter Student's number: 2
Enter Student's name: Richard_Feynman
Enter Physics marks: 99
Enter Chemistry marks: 99
Enter Math marks: 99


Enter Student's number: 3
Enter Student's name: Elon_Musk
Enter Physics marks: 100
Enter Chemistry marks: 99
Enter Math marks: 99

*************Students Report Card*************

        Student's Number:  1
        Student's Name: Stephen_Hawking
        Physics marks: 90
        Chemistry marks: 99
        Math marks: 99
        Average marks: 96.00

        Student's Number:  2
        Student's Name: Richard_Feynman
        Physics marks: 99
        Chemistry marks: 99
        Math marks: 99
        Average marks: 99.00

        Student's Number:  3
        Student's Name: Elon_Musk
        Physics marks: 100
        Chemistry marks: 99
        Math marks: 99
        Average marks: 99.33
```

8 Differentiate between self-referential structure and nested structure with example.

8 Difference between self-referential structure and nested structure with examples.

Self-referential structure	Nested structure
→ Self referential structure are those structures that have one or more pointers which point to the same data type.	→ It is a type of structure in which one structure is a member of other structure.
→ It is useful to create data structures like linked lists, stacks, etc.	→ The members of a nested structure can be accessed by following syntax. variable name of Outer_structure • variable name of Nested structure • data member
→ Example:	→ Example:

```
struct node
{
  int data;
  struct node *next;
};
```

```
struct date
{
  int day, month, year;
};
struct student
{
  int roll;
  struct date. birthday;
};
```

9 Explain three dynamic memory allocation functions with suitable examples.

9. Explain three dynamic allocation functions with suitable examples.

⇒ The 3 dynamic allocation functions are:-

(i) malloc ()

'Malloc' or 'memory allocation' function is the method in C used to dynamically allocate a single large block of memory with specified size.

E.g.

```
ptr =(int *) malloc (n * size of(int));
```

(ii) calloc ()

'calloc' or 'contiguous allocation' method in C is used to dynamically allocate the specified number of blocks of memory of specified type.

E.g.

```
ptr = (float *) calloc (n, sizeof (float));
```

(iii) free ()

'free' method in C is used to dynamically de-allocate the memory.

E.g.

```
free(ptr);
```

10 Explain about storage classes.

10. Explain storage class.

⇒ Storage class in C are used to determine the life time, visibility, memory location and initial value of a variable.

There are four types of storage class:

(a) Automatic: It allocates memory automatically at runtime. Automatic variables are initialized to garbage by default.

(b) Static: The variables defined as static specifier can hold their value between the multiple function call.

(c) Register- Register allocates the memory into the CPU register depending upon the size of the memory remaining in the CPU.

(d) External - It is used to tell the compilier that the variable defined as extern is declared with an external linkage elsewhere in the program.

11 Develop a programme to create a library catalogue with the following members: access number, authors name, title of the book, year of publication and book price using structures.

Hello > CSE Assignment_4 > book_catalogue.c > ...

```c
/* WAP to create a library catalogue to access number, authors name, title of t
#include <stdio.h>
#include <string.h>

#define N 30

typedef struct catalouge
{
    int no;
    char name[N];
    char title[N];
    int year;
    int price;
}lib_cat;
int main()
{
    lib_cat library[20];
    int n, i;
    printf("Enter number of books: ");
    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        printf("\nEnter access number: ");
        scanf("%d", &library[i].no);
        printf("Enter author's first name: ");
        scanf("%s", library[i].name);
        printf("Enter the title of the book: ");
        scanf("%s", library[i].title);
        printf("Enter the year of publication: ");
        scanf("%d", &library[i].year);
        printf("Enter price of the book:$");
        scanf("%d", &library[i].price);
    }
    printf("**************Book's Catalouge**************\n");
    for (i = 0; i < n; i++)
    {
        printf("\tBook acces number: %d\n", library[i].no);
        printf("\tBook author name: %s\n", library[i].name);
        printf("\tBook's title: %s\n", library[i].title);
        printf("\tBook's year of publication: %d\n", library[i].year);
        printf("\tBook's price: $%d\n", library[i].price);
        printf("\n\n");
    }
    return 0;
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

cd "/Users/manjilnepal/Documents/My C Codes/Hello/CSE Assignment_4/" && gcc bo
(base) manjilnepal@Manjils-MacBook-Air My C Codes % cd "/Users/manjilnepal/Doc
ts/My C Codes/Hello/CSE Assignment_4/"book_catalogue
Enter number of books: 3

Enter access number: 1
Enter author's first name: Manjil
Enter the title of the book: Soft_Power
Enter the year of publication: 2030
Enter price of the book:$20

Enter access number: 2
Enter author's first name: Stephen
Enter the title of the book: Black_Hole
Enter the year of publication: 2015
Enter price of the book:$100

Enter access number: 3
Enter author's first name: Robert
Enter the title of the book: Rich_Dad_Poor_Dad
Enter the year of publication: 2000
Enter price of the book:$20
**************Book's Catalouge***************
        Book acces number: 1
        Book author name: Manjil
        Book's title: Soft_Power
        Book's year of publication: 2030
        Book's price: $20


        Book acces number: 2
        Book author name: Stephen
        Book's title: Black_Hole
        Book's year of publication: 2015
        Book's price: $100


        Book acces number: 3
        Book author name: Robert
        Book's title: Rich_Dad_Poor_Dad
        Book's year of publication: 2000
        Book's price: $20


(base) manjilnepal@Manjils-MacBook-Air CSE Assignment_4 %
```

12 Explain about command line arguments with an example.

12. Explain command line arguments with an example.

⇒ Command line arguments are simple parameters that are given on the system's command line and the values of these arguments are passed on to your program during program execution. When a program starts execution without user interaction, command-line arguments are used to pass value or file to it.

* Command line arguments are passed to the main function as argc and argv.
* Command line arguments are used to control the program from outside.
* argv[argc] is a Null pointer

Example:

```
#include <stdio.h>

int main(int argc, char *argv[])
{
  printf("Number of arguments: %d\n", argc);
  printf("The name of user is: %s\n", argv[1]);
  return 0;
}
```

Terminal

```
./a.out Manjil
```

Output:

```
2
Manjil
```

13 What is a Pointer? Explain pointer arithmetic operations with suitable examples.

13 What is a pointer? Explain pointer arithmetic operations with suitable example.

⇒ A pointer is a variable that stores the memory address variable as its value.
A pointer variable points to a data type (like int) of the same type. and is created with '*' operator.

⇒ Pointer arithmetic operations are:-

(a) Increament / Decreament of a pointer

→ It is a condition that also comes under addition. When a pointer is increamented, it actually increments by number equal to the size of data type.
eg-
If a integer pointer storing address 1000 is incremented, resulting address will be 1004.

→ When a pointer is decremented, it actually decrements by number equal to the size of data type.
eg-
An integer pointer storing address when decremented results to address 996.

(b) Addition of a pointer
When a pointer is added with a value, the value is first multiplied by the size of data type and then added to the pointer.

a	b	c
2	4	6
1000	1004	1008

int *a, *b, *c; } → b = a+1;
c = a+2;

(c) Subtraction of a pointer

When a pointer is subtracted with some value, the value is first multiplied by the size of the data type and then subtracted.

e.g.

a	b	c
2	6	3
1000	1004	1008

int *a, *b, *c;

Here,

~~b-1 = a;~~ a = b-1;

~~c-2 = a;~~ a = c-2;

14 What is a file? Explain different modes of opening a file.

14. What is a file? Explain different modes of opening a file.

⇒ A file is a collection of data that is stored permanently on a disk. OR
A file is a stream of bytes of arbitary length, which is used to hold data.

⇒ Different modes of opening a file are:

* (read-only) r → open an existing file for reading only

* (write-only) w → open a new file for writing only

* (append-only) a → open an existing file for appending i.e to add new information of the end of file

* r+ (read and write) → open existing file for update i.e for both reading and writing

* w+ (write and read) → open new file for both writing and reading

* a+ (append and read) → open and existing file for both append and reading

15 Write a programme to demonstrate read and write operations on a file.

Hello > CSE_Assignment_4 > read_and_write_in_file.c

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

int main()
{
    char name[100];
    FILE *fptr;

    fptr = fopen("/Users/manjilnepal/Documents/My C Codes/Hello/CSE Assignment_4\\hello.txt","w");
    if (fptr == NULL)
    {
        printf("Error!");
        exit(1);
    }
    else
    {
        printf("Enter any characters : ");
        scanf("%s",name);
    }

    fprintf(fptr, "%s", name);
    fclose(fptr);

    if ((fptr = fopen("/Users/manjilnepal/Documents/My C Codes/Hello/CSE Assignment_4\\hello.txt","r")) == NULL){
        printf("Error! opening file");
        // Program exits if the file pointer returns NULL.
        exit(1);
    }
    fscanf(fptr,"%s", name);
    printf("Character in hello.txt is = %s", name);
    fclose(fptr);
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
(base) manjilnepal@Manjils-MacBook-Air CSE_Assignment_4 % gcc read_and_write_in_file.c
(base) manjilnepal@Manjils-MacBook-Air CSE_Assignment_4 % ./a.out
Enter any characters : My_name_is_Manjil_Nepal
Character in hello.txt is = My_name_is_Manjil_Nepal
```

Inside the file:

read_and_write_in_file.c | CSE Assignment_4\hello.txt Hello

Hello > CSE Assignment_4\hello.txt

```
My_name_is_Manjil_Nepal
```

17 Write a programme to copy one file contents to another.

Hello > CSE_Assignment_4 > copy_file_contents_to_another.c > main()

```c

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

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("/Users/manjilnepal/Documents/My C Codes/Hello/CSE_Assignment_4:\\Hello.txt", "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("/Users/manjilnepal/Documents/My C Codes/Hello/CSE_Assignment_4:\\Hello.txt", "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;
}
```

16 Explain about fscanf(), fgets(), fprintf() and fwrite() functions with suitable examples.

18 Explain different file handling functions with syntaxes and suitable examples.

16. Explain about fscanf(), fprintf() and fwrite() functions with suitable example:

18. Explain different file handling functions with syntaxes and suitable example.

⇒ File handling refers to the method of storing data in C program in the form of an output or input that might have been generated while running a C program in a data file. ie a binary file or a text file for future analysis.

→ The different file handling functions are:-

(a) fopen() : It is used to open an existing file or a new file.

syntax: FILE *fopen(const char* filename, const char* mode);

(b) fprintf() : It is used to writing data into an available file

syntax:
int fprintf(FILE *stream, const char *format[, argument,...])

(c) fscanf() : It is used to read the data available in a file

syntax:
int fscanf(FILE *stream, const char* format[, argument...])

(d) fclose() : It is used to closed the program.

syntax: int fclose(FILE *stream)

(e) fwrite() : It writes a binary data to a file. It can be used to write any type of data such as integers, float, array, etc.

Syntax:

```
size_t fwrite(const void *ptr, size_t size, size_t count,
              FILE* stream);
```

→ Sample Programs:

* Usage of fscanf, fprintf and fwrite

```
#include <stdio.h>
int main()
{
    FILE *fp;
    fp= fopen("file.txt", "w");
    fprintf(

    int num;
    char str[20];
    FILE *fp;
    //use scanf to read from a file
    fp= fopen("input.txt", "r");
    fscanf(fp, "%d %s", &num, str);
    fclose(fp);
    //use printf() to write to a file
    fp= fopen("output.txt", "w");
    fprintf(fp, "Number: %d, String: %s\n", num, str);
    fclose(fp);
    //Use fwrite() to write binary data to a file
    fp= fopen("data.bin", "wb");
    fwrite(&num, sizeof(int), 1, fp);
    fwrite(str, sizeof(char), 20, fp);
    fclose(fp);
    return 0; }
```