Assignment-4

(7) Explain about call by value and call by reference with suitable examples.

→ Call by value is a method in which only copy of the input goes while original remains in the same place. In call by value, the function receives a copy of the argument's value. This means that if the function modifies the argument, the change will not be reflected outside the function.

Examples:

Call by value method:

```
#include <stdio.h>
 void swap(int,int);
 int main()
{
  int a,b;
  Scanf("%d %d",&a,&b);
  Printf("Before Swap a=%d \t b=%d", a,b);
    Swap(a,b);
 printf(" After Swap a=%d \t b=%d ", a,b);
}
 void swap(int x, int y)
{
  int temp;
  temp=x;
  x=y;
  y=temp;
}
```

OutPut

10 20

Before Swap a=10 b=20

After Swap a=10 b=20

Call by reference method:

```
#include<stdio.h>
 void swap(int, int);
 int main()
{
 int a, b;
 printf("Enter any two numbers:");
 scanf("%d %d", &a, &b);
 printf("\n Before Swap a=%d \t b=%d", a, b);
  swap(&a, &b);
printf("\n After Swap a=%d \t b=%d", a, b);
return 0;
}
 void swap(int x, int y)
 {
   int temp;
   temp = x;
   x = y;
   y = temp;
 }
```

Output

```
Enter any two number : 10  20
Before Swap  a = 10  b = 20
After Swap  a = 20  b = 10
```

(Q) 20) C Program for multiplication of two matrices in C.

=>
```
#include <stdio.h>
#define N 50
 int main ()
{
 int a[N][N] , b[N][N] , c[N][N] ; i, J, k, sum, m, n, P, q ;
Printf(" enter rows and columns for ~~that~~ first matrix:\n");
Scanf("%d %d", &m, &n);
Printf(" enter first matrix :\n");
 for( i=0; i<m; i++).
 {
  for( J=0; J<n ; J++)
  {
   Scanf("%d", &a[i][JJ);
 }
}
Printf ("Enter row & column for second matrix :\n");
Scanf(" %d %d", &P, &a);
Printf("\n enter second matrix:\n");
for (i=0; i<P; i++)
{
 for (J=0; J<9; J++)
{
```

```
scanf("%d", &b[i][j]);
}
}
Printf("\n first matrix is :\n");
for (i=0; i<m; i++)
{
for(j=0; j<n ; j++)
{
printf ("%d", a[i][j]);
}
printf("\n");
}
Printf("\n second matrix is :\n");
for( i=0; i<p; i++)
{
for (j=0; j<q ;j++)
{
Printf("%d", b[i][j]);
}
printf("\n");
}
if(n!=p)
{
Printf("It can not multiply");
}
else
{
for(i=0; i<m; i++)
{
for (j=0 ; j<q ; j++)
{
Sum=0
```

```
for (k=0; k<cm; k++)
{
  sum = sum + (a[i][k] * b[k][j]);
}
  c[i][j] = sum;
  }
  }
  printf("Multiplication is :\n");
  for(i=0;i<m;i++)
{
  for (j=0; j<q; j++)
{
  printf("%d \t", c[i][j]);
}
  printf("\n");
  }
  }
}
```

(2) Write a C Programe to implement fibonacci series using recursion.

```
#include <stdio.h>
  int fibonacci(int);
  int main()
{
  int n, i;
  Printf("Enter the number:");
  Scanf("%d",&n);
  for( i=0; i<=n; i++)
{
  printf("%d", fibonacci(i));
}
 return 0;
}
 int fibonacci(int i)
{
 if(i==0)
{
 return 0;
}
 else if(i==1)
{
 return 1;
}
 else
{
 return (fibonacci(n-1) + fibonacci(n-2));
}
}
```

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

a) The different built-in string handling functions are:-

(i) Strlen():- It counts the number of characters in a given string & return the integer values.

Syntax:- Strlen (String 1)

It returns total number of characters in String 1.

(ii) StrCpy():- It copies contents of one string into another

Syntax:- StrCpy (String1, String 2).

~~It copies first 6 characters String2 in String 1.~~

It copies string 2 value into string 1.

(iii) Strn CPy ():-

Syntax:- Strn CPy (String 1, String 2, 6)

It copies first 6 characters String 2 in string 1.

(iv) Str cat():- It is can concenates (appends) position of one string at the end of another string.

Syntax:- Strcat (String 1, String 2)

It joins (Appends) String 1 & string 2.

(v) Strncat():-

Syntax:- Strncat (string 1, String 2, 4)

Appends first 4 characters of String 2 to String 1.

(5) C Program to sort strings in alphabetical order.

```
#include<stdio.h>
#include<string.h>
int main()
{
 Char string[100];
 printf("Enter the string:");
 Scanf("%s", string);
 Char temp;
 int i, j;
 int n = Strlen(string)
 for(i=0; i<n-1; i++)
 {
  for(j=i+1; j<n; j++)
 {
  if(string[i]>string[j])
  {
   temp = string[i];
   String[i] = String[j];
   String[j] = temp;
  }
 }
 }
printf("The sorted string is: %s", string);
 return 0;
}
```

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

=) A function is a block of code that performs a specific task and can be called by other parts of a program.

=) The structure of user defined function is

```
return_type function_name (Parameter_list)
{

}
```

- return_type is data type of the value returned by the function. If the function does not return value, the return ~~value~~ type should be 'void'.
- 'function_name' is a unique identifier choosen by the programmer.
- 'parameter_list' is a list of variables that are passed to the function when it called. The parameter list can be empty if the function doesn't required any inputs.

Argument value :-

Argument values passed to the function when it is called. They are matched with the parameters in the function definition. The number of arguments passed to the function should match the number of parameter in the function definition.

- Return values:-

Return values is the value returned by the function when it is executed. The return statement is used to return a value from the function.
If the return type of the function is void, the function does not return any value

eg:

```
#include<stdio.h>
int add(int, int)
int main()
{
int a=20, b=30;
int result = add(a,b);
Printf("%d", result);
return 0;
}
int add(int x, int y)
{
return x+y;
}
```

In the above example, the function 'add' takes two arguments of type 'int', 'x' & 'y' are returns their sum as an int. The function is called with the arguments 'a' & 'b' with the value '30' & '20' in the 'main' function & the returned value is assigned to the variable 'result'.

(7) Write a Program to read, Calculate average and Print Student names using array of Structure.

```
#include <stdio.h>
  Struct Student
{
  int Stno;
  Char name [10];
  int m1, m2, m3;
  int total; float average;
}; void main()
{
  Struct Student S[10];
  int n, i
  printf(" Enter number of students: ");
  Scanf("%od", &n);
  for( i=0; i<n; i++)
  {
  Printf(" enter Student number : ");
  Scanf("%d", & S[i].Stno);
  printf(" Enter Student name: ");
  Scanf(" %s", S[i].name);
  printf(" Enter marks m1, m2, m3: ");
  Scanf(" %d %d %d", &S[i].m1, &S[i].m2, &S[i].m3)
 S[i] Total = S[i].m1 + S[i].m2 + S[i].m3;
  S[i]. average = (float) S[i].total /3.0;
}
  Printf(" Students information :\n");
```

```
for(i=0;i<n;i++)
{
 Printf("%d\n", S[i].Slno);
 Printf("%s\n", S[i].name);
 printf("%d %d %d\n", S[i].m1, S[i].m2, S[i].m3);
 Printf("%d\n", S[i].total);
 Printf("%f\n", S[i].average);
 }
}
```

O/P :-

Enter Student number :- 2
Enter Slno: 1
Enter name: Sonu
Enter marks, m1=86, m2=78, m3=90
Enter Slno=2
Enter name = Sonu
Enter names m1=87, m2=98, m3=67
Student information:
Slno: 1
name: Sonu
m1=86, m2=78; m3=90
Total = 254
Average: 84.666
Slno:-2
name: Sonu
m1=87, m2=98; m3=67
Total = 252
Average = 84.50.

(8) Differentiate between self-referrial structure and nested structure with examples.

A self referrial structure that contains a pointer to its own type. for eg:

```
Struct Student
{
  int roll no;
  Char name [20];
  Struct node * w;
};
```

In this example, the 'Student' Structure contains an integer data and pointer to another 'Student' Structure. This allows for the creation of linked lists, where each student points to the next student in the list.

On the other hand, a nested Structure is a structure that contains another structure as a member. for eg:

```
Struct address {
  Char Street [50];
  Char City [50];
  Char State [20];
};
Struct person {
  Char name [50];
  int age;
  Struct address wan;
};
```

In this example, the 'address' Structure is ~~called~~ needed within the 'person' Structure used this nested Structure to represent the address of the person.

In Summary, a self-referential structure is a structure that contains a pointer to its own type, and it is ~~called~~ used to create complex data structures, like linked lists, while a nested structure is a structure that contains another structure as a member, it is used to group related data together and make the code more readable and organized.

(9) Explain three dynamic memory allocations functions with suitable example.

⇒ The three dynamic memory allocation functions are:-

(i) malloc():- The malloc() function is used to dynamically allocate a block of memory of a specified size. For example, the following code allocates memory for an array of 10 integers.

```
int *p;
p = (int*) malloc (10 * sizeof (int));
```

(ii) calloc():- The calloc() function is similar to malloc(), but initializes the allocated memory to zero. For example the following code allocates memory for 2D array of 4x4 integers. and sets all elements to 0.

```
int **p;
p = (int**) calloc (4, sizeof (int*));
for (int i = 0; i < 4; i++)
p[i] = (int*) calloc (4, sizeof (int));
```

(11) Explain about Storage Classes in C.

→ The different Storage Classes are:-

(a) Auto :- variables declared within a function or block have automatic storage class by default. These variables are created when the function or block is called and are destroyed when the function or block exists.

(b) Register :- variables declared with the register Storage class stored in CPU registers, rather than in memory. This can lead to faster access time, but these are fewer registers available than memory locations, so the use of register should be used sparingly.

(c) Static :- variables declared with the static Storage class retain their values between function calls. They are initialized only one and retain their value throughout the program's execution.

(d) External :- variables declared with the extern Storage class are defined in one source file and can be accessed by other source files. This allows for the sharing of variables between source files.

(21) Develop a programme to create a library catalogue with the following members: access number, authors name, title of the book, year of Publication & book price using Structure.

```
#include<stdio.h>
 Struct writer
{
 int number;
 Char name[20];
Char title [20];
int year;
int price;
}
 int main()
{
Struct writer w;
Printf(" Enter access number:");
Scanf("%d", &w.number);
printf(" Enter authors name:");
scanf("%s", w.name);
Printf(" Enter title name:");
gets(w.title);
printf(" Enter year of publication:");
scanf("%d", &w.year);
printf "Enter price of book:");
scanf("%d", &w.price);
Printf(" The data of the following member is:\n");
Printf(" access number = %d, authors name is %s title of
      book is %s year of Publication %d book price = %d",
      w.number, w.title, w.year, w.price);
      return 0;
      }
```

(12) Explain about Command line argument with an example.

Command line arguments are input passed to a program when it is run from the command line. In C, they can be accessed using the 'argc' (argument count) and 'argv' (argument vector) variable in the main function.

Example of person that takes in one command line argument and prints it out.

```
#include <stdio.h>
int main(int argc, char* argv[])
{
  if argc!=2)
  {
  printf("Expected 1 argument. Got %d\n", argc-1);
  return 1;
}
printf("You entered: %s\n", argv[1]);
return 0;
}
```

In this example, 'argc' is the number of arguments passed to the program, including the name of the program itself. 'argv' is an array of strings, where each string is an argument.

(13) what is a pointer? Explain pointer arithmetic operations with suitable example.

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

or: → A pointer is a special type of variable that holds the address as a data item in it.

Pointer are used to dynamically allocate memory and to manipulate memory directly. Pointer arithmetic is a set of operations that can be performed on pointer.

for Example;

we can increment or decrement a pointer to point to the next previous memory locations, respectively.

```
#include<stdio.h>
int main()
{
int num=5
int *ptr=&num;
ptr++;
ptr--;
Printf("%d %d", *ptr, *ptr);
return 0;
}
```

In this example, 'PTr' is a pointer to an 'int' variable and '&num' is the memory address of the 'num' variable. The '++' operator increments the pointer to point to the next memory location. The '*' operator is the dereference operator, which is used to access the value stored at the memory location pointed by the pointer.

(24) What is a file? Explain different modes of opening a file.

A file is a collection of data that is stored on a computer's hardware drive or other storage device.

In C programming, files can be opened & read or written to using the standard input / output library (stdio.h).

There are several different modes that a file can be opened in:

(a) 'r' (read-only) :- It is open an existing file for reading only.

(b) 'w' (write-only) :- It is open a new file for writing only. If a file with specified filename currently exists, it will be destroyed and a new file with the same name will be created in its place.

(c) 'a' (append-only) :- It is open an existing file for appending i.e for adding new information at the end of files. A new file will be created if the file with the specified filename doesn't exist.

(d) 'r+' (read & write) :- It is open an existing file for update (i.e, for both reading and writing.

(e) 'a+' (~~append write read~~ read & write) :- open an ~~file~~ existing file for both appending and reading. A new file will be created if the file with specified filename doesn't exist.

(e) 'w+' (write & read) :- open a file for both writing and reading. If a file with the specified filename currently exists, it will be destroyed and new file will be created in its space.

(vi) write a programe to demonstrate read and write operations on a file.

1) Reading operations on a file.

```
FILE *fp;
fp = fopen("sample.text", w);
if (fp == NULL)
{
printf("\n unable to open a file");
exit (0);
}
```

2) writing operation on a file.

```
#include <stdio.h>
int main()
{
file = fopen ("example.text","w");
if(file == NULL)
{
printf(" Error operating file!\n");
return 0;
}
fprintf (file, "Hello world!\n");
fclose (files);
printf(" writing operations on the file is Successful \n");
return 0;
}
```

(16) Explain about fscanf(), fgets(), fprintf(), and fputs() functions with suitable example

a) fscanf() :- It is a function in C that reads formatted input from a file stream. It works similarly to the scanf() functions, but reads input from a ~~file~~ file rather than the standard input.

E.g:-

```
#include <stdio.h>
int main()
int x,y;
FILE *FP = fopen("input.txt", "r");
fscanf(fp, "%d %d", &x, &y);
printf("x : %d y : %d\n", x, y);
fclose(fp);
return 0;
}
```

(b) fgets() :- It is a function in C that reads a string from a file stream. It reads a line of text from the file, up to a specified maximum number of characters, & stores it in a character array.

Eg:-

```
#include <stdio.h>
int main()
{
char str[100];
FILE *fp = fopen("input.text", "r");
fgets(str, 100, fp);
printf("Read : %s\n", str);
fclose(fp);
return 0;
}
```

(c) fprintf() :- It is a function that writes formatted output to a file stream. It works similarly to the 'printf()' function but writes output to a file rather than the standard output.

E.g:

```
#include <stdio.h>
int main() {
int x=10, y=20;
FILE *fp = fopen("output.txt", "w");
fprintf(fp, "x:%d, y:%d\n", x, y);
fclose(fp);
return 0;
}
```

(d) fwrite() :- It a function that writes a specified number of bytes to a file stream. It can be used to write binary data to a file.

Eg:-

```
#include <stdio.h>
int main(){
int data[] = {1,2,3,4,5};
FILE *fp = fopen("output.bin", "wb");
fwrite(data, sizeof(int), 5, fp);
fclose(fp);
return 0;
}
```

In this, the fwrite() function write 5 integers size of (int) *5 (bytes) to the file "output.bin" in binary format.

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

→
```
#include <stdio.h>
int main()
{
 FILE *fp1, *fp2;
 Char ch; // open source file in read mode.
 fp1 = fopen ("source.text", "r");
 if (fp2 == NULL) // error handling.
{
 Printf(" Error opening source file.\n");
 return 0;
}
 // open destination file in write mode.
fp2 = fopen ("destination.text", "w");
 if (fp2 == NULL) // error handling
{
 Printf (" Error creating destination file.\n");
 return 0;
} // read from source file & write to destination file
while (ch = fgetc(fp1) != EOF)
 fputc ( ch, fp2);
}
 Printf(" file copied successfully.\n");
 fclose (fp1); //close both files.
 fclose (fp2);
 return 0;
}
```

(28) Explain different file handling functions with syntaxes & suitable example.

a) The use of file handling functions to perform operations on files such as opening, reading, writing & closing.

(a) fopen() :- This function is used to open a file and returns a pointer to the file.

Syntax:-

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

E.g:

FILE *fp;
fp = fopen ("test.text", "r");

(b) fclose() :- This function is used to close a file.

Syntax:-

int fclose (FILE *fp);

Eg:- fclose (fp);

(c) fgetc() :- This function is used to read a single character from a file.

Syntax:- int fgets (FILE *fp);

E.g:- char ch;
ch = fgets (fp);

(d) fputs() :- This function is used to write a single character to a file

Syntax:-

int ~~fputc~~ fputs (int char, FILE *fp);

E.g:

fputs ('a', fp);

(e) fread() :- This function is used to read a block of data from a file.

Syntax:
size_t fread(void *ptr, size_t size, size_t count, FILE *fp);

E.g:
char buffer[100];
fread(buffer, sizeof(char), 100, fp);

(f) fwrite() :- This function is used to write a block of data to a file.

Syntax:
size_t fwrite(const void *ptr, size_t size, size_t count, FILE *fp);

E.g.
char buffer[100] = "Hello, world!";
fwrite(buffer, sizeof(char), 100, fp);

(g) fseek() :- This function is used to move file pointer to a specific position in a file.

~~Syntax:- long int ftell(FILE *fp);~~
E.g.: ~~long int position;~~ fseek(fp, 0, SEEK_END);
~~position = ftell(fp);~~
Syntax: int fseek(FILE *fp, long int offset, int origin);

(h) ftell() :- This function is used to determine the current position of the file pointer.
Syntax:- long int ftell(FILE *fp);
E.g:- long int position;
position = ftell(fp);

(i) rewind(-) :- This function is used to move the file pointer to the begining.

Syntax :- void rewind (FILE *fp);

eg. :- rewind(fp)

(j) remove() :- This function is used to delete a file

Syntax :- int remove(const char *filename);

eg :- remove("test.txt");