Q.1. Explain different types of Parameters Passing mechanism techniques with suitable example.

Ans. When function is called, the calling function may have to pass some values to the called function.

There are two ways in which arguments or parameters can be passed to called function. They include:

- Call by value in which values of variables are passed by the calling function to the called function. The programs that we have written so far call functions using call-by-value method of passing parameters.

*example of call by value:-

→Program to swap two no.

```
#include<stdio.h>
void swap(int, int);        <- prototype function.
main()
{
 int a, b;
 scanf("%d%d", &a, &b);
 swap(a, b);                <- calling function.
}
 void swap(int x, int y)
 {
  int temp;
  temp = x;
  x = y;
  y = temp;
 printf("%d %d", x, y);
 }
```

function definition.

- Call by reference in which address of variables are passed by the calling function to called function.

  * example of call by reference:-

  → Program to swap two no.

```
#include<stdio.h>
void swap(int*, int*);
main()
{
int a, b;
scanf("%d%d", &a, &b);
swap(&a, &b);
printf("Swap a = %d \n b = %d", a, b);
}
void swap(int *x, int *y)
{
 int temp;
 temp = *x;
 *x = *y;
 *y = temp;
}
```

C program for multiplication of two matrices in C.

Program:-

```
#include <stdio.h>
void main( )
{
int a[30][30], b[30][30], c[30][30], i, j, k, r1, r2, c1, c2;
printf("Enter the size of row and column of matrix A: \n");
scanf("%d %d", &r1, &c1);
printf("Enter the size of row and column of matrix B: \n");
scanf("%d%d", &r2, &c2);
if(c1 = r2)
{
printf("Enter the element of matrix A: \n");
for(i=0 ; i<r1 ; i++)
 for(j=0; j<c1; j++)
  scanf("%d", a[i][j]);
printf("Enter the element of matrix B: \n");
 for(i=0; i<r2 ; i++)
  for(j=0; j<c2; j++)
   scanf("%d", b[i][j]);
printf("Multiplication of two matrices: \n");
for (i=0; i<r1 ; i++)
{
 for(j=0; j<c2; j++)
 {
  c[i][j]=0;
  for(k=0; k<r2; k++)
  c[i][j] += a[i][k] * b[k][j];
 }
```

```
    printf("\n");
}
else
printf("Matrices cannot be multiplied");
}
```

Write a C program to implement fibonacci series using recursion.

```
#include<stdio.h>
int fibonacci(int);
int main()
{
int n;
printf("Enter the no. of elements to be in series: ");
 scanf("%d", &n);
int i;
for(i=0; i<n; i++)
  printf("%d", fibonacci(i));
}
 int fibonacci(int n);
{
  if(n==0)
    return 0;
  if(n==1)
    return 1;
  else
    return (fibonacci(n-1) + fibonacci(n-2));
}
```

Q.4.) Describe and explain different built-in String handling functions with suitable examples.

Ans:-

① strcpy function :-

Syntax :-

```
char *strcpy(char *str1, const char *str2);
```

This function copies the string pointed to by str2 to str1 including the null character of str2. It returns the argument str1. Here str1 should be big enough to store the contents of str2.

Program :-

```
#include<stdio.h>
#include<string.h>
int main()
{
char str1[10], str2[10] = "HELLO";
strcpy(str1, str2);
printf("\n str1: %s", str1);
return 0;
}
```

Output :-

```
HELLO
```

② strcat function:-

It takes two strings as input and concatenates the second string to the first string, which means it will attach the second string to the end of the first string and save the concantenated string to the first string. The size of the first string should be large enough to save the result.

Program:-

```
#include <stdio.h>
#include <string.h>
int main()
{
char string1[10] = "Hello";
char string2[10] = "World";
strcat (string1, string2);
printf ("%s", string1);
return 0;
}
```

Output:-

HelloWorld

③ strcmp():-

strcmp() takes two strings as input, then compares them, and returns an integer based on the following condition:-

Expression	returns
str1 > str2	positive integer
str1 < str2	negative
str1 = str2	zero

Program:-

```
#include <stdio.h>
#include <string.h>
int main()
{
 char s1[10] = "HELLO";
 char s2[10] = "HEY";
 if (strcmp (s1, s2) == 0)
     printf(" Identical ");
 else
     print f(" Not identical");
 return 0;
}
```

Output:-

Not identical.

④ strlen() :-

It takes a string as input and writes the length of that string without including the end character '\0'.

Program:-

```
#include <stdio.h>
#include <string.h>
 main()
{
 char s1[10];
 scanf("%s", s1);
 printf("%d", strlen(s1));
}
```

⑤ strrev( )

It reverse the given string.

Program:-

```
#include <stdio.h>
#include <string.h>
int main()
{
 char str[10];
 scanf("%s", str);
 printf("%s", strrev(str));
 return 0;
}
```

⑥ strlwr( ):-

It converts the strings into lowercase.

Program:-

```
#include <stdio.h>
#include <string.h>
int main()
{
 char str[10];
 scanf("%s", str);
 printf("%s", strlwr(str));
 return 0;
}
```

⑦ strupr( ):-

It converts the strings into uppercase.

Program:-

```
#include <stdio.h>
#include <string.h>
main()
{
 char str[10];
  scanf ("%s", str);
   printf("%s", strupr (str));
}
```

C Program to sort strings in alphabetical order.

Program:-

```
#include <stdio.h>
#include <string.h>
int main()
{
int n, i, j, str[20][10], temp[10];
printf("Enter the number of string: \n");
 scanf("%d", &n);
 for (i=0; i<n ; i++)
  scanf("%s", str[i]);
 for(j=0; j<n ; i++)
 {
  for(j=i+1; j<n; j++)
  {
   if (strcmp(str[i], str[j]) >0)
    {
     strcpy(temp , str[i]);
     strcpy( str[i], str[j]);
     strcpy ( str[j], temp );
    }
   }
 }
  for (i= 0; i<n; i++)
   printf("%s", str[i]);
   return 0;
}
```

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

Ans. 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:-

```
return_type function_name(argument_list)
{
    // function body
}
```

Arguments:-

It is the values passed to a function when it is called. They are used to pass information to function so that it can perform its task. They are declared in the functions' argument list, and they must match the data type of the corrg parameter in the function definition.

Return Values:-

It is the values that a function returns to the calling code after it has finished executing. It is specified by the return_type in the function defn. If the function does not return a value, the return type should be 'void'. The return statement is used to return a value from a function.

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

```
#include<stdio.h>
 stouct student
{
float m1,m2,m3, avg;
}s[100];
 int main()
{
  int i,n;
  printf("Enter the no. of students");
 scanf("%d",&n);
 for(int i=0; i<n; i++)
  {
    printf("Enter the marks of three subjects \n");
    scanf("%f%f%f",&s[i].m1,&s[i].m2,&s[i].m3);
    s[i].avg = (s[i].m1+s[i].m2+s[i].m3)/3;
  }
  for(i=0; i<n; i++)
   printf("Average marks: %f\n", s[i].avg);
 return 0;
}
```

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

Ans

A self-refrential structure is a structure that contains a pointer or a reference to itself as one of its members. This allows for the creation of linked lists and other similar data structures.

For example :-

```
struct student
{
int name;
struct name* next;
};
```

A nested structure, on the other hand, is a structure that contains another structure as one of its members.

Example:-

```
struct outer
{
int x;
  struct inner
{
  int y, z;
} inner_struct;
};
```

9. Explain three dynamic memory allocation functions with suitable examples.

Ans. Dynamic memory allocation is the process of allocating memory at compile time

Dynamic memory allocation functions are:-

i. Malloc():-

It allocates a block of memory of a specific size.

Prototype:-

void *malloc(size_t size);

It allows a program to allocate an exact amount of memory explicity, as and when needed.

example:- ① malloc(30); allocates 30 bytes of memory and returns the address of byte 0.

② malloc(sizeof(float); allocates 4 bytes of memory and returns the address of byte0.

ii. Calloc():-

It allocates a multiple blocks of memory.

Prototype:-

void *calloc(size_t nitems, size_t size)

It provides access to the C memory heap, which is available for dynamic allocation of variable-sized blocks of memory.

example:- ① calloc(3,5); allocates 15 bytes of memory and returns the address of byte0.

② calloc(6, sizeof(float)); allocates 24 bytes of memory and returns the address of byte0.

iii) realloc() :-

It grows or shrinks allocated memory.

prototype:-

void *realloc(void *block, size_t size);

It adjusts the amount of memory allocated to the block to size, copying the contents to a new location in necessary.

example:-

int *a; a=(int *)malloc(30); // first 30 bytes of memory is allocated
a = (int *)realloc(a, 15); // later the allocated memory is shrink to 15 bytes.

Q. 10. Explain about storage classes in C.

Ans. The different storage classes in C are:-

i. 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.

ii. Register: Variables declared with the register storage class stored in CPU register, rather than in memory. This can lead to faster access time, but there are four registers available than memory locations, so the use of register should be used sparingly.

iii. Static :- Variables declared as "static" have a local scope, but their lifetime is the entire program execution. They are initialized only once and retain ~~ter~~ their value between function calls.

iv. External:- Variables declared as "extern" have a global scope and are visible to all functions in the program. They are defined in one source file and can be used in other source file by declaring them as extern.

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.

```
#include<stdio.h>
struct library
{
int ac_no, yr_pub, book_price;
char ath_name[100], ti_book[100];
}s[60];
int main()
{
int i,n;
printf("Enter the number of books you want to search for \n");
scanf("%d",&n);
for(i=0; i<n; i++)
{
printf("Enter author name of book %d",i+1);
scanf("%s",s[i].ath_name);
printf("Enter title of book %d", i+1);
scanf("%s",s[i].ti_book);
printf("Enter access no., of year of publication and price of book ");
scanf("%d %d %d",&s[i].ac_no, &s[i].yr_pub, &s[i].book_price);
}
```

```
for(i=0; i<n; i++)
{
printf("Author: %s \n", s[i].ath_name);
printf("Title: %s \n", s[i].tit_book);
printf("Access no.: %d \n", s[i].ac_no.);
printf("Year of publication: %d \n", s[i].yr_pub);
printf("price: %d \n", s[i].book_price);
}
return 0;
}
```

12. Explain about command line argument with an example.

Ans. Command line argument 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:-

```
#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;
}
```

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

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

Pointer arithmetic is the manipulation of pointers in arithmetic operations such as addition, subtraction, increment and decrement only. The following arithmetic operations can be performed on pointer:-

- Add^n of a no. to a pointer.
- Subtraction of a no. from a pointer.
- Subraction of a pointer from another pointer.

The following arithmetic operations cannot be performed on pointers:-

- Add^n, multiplication, division and modulus division of two pointers.
- Multiplication, division and modulo division of a pointer by a no.
- Shifting operations.

Example:- arithmetic operation in pointer using increment:

```
int array[] = {1, 2, 3, 4, 5};
int *p = array;
p++;   // p now points to array[1].
```

14. What is a file? Explain diff^n modes of opening a file.

A file is a collection of data or related records that is stored permanently on disk.

Different modes of opening a file:-

- "r" (read only) - for reading an existing file.
- "w" (write only) - for writing to a file (creating a new file if it does not exist). 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.

- "a" (append only) - for appending to an existing file. A new file will be created if the file with the specified file name does not exist.
- "r+" (read and write) - for reading and writing to an existing file.
- "w+" (write and read) - for reading and writing to a file (creating a new file if it does not exist). If a file with the specified filename currently exists, it will destroyed, and a new file will be created in its place.
- "a+" (append and read) - for reading and appending to an existing file. A new file will be created if the file with specified filename does not exist.

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

```
#include<stdio.h>
int main()
{ char a[] = "Hello";
  File *file;
  file = fopen("text.txt", "w");
  fwrite(a, sizeof(char), sizeof(a), file);
  fclose(file);

  //read from a file
  file = fopen("text.txt", "r");
  char b[100];
  fread(b, sizeof(char), sizeof(b), file);
```

```
printf("Data from file: %s \n", b);
fclose(file);
return 0;
}
```

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

Ans. The fscanf(), fgets(), fprintf() and fwrite() functions are used for reading and writing operations on files in C.

- fscanf() is used to read formatted data from a file.
- fgets() is used to read a line of text from a file.
- fprintf() is used to write formatted data to a file.
- fwrite() is used to write binary data to a file.

example:-

*fscanf():-

```
struct student
{ char a[100];
  int rno, rank;
}s1;
File *file;
file = fopen("text.txt", "r");
fscanf(file, "%d%d", &s1.rno, &s1.rank);
printf("Roll = %d \n Rank = %d ", s1.rno, s1.rank);
fclose(file);
```

```
file = fopen("text.txt", "r");
fgets(a, sizeof(a), file);
printf("File = %s \n", a);
fclose(file);

file = fopen("text.txt", "w");
fprintf(file, "Integer: %d \n", rno);
fprintf(file, "String = %s", a);
fclose(file);

file = fopen("text.txt", "wb");
fwrite(a, sizeof(char), sizeof(a), file);
fclose(file);
return 0;
}
```

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

```
#include <stdio.h>
int main()
{
File *file1, *file2;
char ch;
file1 = fopen("text.txt", "r");
if(file1 == NULL)
{ printf("Error opening file");
return 0;
}
file2 = fopen("dtext.txt", "r");
```

```
if (file2 == NULL)
{
  printf ("Error");
   return 0;
}
while (ch = fgets (file1) != EOF)
 fputs (ch, file2);
printf ("File copied successfully \n");
fclose (file1);
fclose (file2);
return 0;
}
```

18. Explain diff" file handling functions with syntaxes and suitable examples.

Ans. The use of file handling functions to perform operations on files such as opening, reading, writing and closing.

- fopen( ): This function is used to open a file and returns a pointer to the file.
  - * Syntax:-
    FILE *fopen (const char *file_name, const char *mode);
  - * example:-
    FILE *file;
    file = fopen ("text.txt", "r");

- fclose( ):

This function is used to close a file.

*Syntax:

```
int fclose(FILE *file);
```

*e.g:-

```
fclose(file);
```

- fgetc( ):

It is used to read a character from a file

*syntax:-

```
int fgetc(pointer_to_FILE);
```

*e.g:-

```
char ch;
ch= fgetc(fp);
```

- fputs( ):

It is used to write a single characted to a file.

*syntax:-

```
int fputs(int char, FILE *file);
```

*e.g:-

```
fputs('A', file);
```