Assignment-3 (Functions): -

- Write a program with a function declaration, definition and call (eg:- sum)
- Without prototype for a function give different type of, different no.of arguments to a function and test the behavior
- Create a local & global variable of same name and test the value
- Write a program to find how many times a function is being called (use local static variable as count)
- Try register storage class for local variables. Can we get address of register variable
- Try some nested calls

  sqrt(pow(2,abs(x))), putchar(toupper(ch)) etc
- Test linking of a extern variable & global variable within single program
- Create multifile program

  main.c – calling sum, square function

  sum.c - sum definition sqr.c – square definition

  compile each file separately and link them (* preferably use Makefile)

  Try **extern**, **static** linkage specifiers for global variables, functions,check symbol table of each object file using **nm** for every change

  *Create static/shared library of sum, square function and link with main

  *Write a single Makefile for creation of static/dynamic libraries, linking and execution
- Write a function to swap two variables using Pass by reference
- Write a single function to return sum, product of two no.s
- Recursion programs
  - factorial, sum of n no.s,
  - gcd, fibonacci series,
  - No. format conversions(decimal,binary and octal)
  - count no.of 1s or no.of 0's in a binary code

- Does this code follow healthy practices?

```
int* test(int x)
{
   int y=x*x;
   return &y;
}
```

- Try conversions between int*, const int* while passing parameters to functions

```
int *p;        const int *q;
test(p);       void test(const int* );
test(q);       void test2(int *);
```

- Passing 1D, 2D arrays to a function
  - sum,min,max of array elements
  - Matrix operations
- Can you return arrays from a function (a) base address (b) whole array
- Rewrite the following code using typedef.(Functions returning pointer to array)

```
 int ( *afun(int)  )[5];
int ( *afun(int x) )[5]
{
        int arr[5]={10,20,30,40,50};
        return &arr;    //This kind of return statement is a healthy practice?
}  //Hint:-  typedef int (*atype)[5];
```

- Function Pointers
  - Write a simple program to test function pointer
  - typedef for function pointer

    typedef int (*pftype)(); **(or)** typedef int (*pftype)(int,int);

    pftype pf1; pf1=sum; pf1(10,20);
  - Menu driven programs without if,else,switch(array of function pointers)
  - Rewrite this code using typedef

- Passing function names as parameters,Rewrite this code using typedef

```
void test(int x, int y, int (*fp) (int,int))
{
    int z = fp(x,y);
}
test(10,20,sum);
```