1. Write a C program to add, subtract, multiply, and divide two integers using a user-defined type function with a return type.

```
#include <stdio.h>
int add (int a, int b){
  return a + b;
}
int subtract (int a, int b){
  return a - b;
}
int multiply (int a, int b){
  return a * b;
}
int divide (int a, int b){
  return (float)a / b;
}

int main() {
  int a, b, choice;
  do {
    printf("\n\nChoose an Operation: \n");
    printf("Press 1 to add\n");
    printf("Press 2 to subtract\n");
```

```
printf("Press 3 to multiply\n");
printf("Press 4 to divide\n");
printf("Press 5 to exit\n");
scanf("%d", &choice);
printf("\n\nEnter two integers : ");
scanf("%d %d", &a, &b);
switch (choice){
case 1:
  printf("Sum :  %d", add(a,b));
  break;
case 2:
  printf("Difference :  %d", subtract(a,b));
  break;
case 3:
  printf("Product :  %d", multiply(a,b));
  break;
case 4:
  if(b != 0)printf("Quotient :  %d", divide(a,b));
  else printf("Error: Division by Zero!");
  break;
case 5:
  printf("Exit");
```

```
    break;
   default: printf("Invalid Choice!!");
   }
  } while (choice != 5);
  return 0;
}
```

2. Write a C program to calculate the sum of the first 20 natural numbers using a recursive function.

```
#include<stdio.h>


int sum(int n){
  if(n == 0 || n == 1) return n;


 else  return n + sum(n - 1);
}


int main(){
  printf("Result : %d", sum(20));
  return 0;
}
```

3. Write a C program to generate a Fibonacci series using a recursive function.

```
#include<stdio.h>


int Fibonacci(int term){
  if(term == 0 || term == 1) return term;
  else return Fibonacci(term - 1) + Fibonacci(term - 2);
}


int main(){
  int term;
  printf("Enter the term : ");
  scanf("%d", &term);
  printf("Result : " );
  for(int i = 0; i < term; i++) printf("%d , ", Fibonacci(i));


  return 0;
}
```

4. Write a C program to swap two integers using call-by-value and call-by-reference methods of passing arguments to a function.

```
#include<stdio.h>


void swapPassByValue(int a, int b){
  int temp = a;
  a = b;
  b = temp;
  printf("Inside the 'swapPassByValue' function =>  a : %d ,  b: %d \n" , a ,b);
}
void swapPassByReference(int *a, int *b){
  int temp = *a;
  *a = *b;
  *b = temp;
  printf("\n\nInside the 'swapPassByReference' function =>  a : %d ,  b: %d \n" , *a ,*b);
}


int main(){
  int a, b;
  printf("Enter the value of a : ");
  scanf("%d", &a);
```

```
  printf("Enter the value of b : ");
  scanf("%d", &b);

  //Pass By value
  swapPassByValue(a, b);
  printf("\n  outside 'swapPassByValue' function a : %d ,  b: %d " , a ,b);  //no change

  //Pass by Reference
  swapPassByReference(&a, &b);
  printf("\n  outside 'swapPassByReference' function a : %d ,  b: %d " , a ,b);  //no change

  return 0;
}
```

5. Write a C program to find the sum of the digits of the number using a recursive function.

```
#include<stdio.h>


int sumOfDigit(int n){
  if(n == 0) return 0;
 else  return (n% 10 ) + sumOfDigit(n / 10);
}


int main(){


  int n;
  printf("Enter a number : ");
  scanf("%d", &n);
  printf("Result : %d", sumOfDigit(n));


  return 0;
}
```

6. Write a C program to read an integer number and print the reverse of that number using recursion.

```
#include<stdio.h>
void reverse(int n){
  if(n == 0) return ;
  else {
    printf("%d", n%10);
    return reverse(n / 10);
  }
}

int main(){
  int n;
  printf("Enter a number : ");
  scanf("%d", &n);
  printf("Result : ");
  reverse(n);
  return 0;
}
```

7. Using functions, write a C program to find the maximum and minimum between two numbers.

```
#include<stdio.h>
int max(int a, int b){
 if(a > b) return a;
 else return b;
}
int min(int a, int b){
 if(a < b) return a;
 else return b;
}

int main(){
  int a , b;
  printf("Enter two numbers : ");
  scanf("%d %d", &a, &b);
  printf("Maximum : %d\n", max(a,b));
  printf("Minimum : %d", min(a,b));
  return 0;
}
```

8. Write a C program to check whether a number is even or odd using functions.

```
#include<stdio.h>
char* OddEven(int n){
  if(n == 0) return "Zero";
  else if( n % 2 == 0 ) return "Even";
  else return "Odd";
}

int main(){
  int num;
  printf("Enter a number : ");
  scanf("%d", &num);

  printf("The number %d is :  %s", num, OddEven(num));
  return 0;
}
```

9. Write a C program to check whether a number is a prime, Armstrong, or Perfect number using functions.

```
#include<stdio.h>


void isArmstrong (int num){
  int n, digitCount = 0, remainder, sum = 0;
  n = num;
    while (n != 0) {
        n = n / 10;
        digitCount++;
    }
    n = num;
    while (n != 0) {
        remainder = n % 10;
        int power = 1;
        for (int i = 0; i < digitCount; i++) {
            power *= remainder;
        }
        sum += power;
        n /= 10;
    }
```

```
   if( num == sum) printf("%d is an Armstrong number.\n\n", num);
   else printf("%d is an Armstrong number.\n\n", num);
}


void isPrime (int n){
  for (int i = 2; i < n; i++) {
    if(n % i == 0) {
      printf("%d is not a Prime number.\n\n" , n);
      return;
    }
  }
  if(n == 1) printf("1 is not a Prime Number.\n\n");
  else printf("%d is a Prime Number.\n\n", n);
}


void isPrefect(int n){
  int sum = 0;


  for(int i = 1; i < n; i++){
    if(n % i == 0){
      sum += i;
    }
```

```
  }

  if(sum == n) printf("%d is a Prefect Number.\n\n", n);
  else printf("%d is not a Prefect Number.\n\n", n);

}

int main(){
  int n;
  printf("Enter a number : ");
  scanf("%d", &n);
  printf("Result : \n");
  isArmstrong(n);
  isPrime(n);
  isPrefect(n);
  return 0;
}
```

10. Write a C program to find the power of any number using recursion.

```
#include<stdio.h>

int power(int base, int pow){
  if(pow == 1) return base;
  return base * power(base, pow - 1);
}

int main(){
  int base, pow;
  printf ("Enter base and Exponent : ");
  scanf("%d %d", &base, &pow);

  printf("Result: %d", power(base, pow));

  return 0;
}
```