PRACTICAL 1

1) Write a c program to implement function Swap using two different parameter passing mechanism.

```
#include <stdio.h>

void swap(int a ,int b)

{

   int temp;

   temp=a;

   a=b;

   b=temp;

   printf("After swapping a=%d, b=%d" ,a ,b);

}

int main()

{

   int a=2,b=3;

   printf("Before swapping a=%d, b=%d\n" ,a, b);

   swap(a,b);

}
```

OUTPUT

```
Before swapping a=2, b=3

After swapping a=3, b=2
```

2) Write a c program to store 5 values in appropriate data structure and compute addition for the same, modify the size to store 10 values and compute addition.

```
#include <stdio.h>

#include <stdlib.h>

int main()

{

   int* ptr;

   int n1, n2, i;

   printf("Enter size: ");

   scanf("%d", &n1);

   printf("Enter number of elements: %d\n", n1);

   // Dynamically allocate memory using malloc()

   ptr = (int*) malloc(n1 * sizeof(int));

   // Check if the memory has been successfully

   // allocated by malloc or not

   if (ptr == NULL) {

      printf("Memory not allocated.\n");

      exit(0);

   }

   else {

      printf("Memory successfully allocated using calloc.\n");

      for (i = 0; i < n1; ++i) {

         ptr[i] = i + 1;

      }

      printf("The elements of the array are: ");

      for (i = 0; i < n1; ++i) {

         printf("%d, ", ptr[i]);

      }
```

```
        printf("\nEnter the new size: ");
        scanf("%d", &n2);
        ptr = realloc(ptr, n1 * sizeof(int));
        printf("Memory successfully re-allocated using realloc.\n");
        // Get the new elements of the array
        for (i = 5; i < n2; ++i) {
          ptr[i] = i + 1;
        }
        // Print the elements of the array
        printf("The elements of the array are: ");
        for (i = 0; i < n2; ++i) {
          printf("%d, ", ptr[i]);
        }
        free(ptr);
    }
return 0;
}
```

OUTPUT

```
Enter size: 5
Enter number of elements: 5
Memory successfully allocated using calloc.
The elements of the array are: 1, 2, 3, 4, 5,
Enter the new size: 10
Memory successfully re-allocated using realloc.
The elements of the array are: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
```

3) Write a c program to get record (Player name, team name & runs of innings) of any three players from Indian cricket team. Print the record according to name of players in ascending order.

```
#include<stdio.h>

#include<string.h>

struct crickter{

  char name[10];

  char team[10];

  int runs;

}player[100],temp;

int main()  {

  printf("Enter info of 3 players as follow name, team and runs\n");

  for(int i=0; i<3; i++){

     scanf("%s %s %d",player[i].name, player[i].team, &player[i].runs);

  }

  for(int i=0; i<2; i++){

     for(int j=i+1; j<3; j++) {

        if(strcmp(player[i].name, player[j].name)>0) {

           temp=player[i];

           player[i]=player[j];

           player[j]=temp;

        }

     }

  }

  printf("Record of players in ascending  order\n");

  printf("---------------------------------------------\n");

  printf("Name\t\t Team name\t\t Runs\n");

  printf("---------------------------------------------\n");
```

```
    for(int i=0; i<3; i++){
        printf("%s\t\t %s\t\t\t%d\n", player[i].name ,player[i].team, player[i].runs);
    }
}
```

OUTPUT

```
Enter info of 3 players as follow name, team and runs
Virat India 200
Rohit India 150
Rahul India 100
Record of players in ascending  order
--------------------------------------------
Name          Team name          Runs
--------------------------------------------
Rahul          India              100
Rohit          India              150
Virat          India              200
```

4) Write a c program to create calculator (use user defined function named Calculator).

```
#include <stdio.h>
int calculator(double val1, double val2, char ope)
  {
  if(ope == '+') {
  printf("Addition of two numbers is %lf ", val1 + val2);
  }
  else if(ope == '-') {
  printf("Subtraction of two numbers is %lf ", val1 - val2);
  }
  else if(ope == '*')  {
  printf("Multiplication of two numbers is %lf ", val1 * val2);
  }
  else if(ope == '/')  {
  printf("Division of two numbers is %lf ", val1 / val2);
  }  else {
  printf("Invalid operator");
  }
}
void main()
  {
  double val1 , val2;
  char ope;
  printf("Enter First Number ");
  scanf("%lf", &val1);
  printf("Enter Second Number ");
  scanf("%lf", &val2);
  printf("Enter '+' for Add\n '-' for Sub\n '*' for Mul\n '/' for Div\n");
```

```
    scanf(" %c", &ope);
    calculator(val1,val2,ope);
}
```

OUTPUT

```
Enter First Number 5
Enter Second Number 7
Enter '+' for Add
 '-' for Sub
 '*' for Mul
 '/' for Div
+
 Addition of two numbers is 12.000000
```

Question 1: What do you mean by scope of variables? What is the scope of variable sin C?

In simple terms, scope of a variable is its lifetime in the program. This means that the scope of a variable is the block of code in the entire program where the variable is declared, used, and can be modified. The following example declares the variable x on line 1, which is different from the x it declares on line 2. The declared variable on line 2 has function prototype scope and is visible only up to the closing parenthesis of the prototype declaration. A scope in any programming is a region of the program where a defined variable can have its existence and beyond that variable it cannot be accessed. There are three places where variables can be declared in C programming language − Inside a function or a block which is called local variables.

Question 2: How can we store negative integer in C?

The C standard doesn't mandate any particular way of representing negative signed numbers. In most implementations that you are likely to encounter, negative signed integers are stored in what is called two's complement. The other major way of storing negative signed numbers is called one's complement. The two's complement of an N-bit number x is defined as 2^N - x. For example, the two's complement of 8-bit 1 is 2^8 - 1, or 1111 1111. The two's complement of 8-bit 8 is 2^8 - 8, which in binary is 1111 1000. This can also be calculated by flipping the bits of x and adding one.

For example:

```
 1          = 0000 0001
~1          = 1111 1110 (1's complement)
~1 + 1      = 1111 1111 (2's complement)
-1          = 1111 1111
```

Question 3: Differentiate between actual parameters and formal parameters.

Parameter

- A parameter is a special kind of variable, used in a function to refer to one of the pieces of data provided as input to the function to utilise.
- These pieces of data are called arguments.
- Parameters are Simply Variables.

Formal Parameter

- Parameter Written in Function Definition is Called “Formal Parameter.
- Formal parameters are always variables, while actual parameters do not have to be variables.

Actual Parameter

- Parameter Written in Function Call is Called “Actual Parameter”.
- One can use numbers, expressions, or even function calls as actual parameters.

  Example
  ```
  void display(int para1)
  {
  printf( “ Number %d “ , para1);
  }
   void main()
   {
   int num1; display(num1);
   }
  ```
  In above,
  para1 is called the Formal Parameter
  num1 is called the Actual Parameter.