Ex.No. : 1a **PROGRAM USING I/O STATEMENTS AND EXPRESSIONS**

SUM AND AVERAGE OF THREE INTEGERS

AIM

To write a C Program to perform I/O statements and expressions for Sum and average of three integers .

ALGORITHM

```
Step1: Start
Step 2: Declare variables and initializations
Step 3: Read the Input variable.
Step 4: Using I/O statements and expressions for computational processing.
Step 5: Display the output of the calculations.
Step 6: Stop
```

PROGRAM:

```c
#include<stdio.h>
#include<conio.h>
int main( )
{
int a,b,c;
int sum,average;
clrscr();
printf("Enter any three integers: ");
scanf("%d%d %d",&a,&b,&c);
sum = a+b+c;
average=sum/3
printf("Sum and average of three integers: %d %d",sum,average);
return 0;
}
```

Output:

```
Enter any three integers:

5

6

20

31

10
```

RESULT: Thus the program to Sum and average of three integers has been executed in C successfully.

Ex.No. : 1b **DISPLAY THE DIFFERENT DATA TYPES**

AIM

To write a C Program to perform I/O statements and display the different data types.

ALGORITHM:

Step 1**.** Start
Step 2. Read a character, a string, a float value, an integer number, a double value from the user using scanf() statement
Step 3. store it in respective variables ch, str, flt, num, dbl.
Step 4. Display all the variables using printf() statement.
Step 5. Stop

PROGRAM:

```
#include <stdio.h>
int main()
{
  char ch, str[20];
  float flt;
  int num,a=6,b=3;
  double dbl;
  printf(“Enter a character\n”);
  scanf(“%c”,&ch);
  printf(“Enter a string\n”);
  scanf(“%s”,&str);
  printf(“Enter a float value\n”);
  scanf(“%f”,&flt);
  printf(“Enter an integer number\n”);
  scanf(“%d”,&num);
  printf(“Enter a double value\n”);
  scanf(“%”,&dbl);
  printf("Displaying:\n”);
  printf("Character is %c \n", ch);
  printf("String is %s \n" , str);
  printf("Float value is %f \n", flt);
  printf("Integer value is %d\n" , no);
  printf("Double value is %lf \n", dbl);
  printf("\nExpressions:\n”);
  printf("Infix value is %d\n" , ((a+(2*b))/2));
  return 0;
}
```

Output:

```
Enter a character
'y'
Enter a string
“welcome”
Enter a float value
10.53
Enter an integer number
350
Enter a double value
20.23451
Displaying:
Character is
'y'
String is
“welcome”
Float value is
10.53
Integer value is
350
Double value is
20.23451
Expressions:
Prefix value is 8
Postfix value is 8
Infix value is 8
```

RESULT:

Thus the program to display the different data types has been executed in C successfully.

Ex.No. : 1c **ARITHMETIC OPERATORS**

AIM

To write a C Program to perform Arithmetic Operators.

ALGORITHM:

STEP 1: Start the program.

STEP 2: Declare all required variables and initialize them.

STEP 3: Get inputs and initialized the values.

STEP 4: Use the operator to perform the arithmetic operation.

STEP 5 : Display results.

STEP 6: Stop the program.

```
// Working of arithmetic operators
#include <stdio.h>
int main()
{
   int a = 9,b = 4, c;
   c = a+b;
   printf("a+b = %d \n",c);
   c = a-b;
   printf("a-b = %d \n",c);
   c = a*b;
   printf("a*b = %d \n",c);
   c = a/b;
   printf("a/b = %d \n",c);
   c = a%b;
   printf("Remainder when a divided by b = %d \n",c);
   return 0;
}
```

Output

```
a+b = 13
a-b = 5
a*b = 36
a/b = 2
Remainder when a divided by b = 1
```

RESULT:

Thus the program to perform Arithmetic Operators has been executed in C successfully.

Ex.No. : 1d INCREMENT AND DECREMENT OPERATORS

AIM

To write a C Program to Increment and Decrement Operation.

ALGORITHM:

STEP 1: Start the program.

STEP 2: Declare all required variables and initialize them.

STEP 3: Get inputs and initialized the values.

STEP 4: Use the operator to perform the Increment and Decrement operation.

STEP 5 : Display results.

STEP 6: Stop the program.

PROGRAM:

```
// Working of increment and decrement operators
#include <stdio.h>
int main()
{
   int a = 10, b = 100;
   float c = 10.5, d = 100.5;

   printf("++a = %d \n", ++a);
   printf("--b = %d \n", --b);
   printf("++c = %f \n", ++c);
   printf("--d = %f \n", --d);

   return 0;
}
```

Output:

```
++a = 11
--b = 99
++c = 11.500000
--d = 99.500000
```

RESULT:

Thus the program to perform Increment and Decrement operation has been executed in C successfully.

Ex.No. : 1e **ASSIGNMENT OPERATORS**

AIM

To write a C Program to perform the assignment operation.

ALGORITHM:

STEP 1: Start the program.

STEP 2: Declare all required variables and initialize them.

STEP 3: Get inputs and initialized the values.

STEP 4: Use the operator to perform the assignment operation.

STEP 5 : Display results.

STEP 6: Stop the program.

PROGRAM:

```
// Working of assignment operators
#include <stdio.h>
int main()
{
    int a = 5, c;

    c = a;      // c is 5
    printf("c = %d\n", c);
    c += a;     // c is 10
    printf("c = %d\n", c);
    c -= a;     // c is 5
    printf("c = %d\n", c);
    c *= a;     // c is 25
    printf("c = %d\n", c);
    c /= a;     // c is 5
    printf("c = %d\n", c);
    c %= a;     // c = 0
    printf("c = %d\n", c);
    return 0;
}
```

Output

```
c = 5
c = 10
c = 5
c = 25
c = 5
c = 0
```

RESULT:

Thus the program to perform the assignment operation has been executed in C successfully.

Ex.No. : 1f RELATIONAL OPERATORS

AIM

To write a C Program to perform the Relational operation.

ALGORITHM:

STEP 1: Start the program.

STEP 2: Declare all required variables and initialize them.

STEP 3: Get inputs and initialized the values.

STEP 4: Use the operator to perform the Relational operation.

STEP 5 : Display results.

STEP 6: Stop the program.

PROGRAM:

```
// Working of relational operators
#include <stdio.h>
int main()
{
   int a = 5, b = 5, c = 10;

   printf("%d == %d is %d \n", a, b, a == b);
   printf("%d == %d is %d \n", a, c, a == c);
   printf("%d > %d is %d \n", a, b, a > b);
   printf("%d > %d is %d \n", a, c, a > c);
   printf("%d < %d is %d \n", a, b, a < b);
   printf("%d < %d is %d \n", a, c, a < c);
   printf("%d != %d is %d \n", a, b, a != b);
   printf("%d != %d is %d \n", a, c, a != c);
   printf("%d >= %d is %d \n", a, b, a >= b);
   printf("%d >= %d is %d \n", a, c, a >= c);
   printf("%d <= %d is %d \n", a, b, a <= b);
   printf("%d <= %d is %d \n", a, c, a <= c);

   return 0;
}
```

Output

```
5 == 5 is 1
5 == 10 is 0
5 > 5 is 0
5 > 10 is 0
5 < 5 is 0
5 < 10 is 1
5 != 5 is 0
5 != 10 is 1
5 >= 5 is 1
5 >= 10 is 0
5 <= 5 is 1
5 <= 10 is 1
```

RESULT:

Thus the program to perform the Relational operation has been executed in C successfully.

Ex.No. : 1g **LOGICAL OPERATORS**

AIM

To write a C Program to perform the Logical operation.

ALGORITHM:

STEP 1: Start the program.

STEP 2: Declare all required variables and initialize them.

STEP 3: Get inputs and initialized the values.

STEP 4: Use the operator to perform the Logical operation.

STEP 5 : Display results.

STEP 6: Stop the program.

PROGRAM:

```
// Working of logical operators

#include <stdio.h>
int main()
{
   int a = 5, b = 5, c = 10, result;

   result = (a == b) && (c > b);
   printf("(a == b) && (c > b) is %d \n", result);

   result = (a == b) && (c < b);
   printf("(a == b) && (c < b) is %d \n", result);

   result = (a == b) || (c < b);
   printf("(a == b) || (c < b) is %d \n", result);

   result = (a != b) || (c < b);
   printf("(a != b) || (c < b) is %d \n", result);
```

```
    result = !(a != b);
    printf("!(a != b) is %d \n", result);

    result = !(a == b);
    printf("!(a == b) is %d \n", result);

    return 0;
}
```

Output:

```
(a == b) && (c > b) is 1
(a == b) && (c < b) is 0
(a == b) || (c < b) is 1
(a != b) || (c < b) is 0
!(a != b) is 1
!(a == b) is 0
```

RESULT:

Thus the program to perform the Logical operation has been executed in C successfully.

Ex.No. : 1h **SIZEOF OPERATOR**

AIM

To write a C Program to perform the sizeof operation.

ALGORITHM:

STEP 1: Start the program.

STEP 2: Declare all required variables and initialize them.

STEP 3: Use the operator to perform the sizeof operation.

STEP 4 : Display results.

STEP 5: Stop the program.

PROGRAM:

```
#include <stdio.h>
int main()
{
   int a;
   float b;
   double c;
   char d;
   printf("Size of int=%lu bytes\n",sizeof(a));
   printf("Size of float=%lu bytes\n",sizeof(b));
   printf("Size of double=%lu bytes\n",sizeof(c));
   printf("Size of char=%lu byte\n",sizeof(d));

   return 0;
}
```

Output:

```
Size of int=4 bytes
Size of float=4 bytes
Size of double=8 bytes
Size of char=1 byte
```

RESULT:

Thus the program to perform the sizeof operation has been executed in C successfully.

Ex.No: 2a **PROGRAM USING DECISION-MAKING**

LARGEST OF THREE NUMBERS

AIM:

To find the largest of three numbers using if...else if.

ALGORITHM:

Step 1. Start

Step 2. Read the values of x, y and z.

Step 3. If x is greater than y and x is greater than z then print x is greatest, otherwise go to step 3.

Step 4. If y is greater than z then print y is greatest, otherwise go to step 4.

Step 5. display z is greatest.

Step 6. Stop

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int x,y,z;
printf("Enter the values for x,y and z \n");
scanf("%d%d%d",&x,&y,&z);
if((x>y)&& (x>z))
printf(" %d is greatest",x);
else if (y>z)
printf ("%d is greatest",y);
else
printf("%d is greatest",z);
getch();
}
```

Output:

```
Enter the values for x, y and z
25
46
22
46 is greatest
```

RESULT:

Thus the program to find the largest of three numbers using if -else statement has been executed in C successfully.

Ex.No: 2b TO CHECK WHETHER THE GIVEN NUMBER IS POSITIVE NUMBER OR NEGATIVE NUMBER

AIM:

To write C programs to check whether the given number is a positive number or negative number and display the result.

ALGORITHM:

STEP 1: Start the program.
STEP 2: Declare all required variables and initialize them.
STEP 3: Get input number using scanf() function.
STEP 4: Use if condition to check whether the input number is greater than 0 if so print the given number is positive else print the given number is negative.
STEP 5: Stop the program.

PROGRAM:

```
#include<stdio.h> void main()
{
int n; printf("~~~~~~~~~~~~~~~~~~~~\n");
printf("POSITIVE OR NEGATIVE\n");
printf("~~~~~~~~~~~~~~~~~~~~\n");
printf("Enter the number\n"); scanf("%d", &n);
if(n>0)
printf("The given number %d is positive\n",n);
 else if(n<0)
printf("The given number %d is negative",n);
else
printf("The given number %d is neither positive nor negative",n);
}
```

OUTPUT:

```
POSITIVE OR NEGATIVE
 Enter the number
-15
The given number -15 is negative
```

RESULT:

Thus the program to check for number is positive number or negative number has been executed in C successfully.

Ex.No: 2c TO CHECK WHETHER THE GIVEN YEAR IS A LEAP YEAR OR NOT

AIM:

To write a C program to check whether the given year is a LEAP year or not.

ALGORITHM:

```
STEP 1:Start the program.
STEP 2:Declare the required variables.
STEP 3:Read the input from the user and store it in a variable.
STEP 4:Using if..else statement,
Check whether the given input is divisible by 400
Check whether the given input is divisible by 100
Check whether the given input is divisible by 4
If all conditions are stratified, then go to step 5 otherwise go to step 6.
STEP 5:Print the result as "It is a leap year" and goto step 7.
STEP 6:Print the result as "It is not a leap ear" STEP 7:Stop the program.
```

PROGRAM:

```
#include<stdio.h>
void main()
{
 int n;
printf("*******************\n");
printf("Leap year or not\n");
printf("*******************\n");
printf("Enter the year\n");
scanf("%d",&n);
if(n%100==0)
n%400==0 ? printf("The given year %d is a leap year",n):
printf("The given year %d is a non-leap year",n);
else if(n%4==0)
printf("The given year %d is a leap year",n);else
printf("The given year %d is non leap year",n);
}
```

OUTPUT:

```
********************
Leap year or not
********************Enter the year
2024
The given year 2024 is a leap year
```

RESULT:

Thus the program to check for leap year has been executed in C successfully.

Ex.No: 2d PAY CALCULATION USING SWITCH STATEMENT

AIM

To write a C Program to perform decision-making constructs- Pay Calculation.

ALGORITHM

1. Start
2. Declare variables and initializations
3. Read the Input variable.
4. Codes are given to different categories and da is calculated as follows:

   For code 1,10% of basic salary.

   For code 2, 15% of basic salary.

   For code 3, 20% of basic salary.

   For code >3 da is not given.
5. Display the output of the calculations .
6. Stop

PROGRAM

```
#include <stdio.h>
#include<conio.h>
void main ()
{
        float basic , da , salary ;
        int code ;
        char name[25];
        da=0.0;
        printf("Enter employee name\n");
        scanf("%[^\n]",name);
        printf("Enter basic salary\n");
        scanf("%f",&basic);
        printf("Enter code of the Employee\n");
        scanf("%d",&code);
        switch (code)
        {
                case 1:
                        da = basic * 0.10;
                        break;
                 case 2:
                        da = basic * 0.15;
                        break;
                case 3:
                        da = basic * 0.20;
                        break;
                default:
                        da = 0;
        }
        salary = basic + da;
        printf("Employee name is\n");
        printf("%s\n",name);
        printf ("DA is %f and Total salary is =%f\n",da, salary);
        getch();
}
```

OUTPUT

```
Enter employee name
Amsa
Enter basic salary
5000
Enter code of the Employee
1
Employee name is
Amsa
DA is 500.000000 and Total salary is =5500.000000
```

RESULT

Thus a C Program using decision-making constructs was executed and the output was obtained.

Ex.No: 2e PROGRAM TO CHECK IF ENTERED ALPHABET IS VOWEL OR A CONSONANT

AIM:

To write a C program to check if entered alphabet is vowel or a consonant using switch case.

ALGORITHM:

1. Start the program
2. Get the character for choice
3. Give the multiple choices using case statement whether one of the choices are in the consonants and print the same.
4. In the default case print that it is a consonant.
5. Display the result
6. Stop the program.

PROGRAM:

```
#include <stdio.h>
int main()
{
char alphabet;
printf("Enter an alphabet:");
scanf("%c",&alphabet);
switch(alphabet)
   {
        case 'a':
                    printf("Alphabet a is a vowel.\n");
                    break;
        case 'e':
                    printf("Alphabet e is a vowel.\n");
                    break;
        case 'i':
                    printf("Alphabet i is a vowel.\n");
                    break;
        case 'o':
                printf("Alphabet o is a vowel.\n");
                break;

        case 'u':  vowel.\n");break;


                printf("You entered a consonant.\n");

        default:

    }
p
r
i
n
t
f
(
"
A
l
p
h
a
b
e
t

u

i
s

a
```

```
return 0;
}
```

Output

```
Enter an
alphabet: i
Alphabet i
is a vowel.

Enter an
alphabet: o
Alphabet o
is a vowel.

Enter an
alphabet: u
Alphabet u
is a vowel.

Enter an alphabet: c
You entered a consonant.
```

Result:

Thus the C program using decision-making construct has been verified and executed successfully.

Ex. No: 2f PROGRAM TO CALCULATE THE SUM AND AVERAGE OF POSITIVE NUMBERS USING GOTO STATEMENT

AIM

To write a C Program to perform the goto operation.

ALGORITHM:

STEP 1: Start the program.

STEP 2: Declare all required variables and initialize them.

STEP 3: Get inputs and initialized the values.

STEP 4: Use the operator to perform the goto operation and calculate the sum and average.

STEP 5 : Display results.

STEP 6: Stop the program.

PROGRAM:

```
// Program to calculate the sum and average of positive numbers
// If the user enters a negative number, the sum and average are displayed.

#include <stdio.h>

int main()
 {

   const int maxInput = 100;
   int i;
   double number, average, sum = 0.0;

   for (i = 1; i <= maxInput; ++i)
 {
     printf("%d. Enter a number: ", i);
     scanf("%lf", &number);

     // go to jump if the user enters a negative number
     if (number < 0.0)
     {
       goto jump;
     }
```

```
    sum += number;
  }

jump:
  average = sum / (i - 1);
  printf("Sum = %.2f\n", sum);
  printf("Average = %.2f", average);

  return 0;
}
```

OUTPUT:

```
1. Enter a number: 3
2. Enter a number: 2.44
3. Enter a number: 5.9
4. Enter a number: -2.5
Sum = 11.34
Average = 3.78
```

RESULT:

Thus the program to perform the goto operation has been executed in C successfully.

Ex.No: 2g PROGRAM TO CALCULATE THE SUM OF NUMBERS USING BREAK-STATEMENT

AIM

To write a C Program to perform the break statement operation.

ALGORITHM:

STEP 1: Start the program.

STEP 2: Declare all required variables and initialize them.

STEP 3: Get inputs and initialized the values.

STEP 4: Use the operator to perform the break operation and calculate the sum.

STEP 5 : Display results.

STEP 6: Stop the program.

PROGRAM:

```
// Program to calculate the sum of numbers (10 numbers max)
// If the user enters a negative number, the loop terminates

#include <stdio.h>

int main() {
  int i;
  double number, sum = 0.0;

  for (i = 1; i <= 10; ++i) {
    printf("Enter n%d: ", i);
    scanf("%lf", &number);

    // if the user enters a negative number, break the loop
    if (number < 0.0) {
      break;
    }

    sum += number; // sum = sum + number;
  }

  printf("Sum = %.2lf", sum);
  return 0;
}
```

OUTPUT:

```
Enter n1: 3.5
Enter n2: 5.5
Enter n3: -4.5
Sum = 9.00
```

RESULT:

Thus the program to perform the break statement operation has been executed in C successfully.

Ex.No: 2h PROGRAM TO CALCULATE THE SUM OF NUMBERS USING CONTINUE STATEMENT

AIM

To write a C Program to perform the continue statement operation.

ALGORITHM:

STEP 1: Start the program.

STEP 2: Declare all required variables and initialize them.

STEP 3: Get inputs and initialized the values.

STEP 4: Use the operator to perform the continue operation and calculate the sum.

STEP 5 : Display results.

STEP 6: Stop the program.

PROGRAM:

```
// Program to calculate the sum of numbers (10 numbers max)
// If the user enters a negative number, it's not added to the result

#include <stdio.h>
int main()
{
  int i;
  double number, sum = 0.0;

  for (i = 1; i <= 10; ++i) {
    printf("Enter a n%d: ", i);
    scanf("%lf", &number);

    if (number < 0.0) {
      continue;
    }

    sum += number; // sum = sum + number;
  }
  printf("Sum = %.2lf", sum);

  return 0;
}
```

Output

```
Enter a n1: 2.3
Enter a n2: 4.8
Enter a n3: 4.6
Enter a n4: -9.6
Enter a n5: 4.4
Enter a n6: -6.9
Enter a n7: 3.5
Enter a n8: 2.66
Enter a n9: -7.8
Enter a n10: 5.66
 Sum = 27.92
```

RESULT:

Thus the program to perform the continue statement operation has been executed in C successfully.

LOOPS: FOR, WHILE, DO-WHILE

Ex. No: 3a **SUM OF FIRST N NATURAL NUMBERS**

AIM:

To check whether a given number is Sum of First N Natural Numbers.

ALGORITHM:

```
Step 1. Initialize the i value.
Step 2. The test expression i<11 is evaluated. Since 1 less than 11 is true, the body of for  loop
        is executed. This will print the 1
Step 3. The update statement is executed. Now, the value of i will be 2. Again, the test
        expression is evaluated to true, and the body of for loop is executed. This will print 2
Step 4. Again, the update statement i++ is executed and the test expression i< 11is
        evaluated. This process goes on until i becomes 11.
Step 5. When i becomes 11,i<11 will be false, and the for loop terminates.
```

PROGRAM:

```
// Program to calculate the sum of first n natural numbers
// Positive integers 1,2,3...n are known as natural numbers

#include <stdio.h>
int main()
{
   int num, count, sum = 0;
   printf("Enter a positive integer: ");
   scanf("%d", &num);
   // for loop terminates when num is less than count
   for(count = 1; count <= num; ++count)
   {
       sum += count;
   }

   printf("Sum = %d", sum);

   return 0;
}
```

Output:

```
Enter a positive integer: 10
Sum = 55
```

RESULT:

Thus the program to Sum of First N Natural Numbers has been executed in C successfully.

Ex. No: 3b **ARMSTRONG NUMBER OR NOT**

AIM:

To check whether a given number is Armstrong number or not.

ALGORITHM:

```
Step 1. Initialize the value of res to 0.
Step 2. Read the three-digit number in num variable to check for Armstrong number.
Step 3. Assign originalNum to the variable num.
Step 4. Extract the digits from the num.
Step 5. Find the cube of each digit in num and add them and store it in variable res.
Step 6. Repeat the step 5 untill the num is not equal to zero.
Step 7. Compare the res and original Num, if it is equal display the number is an
Armstrong number, otherwise display the number is not an Armstrong number.
```

PROGRAM:

```c
#include <stdio.h>
int main()
{
   int num, originalNum, rem, res = 0;
   printf("Enter a three digit integer: ");
   scanf("%d", &originalNum);
   num = originalNum;
   while (num != 0)
   {
      rem = num%10;
      res+= rem*rem*rem;
      num /= 10;
   }
   if(res == originalNum)
      printf("%d is an Armstrong number.",originalNum);
```

```
    else
        printf("%d is not an Armstrong number.",originalNum);
    return 0;
}
```

OUTPUT:

```
Enter a three-digit integer:
153
153 is an Armstrong number
```

RESULT:

Thus the program to check whether a given number is Armstrong number or not has been executed in C successfully.

Ex.No: 3c **SUM OF INTEGER USING DO-WHILE STATEMENT**

AIM:

To write a program to perform the sum of integer using do-while statement.

ALGORITHM:

STEP 1: Variable initialization, and then it enters the Do While loop.

STEP 2: Execute/Run a group of statements within the Programming loop.

STAEP 3:use Increment and Decrement Operator inside the loop to increment or decrements the values.

STEP 4:it checks the while condition. If the condition output is True, the code inside the C

STEP 5: Do while loop executes again. The process will last until the condition fails.

STEP 6: If it is False, compiler exits from it.

PROGRAM:

```
#include <stdio.h>

int main()

{

  int number, total=0;

   printf("\n Please Enter any integer below 10 \n");

  scanf("%d", &number);

  do

   {

     total = total + number;

     printf(" Number = %d\n", number);
```

```
    printf(" Total Value is: %d\n", total);

    number++;

  }while (number< 10);

 printf(" Total Value from outside the Loop is: %d \n", total);

 return 0;

}
```

OUTPUT:

```
Please Enter any integer below 10

6

 Number = 6

 Total Value is: 6

 Number = 7

 Total Value is: 13

 Number = 8

 Total Value is: 21

 Number = 9

 Total Value is: 30

 Total Value from outside the Loop is: 30
```

RESULT:

Thus the program to perform the sum of integer using do-while statement has been executed in C successfully.

Ex.No: 4a ARRAYS: 1D AND 2D, MULTI-DIMENSIONAL ARRAYS, TRAVERSAL

LIST OF EVEN NUMBERS

AIM:

To write a C program to Perform input 10 numbers in an array and display only the even numbers if present in the array.

ALGORITHM:

STEP 1:Start the program.

STEP 2:Include all required header files and declare all required variables.

STEP 4:Get the number of value from the user and store it in “n”.

STEP 5: sum of even value of “n” from the user and store it in an array.

STEP 6:Call the user-defined functions and display the result.

STEP 7:Stop the program.

PROGRAM:

```
#include <stdio.h>
#include <conio.h>

int main()
{
   int a[10], i;

   printf("Enter 10 numbers\n");
   for(i=0; i<10; i++)
   {
      scanf("%d",&a[i]);
   }

   printf("List of even numbers\n");
   for(i=0; i<10; i++)
   {
      if(a[i]%2==0)
      {
         printf("%d ",a[i]);
      }
   }
   return 0;
}
```

Output:

```
Enter 10 numbers
11
15
28
31
49
54
72
81
93
14
List of even numbers
28 54 72 14
```

RESULT:

Thus the program to Program to input 10 numbers in an array and display only the even numbers if present in the array has been executed in C successfully.

Ex.No: 4b **ADDITION OF TWO MATRIX**

AIM:

To write a C program to Perform the addition of two matrix in an array and display the sum of arrays.

ALGORITHM:

```
Step 1: Input matrix 1 and matrix 2.
Step 2: If the number of rows and number of columns of matrix 1 and matrix 2
        is equal,
Step 3: for i=1 to rows [matrix 1]
Step 4: for j=1 to columns [matrix 1]
Step 5: Input matrix 1 [i,j]
Step 6: Input matrix 2 [i,j]
Step 7:matrix 3 [i,j]= matrix 1 [i,j]+ matrix 2 [i,j];
Step 8: Display matrix 3 [i,j];
```

PROGRAM:

```
#include
int main(){
int r, c, mat1[100][100], mat2[100][100], sum[100][100], i, j;
printf("\nEnter the number of rows and columns : ");
scanf("%d %d", &r, &c);
printf("\nInput Matrix 1 elements : ");
for(i=0; i<r; ++i)
for(j=0; j<c; ++j)
{
scanf("%d",&mat1[i][j]);
}
```

```
printf("\nMatrix 1\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
printf("%d ",mat1[i][j]);
}
printf("\n");
}

printf("\nInput Matrix 2 elements : ");
for(i=0; i<r; ++i)
for(j=0; j<c; ++j)
{
scanf("%d", &mat2[i][j]);
}
printf("\nMatrix 2\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
printf("%d ",mat1[i][j]);
}
printf("\n");
}
// Adding Two matrices
printf("\nAdded Matrix\n");
for(i=0;i<r;++i)
for(j=0;j<c;++j)
{
sum[i][j]=mat1[i][j]+mat2[i][j];
}

// print the result

for(i=0;i<r;++i)
for(j=0;j<c;++j)
{
printf("%d ",sum[i][j]);

if(j==c-1)
{
printf("\n");
}
}
return 0;
}
```

Output:

```
Enter the number of rows and columns : 3 3
Input Matrix 1 elements : 1 0 2 0 3 0 4 0 5
Matrix 1
1 0 2
0 3 0
4 0 5
Input Matrix 2 elements : 1 0 2 0 3 0 4 0 2
Matrix2
1 0 2
0 3 0
4 0 2

Added Matrix
2 0 4
0 6 0
8 0 7
```

RESULT:

Thus the program to Perform the addition of two matrix in an array has been executed in C successfully.

Ex. No: 4c PROGRAM TO PRINT ELEMENTS OF THREE-DIMENSIONAL ARRAY

AIM:

To write a C program to Print elements of Three-Dimensional Array

ALGORITHM:

```
Step 1: Input matrix 1
Step 2: If the number of rows and number of columns of matrix 1
Step 3: for i=1 to rows [matrix 1]
Step 4: for j=1 to columns [matrix 1]
Step 5: Input matrix 1 [i,j,k]
Step 6: Display matrix 1 [i,j,k];
```

PROGRAM:

```
#include <stdio.h>
 int main(void)
{
   // initializing the 3-dimensional array
   int x[2][3][2] = { { { 0, 1 }, { 2, 3 }, { 4, 5 } },
                    { { 6, 7 }, { 8, 9 }, { 10, 11 } } };

   // output each element's value
   for (int i = 0; i < 2; ++i) {
       for (int j = 0; j < 3; ++j) {
           for (int k = 0; k < 2; ++k) {
               printf("Element at x[%i][%i][%i] = %d\n", i, j, k, x[i][j][k]);
           }
       }
   }
   return (0);
}
```

Output:

```
Element at x[0][0][0] = 0
Element at x[0][0][1] = 1
Element at x[0][1][0] = 2
Element at x[0][1][1] = 3
Element at x[0][2][0] = 4
Element at x[0][2][1] = 5
Element at x[1][0][0] = 6
Element at x[1][0][1] = 7
Element at x[1][1][0] = 8
Element at x[1][1][1] = 9
Element at x[1][2][0] = 10
Element at x[1][2][1] = 11
```

RESULT:

Thus the program to display the elements of Three-Dimensional Array has been executed in C successfully.

Ex. No: 5a STRING OPERATIONS USING BUILT-IN FUNCTIONS

AIM:

To write a C program to perform various string operations using built-in functions in string.h and stdlib.h.

ALGORITHM:

STEP 1: Start the program.

STEP 2: Declare the necessary variables.

STEP 3: Display the menu list and get the user's option and based on the input, perform the string operations.

STEP 4: Define user-defined functions to perform the following,

i. Count total number of words in the given sentence.

ii. Copy the string.

iii. Compare the two string.

STEP 5: Get an input string from the user and perform the requested operation.

STEP 6: Count the number of words in the given sentence and display the result.

STEP 7: Stop the program.

PROGRAM: To Print the String using scanf() and printf () Statement:

```
#include<stdio.h>
#include<string.h>

int main()
{
   char Name[10];
   printf("Enter A Name : ");
   scanf("%s",&Name);

   printf("Name You Have Typed : %s",Name);

   return 0;
}
```

OUTPUT:

```
Enter A Name : AIHT
Name You Have Typed : AIHT
```

PROGRAM: To Print the String using gets() and puts() function

```
#include<stdio.h>
#include<string.h>

int main()
{
   char Name[10];
   printf("String Functions Using Puts & Gets :\n");
   puts("Enter A Name :");
   gets(Name);

   puts(Name);

   return 0;
}
```

OUTPUT:

```
String Functions Using Puts & Gets :
Enter A Name :
AIHT
AIHT
```

PROGRAM: **To Perform the String Compare (strcmp) Operation**

```
#include<stdio.h>
#include<string.h>

int main()
{
   char strng[10],strng2[10];

   printf("Enter A 1st String : ");
   scanf("%s",&strng);

   printf("Enter A 2nd String : ");
   scanf("%s",&strng2);

   if(strcmp(strng,strng2)==0)
   {
      printf("Both Strings Are Same...!");
   }
   else
   {
      printf("Both Strings Are Not Same...!");
   }

  return 0 ;

}
```

OUTPUT:

```
Enter A 1st String : Hello
Enter A 2nd String : World
Both Strings Are Not Same...!
```

PROGRAM: To Perform the String Concatenation (strcat) Operation

```
#include <stdio.h>
#include <string.h>
int main()
{
   char s1[10] = "Hello";
   char s2[10] = "World";
   strcat(s1,s2);
   printf("String After Concatenation: %s\n\n", s1);
   return 0;
}
```

OUTPUT:

```
String After Concatenation: HelloWorld
```

PROGRAM: To Perform the String Copy (strcpy) Operation

```
#include <stdio.h>
#include <string.h>
int main()
{
   char s1[30] = "string 1";
   char s2[60] = "string 2 : I am gonna copied into s1";
   /* this function has copied s2 into s1*/
   strcpy(s1,s2);
   printf("String s1 is: %s\n\n", s1);
   return 0;
}
```

OUTPUT:

```
String s1 is: string 2 : I am gonna copied into s1
```

PROGRAM: To Perform the String Length (strlen) Operation

```
#include<stdio.h>
#include<string.h>

int main()
{
   char strng[10];

   printf("Enter A String : ");
   scanf("%s",&strng);
   printf("Length Of The String : %d",strlen(strng));

   return 0;
}
```

OUTPUT :

```
Enter A String : Programming
Length Of The String : 11
```

RESULT:

Thus the program to perform various string operations using built-in functions has been executed in C successfully.

Ex.No:6a

FUNCTIONS: CALL, RETURN, PASSING PARAMETERS BY (VALUE, REFERENCE), PASSING ARRAYS TO FUNCTION

SWAP OF TWO NUMBERS (CALL BY VALUE)

AIM:

To write a C program to perform call by value -Swap of two numbers using function concept.

ALGORITHM:

Main Function

```
Step 1: Start
Step 2: Initialize two variables 'x' and 'y', assign values to it
Step 3: Display the values of 'x' and 'y'
Step 4: Call the function swap with the values 'x' and 'y' as arguments
Step 5: Display the values of 'x' and 'y'
Step 6: Stop
```

Sub Function - Swap

```
Step 1 : Get two variables 'x' and 'y' as parameters.
Step 2 : Initialize temp
Step 3 : Assign,
             temp <- x
              x <- y
              y <- temp
Step 4 : Stop
```

PROGRAM:

```
#include <stdio.h>
void swap(int x, int y){
 int temp = x;
 x = y;
 y = temp;
}
int main(){
 int x = 22;
 int y = 18;
 printf("Values before swap: x = %d, y = %d\n", x,y);
 swap(x,y);
 printf("Values after swap: x = %d, y = %d\n\n", x,y);
}
```

OUTPUT :

```
Values before swap: x = 22, y = 18
Values after swap: x = 22, y = 18
```

RESULT:

Thus the program to call by value -Swap of two numbers using function concept has been executed in C successfully.

Ex.No:6b **SWAP OF TWO NUMBERS (CALL BY REFERENCE)**

AIM:

To write a C program to perform call by reference -Swap of two numbers using function concept.

ALGORITHM:

Main Function

```
Step 1: Start
Step 2: Initialize two variables 'x' and 'y', assign values to it
Step 3: Display the values of 'x' and 'y'
Step 4: Call the function swap with the address of  'x' and 'y' as arguments
Step 5: Display the values of 'x' and 'y'
Step 6: Stop
```

Sub Function - Swap

```
    Step 1: Get two pointer variables 'x' and 'y' as parameters.
    Step 2: Initialize temp
    Step 3: Assign,
                temp <- *x
                 *x <- *y
                 *y <- temp
    Step 4: Stop
```

PROGRAM:

```
#include <stdio.h>
void swap(int *x, int *y){
 int temp = *x;
 *x = *y;
 *y = temp;
}
int main(){
 int x = 22;
 int y = 18;
 printf("Values before swap: x = %d, y = %d\n", x,y);
 swap(&x,&y);
 printf("Values after swap: x = %d, y = %d\n\n", x,y);
 return 0;
}
```

OUTPUT:

```
Values before swap: x = 22, y = 18
Values after swap: x = 18, y = 22
```

RESULT:

Thus the program to call by reference -Swap of two numbers using function concept has been executed in C successfully.

Ex. No:7 **RECURSION**

AIM:

To write a C program to perform Factorial given number using recursion function.

ALGORITHM:

```
Step 1: Start
Step 2: Take integer variable A
Step 3: Assign value to the variable
Step 4: From value A upto 1 multiply each digit and store
Step 5: the final stored value is factorial of A
Step 6: Stop
```

PORGRAM

```
#include <stdio.h>
int main()
{
  int loop;
  int factorial=1;
  int number = 5;
  for(loop = 1; loop<= number; loop++)
{
    factorial = factorial * loop;
  }

  printf("Factorial of %d = %d \n", number, factorial);

  return 0;
}
```

OUTPUT:

```
Factorial of 5 = 120
```

RESULT:

Thus the program to perform Factorial given number using recursion function has been executed in C successfully.

Ex.No:8

1. Pointers: Pointers to functions, Arrays,Strings, Pointers to Pointers, Array of Pointers

Pointers to functions

```
#include <stdio.h>

void addOne(int* ptr)
{
   (*ptr)++;
}

int main()
{
   int* p, i = 22;
   p = &i;
   addOne(p);
   printf("Adding 1 using Pointer : %d", *p);
   return 0;
}
```

output :

```
Adding 1 using Pointer : 23
```

Array to function

```
#include <stdio.h>
float calculateSum(float num[]);
int main()
{
   float result, num[] = {23.4, 55, 22.6, 3, 40.5, 18};
// num array is passed to calculateSum()
   result = calculateSum(num);
   printf("Result = %.2f", result);
   return 0;
}
```

```
float calculateSum(float num[])
{
    float sum = 0.0;
    for (int i = 0; i < 6; ++i)
    {
        sum += num[i];
    }
    return sum;
}
```

OUTPUT :

```
Result = 162.50
```

Pointer to Array:

```
#include<stdio.h>
int main()
{
    int row =0;
    char arr[5][10] = {"RajeS1", "RajeS2", "RajeS3", "RajeS4", "RajeS5"};
    char (*ptrArr)[10] = NULL;
    ptrArr = arr;
    for (row = 0; row < 5; ++row)
    {
        printf("%s \n", ptrArr[row]);
    }
    return 0;
}
```

OUTPUT :

```
RajeS1
RajeS2
RajeS3
RajeS4
RajeS5
```

Pointer to Pointer :

```
#include<stdio.h>
int main()
{
    int var;
    int *ptr;
    int **pptr;
    var=3000;
    ptr=&var;
    pptr=&var;
    printf("value of var =%d\n",var);
    printf("value at *ptr=%d\n",*ptr);
    printf("value at *pptr=%d\n",*pptr);
    return 0;
}
```

OUTPUT :

```
value of var =3000
value at *ptr=3000
value at *pptr=3000
```

Ex.No:9

Structures: Nested Structures, Pointers to Structures, Arrays of Structures and Unions