List of experiments to be carried out CO wise

CO-1

L. No.	Theme	List of programs to be carried out	Extra program for fast learners
1	**General Programs**	1. WAP that accepts the marks of 5 subjects and finds the sum and percentage marks obtained by the student. 2. WAP that calculates the Simple Interest and Compound Interest. The Principal, Amount, Rate of Interest and Time are entered through the keyboard. 3. WAP to calculate the area and circumference of a circle. 4. WAP that accepts the temperature in Centigrade and converts into Fahrenheit using the formula C/5=(F-32)/9. 5. WAP that swaps values of two variables using a third variable.	1. C program to find maximum between two numbers 2. C program to check whether a number is positive, negative or zero 3. C program to check whether a number is divisible by 5 and 11 or not 4. C program to check whether a character is alphabet or not 5. C program to check vowel or consonant 6. Program to check uppercase or lowercase characters 7. Program to print day of week 8. C program to enter month number and print number of days in month 9. C program to find all roots of a quadratic equation 10. C program to check whether triangle is valid or not if angles are given
2	**If-Else & Case Switch Construct**	1. WAP that checks whether the two numbers entered by the user are equal or not. 2. WAP to find the greatest of three numbers. 3. WAP that finds whether a given number is even or odd. 4. WAP that tells whether a given year is a leap year or not. 5. WAP that accepts marks of five subjects and finds percentage and prints grades according to the following criteria: Between 90-100%--------------Print „A‟ 80-90%----------------------------Print „B‟ 60-80%---------------------------Print „C‟ Below 60%----------------------Print „D‟ 6. WAP that takes two operands and one operator from the user and perform the operation and prints the result by using Switch statement.	
3	**Loop Construct-1**	1. WAP to print the sum of all numbers up to a given number. 2. WAP to find the factorial of a given number. 3. WAP to print sum of even and odd numbers from 1 to N numbers. 4. WAP to print the Fibonacci series. 5. WAP to check whether the entered number is prime or not.	1. C program to print all natural numbers from 1 to n 2. C program to count number of digits in an integer 3. C program to calculate product of digits of a number 4. C program to display number in words 5. C program to find power of a number using for loop
4	**Loop**	1. WAP to find the sum of digits of the entered	6. C program to find GCD (HCF) of any two

	Construct-2	number. 2. WAP to find the reverse of a number. 3. WAP to print Armstrong numbers from 1 to 100. 4. WAP to convert binary number into decimal number and vice versa.	numbers 7. C program to find LCM of any two numbers C program to check whether a number is perfect number or not

CO-2

L. No.	Theme	List of programs to be carried out	Extra program for fast learners
5	Modular Programming	1. C program to find cube of a number using function 2. C program to check prime, Armstrong, perfect number using functions 3. C program to print all Armstrong numbers between given interval using function 4. C program to find reverse of a number using recursion 5. C program to find sum of even or odd number from 1 to n using recursion 6. C program to find factorial of a number using recursion 7. C program to generate nth Fibonacci term using recursion	
6	Array-1	1. WAP that simply takes elements of the array from the user and finds the sum of these elements. 2. WAP that inputs two arrays and saves sum of corresponding elements of these arrays in a third array and prints them. 3. WAP to find the minimum and maximum element of the array.	1. C program to input and print n elements in an array 2. C program to print all negative elements in an array 3. C program to count even and odd elements in an array 4. C program to print all unique element in an array 5. C program count total number of duplicate elements in an array 6. C program to count frequency of each element in an array 7. C program to merge two sorted array
7	Array-2	1. WAP to search an element in a array using Linear Search. 2. WAP to sort the elements of the array in ascending order using Bubble Sort technique.	

		3. WAP to add and multiply two matrices of order nxn. 4. WAP that finds the sum of diagonal elements of a mxn matrix.	8. C program to find reverse of an array 9. C program to search an element in the array 10. C program to sort an array in ascending order 11. C program to sort an array in descending order 12. C program to check whether two matrices are equal or not 13. C program to find sum of main diagonal elements of a matrix 14. C program to interchange diagonals of a matrix 15. C program to find transpose of a matrix 16. C program to count frequency of digits in an integer 17. C program to check symmetric matrix 18. C program to find length of a string 19. C program to compare two strings 20. C program to convert lowercase string to uppercase
8	**Array-3 & Structures**	1. Define a structure data type TRAIN_INFO. The type contain Train No.: integer type Train name: string Departure Time: aggregate type TIME Arrival Time : aggregate type TIME Start station: string End station : string The structure type Time contains two integer members: hour and minute. Maintain a train timetable and implement the following operations: (i)List all the trains (sorted according to train number) that depart from a particular section. (ii)List all the trains that depart from a particular station at a particular time. (iii)List all he trains that depart from a particular station within the next one hour of a given time. (iv)List all the trains between a pair of start station and end station.	

CO-3

9	**File Handling & Pattern**	1. WAP to swap two elements using the concept of pointers. 2. WAP to compare the contents of two files and determine whether they are same or not.	1. Write to a text file using fprintf() 2. Read from a text file using fscanf() 3. Writing to a binary file using fwrite() 4. Reading from a binary file using fread() 5. Program to Write a Sentence to a File 6. Program to read text from a file 7. C program to print pascal triangle 8. C program to print square or rectangle star pattern 9. C program to print right triangle star pattern 10. C program to print mirrored right triangle star pattern
10	**File Handling & Pattern**	1. WAP to check whether a given word exists in a file or not. If yes then find the number of times it occurs.	

Guidelines regarding lab work

- You should attempt all problems/assignments given in the list session wise.
- You may seek assistance in doing the lab exercises from the concerned lab instructor/Faculty In-charge. Since the assignments have credits, the lab instructor is obviously not expected to tell you how to solve these, but you may ask questions concerning the C language or a technical problem.
- For each program you should add comments (i.e. text between /* ... */ delimiters) above each function in the code, including the main function. This should also include a description of the function written, the purpose of the function, meaning of the argument used in the function and the meaning of the return value (if any).
- The program should be interactive, general and properly documented with real Input/ Output data.
- If two or more submissions from different students appear to be of the same origin (i.e. are variants of essentially the same program), none of them will be counted. You are strongly advised not to copy somebody else's work.
- It is your responsibility to create a separate directory to store all the programs, so that nobody else can read or copy.
- Lab record is compulsory.
- The list of the programs (list of programs given at the end, session-wise) is available to you in this lab manual. For each session, you must come prepare with the algorithms and the programs written in the Observation Book. You should utilize the lab hours for executing the programs, testing for various desired outputs and enhancements of the programs.
- As soon as you have finished a lab exercise, contact one of the lab instructor / in charge in order to get the exercise evaluated and also get the signature from him/her on the Observation book.
- Completed lab assignments should be submitted in the form of a Lab Record in which you have to write the algorithm, program code along with comments and output for various inputs given.
- The total no. of lab sessions are 10 and the list of assignments is provided session-wise. It is important to observe the deadline given for each assignment.

List of relevant sources of information

https://www.tutorialspoint.com/cprogramming/

https://www.codechef.com/c-programming

http://www.cprogramming.com/

https://www.codeproject.com

List of Equipments / Machines etc.

Borland C/C++ Compiler

Computer Machine.

Sample Quiz

GALGOTIAS COLLEGE OF ENGINEERING AND TECHNOLOGY

B.Tech: Programming Lab : QUIZ :Session-2019-20

Student Name : **Roll No-** **Branch:**

Max. Marks: 10 **Time: 15 minutes**

Put tick by choosing the correct option

CO1:

1. Who is father of C Language?

A. Bjarne Stroustrup B. Dennis Ritchie

C. James A. Gosling D. Dr. E.F. Codd

2. For 16-bit compiler allowable range for integer constants is ______ ?

A. -3.4e38 to 3.4e38 B. -32767 to 32768

C. -32768 to 32767 D. -32668 to 32667

3. C programs are converted into machine language with the help of

A. An Editor B. A compiler

C. An operating system D. None of the above

4. Which of the following shows the correct hierarchy of arithmetic operations in C

A. / + * - B. * - / +

C. + - / * D. * / + -

5. What is Keywords?

A. Keywords have some predefine meanings and these meanings can be changed.

B. Keywords have some unknown meanings and these meanings cannot be changed.

C. Keywords have some predefine meanings and these meanings cannot be changed.

D. None of the above

CO2:

6. The storage class in c

i) Automatic ii)Static iii)External iv) All the above

7. What is function?

A. Function is a block of statements that perform some specific task.

B. Function is the fundamental modular unit. A function is usually designed to perform a specific task.

C. Function is a block of code that performs a specific task. It has a name and it is reusable D. All the above

8. In switch statement, each case instance value must be _______?

A. Constant B. Variable

C. Special Symbol D. None of the above

9. What is the work of break keyword?

A. Halt execution of program B. Restart execution of program

C. Exit from loop or switch statement D. None of the above

CO3:

10.Choose the wrong file mode-

a. Read b.Write

c.Append d.None of these

Date: Signature of Examiner

Viva-questions

1. What is a pointer on pointer?
2. Distinguish between malloc() & calloc() memory allocation.
3. What is keyword auto for?
4. Explain the syntax for for loop.
5. What is a static variable?
6. Explain the purpose of the function sprintf().
7. What is the meaning of base address of the array?
8. What is a dangling pointer?
9. What is the purpose of the keyword typedef?
10. What is lvalue and rvalue?
11. What is the difference between actual and formal parameters?
12. Can a program be compiled without main() function?
13. What is the advantage of declaring void pointers?
14. Where an automatic variable is stored?
15. What is a nested structure?
16. What is the difference between variable declaration and variable definition?
17. Explain modular programming.
18. What is a token?
19. What is a preprocessor?
20. Does a break is required by default case in switch statement?
21. What are command line arguments?
22. What are the different ways of passing parameters to the functions? Which to use when?
23. Describe the file opening mode “w+”.
24. Is FILE a built-in data type?
25. Which key word is used to perform unconditional branching?
26. Explain the use of comma operator (,).
27. What is a static function?
28. Which built-in library function can be used to re-size the allocated dynamic memory?
29. Define an array.
30. What are enumerations?
31. Which built-in function can be used to move the file pointer internally?
32. What is a variable?
33. Who designed C programming language?
34. C is successor of which programming language?
35. What is the full form of ANSI?
36. Which operator can be used to determine the size of a data type or variable?
37. How does bitwise operator XOR works.
38. What is an infinite loop?

39. What is the default value of local and global variables?
40. Can a pointer access the array?
41. Name a function which can be used to close the file stream.
42. What is the purpose of #undef preprocessor?
43. Define a structure.
44. What is typecasting?
45. What is recursion?
46. What is the maximum length of an identifier?
47. When the macros gets expanded?
48. Can the structure variable be initialized as soon as it is declared?
49. Can we nest comments in a C code?
50. What is a constant?

List of Objectives

Programming for Problem Solving (KCS101) - Lab

Exp. No.	Object / Experiment Title
1	Theme-1:General C Programs
2	Theme-2:If-Else and Case Switch Constructs
3	Theme-3:Loop Constructs-1
4	Theme-4: Loop Constructs-1
5	Theme-5:Modular Programming
6	Theme-6:Array-1
7	Theme-7:Array-2
8	Theme-8:Structures
9	Theme-9:File Handling-1
10	Theme-10:File Handling & Pattern

Experiment wise Lab Manuals

Experiment No.	1
Theme	General Programs
Students are required to implement/execute/draw/make document	1. Input Window & Output Window 2. Algorithm 3. Flowchart 4. Program 5. Observation (Error Listing) 6. Final document with evaluator signature.
List of Programs to be implemented	1. WAP that accepts the marks of 5 subjects and finds the sum and percentage marks obtained by the student. 2. WAP that calculates the Simple Interest and Compound Interest. The Principal, Amount, Rate of Interest and Time are entered through the keyboard. 3. WAP to calculate the area and circumference of a circle. 4. WAP that accepts the temperature in Centigrade and converts into Fahrenheit using the formula C/5=(F-32)/9. 5. WAP that swaps values of two variables using a third variable. **Note: For Source code see the Appendix-1**

Experiment No.	**2**
Theme	**If-Else & Case Switch Construct**
Students are required to implement/execute/draw/make document	1. Input Window & Output Window 2. Algorithm 3. Flowchart 4. Program 5. Observation (Error Listing) 6. Final document with evaluator signature.
List of Programs to be implemented	1. WAP that checks whether the two numbers entered by the user are equal or not. 2. WAP to find the greatest of three numbers. 3. WAP that finds whether a given number is even or odd. 4. WAP that tells whether a given year is a leap year or not. 5. WAP that accepts marks of five subjects and finds percentage and prints grades according to the following criteria: Between 90-100%--------------Print „A" 80-90%---------------------------- Print „B" 60-80%--------------------------- Print „C" Below 60%--------------------- Print „D" 6. WAP that takes two operands and one operator from the user and perform the operation and prints the result by using Switch statement. **Note: For Source code see the Appendix-1**

Experiment No.	**3**
Theme	**Loop Construct-1**
Students are required to implement/execute/draw/make document	1. Input Window & Output Window 2. Algorithm 3. Flowchart 4. Program 5. Observation (Error Listing) 6. Final document with evaluator signature.
List of Programs to be implemented	1. WAP to print the sum of all numbers up to a given number. 2. WAP to find the factorial of a given number. 3. WAP to print sum of even and odd numbers from 1 to N numbers. 4. WAP to print the Fibonacci series. 5. WAP to check whether the entered number is prime or not. **Note: For Source code see the Appendix: 1**

Experiment No.	**4**
Theme	**Loop Construct-2**
Students are required to implement/execute/draw/make document	1. Input Window & Output Window 2. Algorithm 3. Flowchart 4. Program 5. Observation (Error Listing) 6. Final document with evaluator signature.
List of Programs to be implemented	1. WAP to find the sum of digits of the entered number. 2. WAP to find the reverse of a number. 3. WAP to print Armstrong numbers from 1 to 100. 4. WAP to convert binary number into decimal number and vice versa. **Note: For Source code see the Appendix-1**

Experiment No.	**5**
Theme	**Modular Programming**
Students are required to implement/execute/draw/make document	1. Input Window & Output Window 2. Algorithm 3. Flowchart 4. Program 5. Observation (Error Listing) 6. Final document with evaluator signature.
List of Programs to be implemented	1. C program to find cube of a number using function 2. C program to check prime, Armstrong, perfect number using functions 3. C program to print all Armstrong numbers between given interval using function 4. C program to find reverse of a number using recursion 5. C program to find sum of even or odd number from 1 to n using recursion 6. C program to find factorial of a number using recursion 7. C program to generate nth Fibonacci term using recursion **Note: For Source code see the Appendix-1**

Experiment No.	**6**
Theme	**Array-1**
Students are required to implement/execute/draw/make document	1. Input Window & Output Window 2. Algorithm 3. Flowchart 4. Program 5. Observation (Error Listing) 6. Final document with evaluator signature.
List of Programs to be implemented	1. WAP that simply takes elements of the array from the user and finds the sum of these elements. 2. WAP that inputs two arrays and saves sum of corresponding elements of these arrays in a third array and prints them. 3. WAP to find the minimum and maximum element of the array. **Note: For Source code see the Appendix-1**

Experiment No.	**7**
Theme	**Array-2**
Students are required to implement/execute/draw/make document	1. Input Window & Output Window 2. Algorithm 3. Flowchart 4. Program 5. Observation (Error Listing) 6. Final document with evaluator signature.
List of Programs to be implemented	1. WAP to search an element in a array using Linear Search. 2. WAP to sort the elements of the array in ascending order using Bubble Sort technique. 3. WAP to add and multiply two matrices of order nxn. 4. WAP that finds the sum of diagonal elements of a mxn matrix. **Note: For Source code see the Appendix-1**

Experiment No.	**8**
Theme	**Array-3 & Structures**
Students are required to implement/execute/draw/make document	1. Input Window & Output Window 2. Algorithm 3. Flowchart 4. Program 5. Observation (Error Listing) 6. Final document with evaluator signature.
List of Programs to be implemented	1. Define a structure data type TRAIN_INFO. The type contain Train No.: integer type Train name: string Departure Time: aggregate type TIME Arrival Time : aggregate type TIME Start station: string End station : string The structure type

	Time contains two integer members: hour and minute. Maintain a train timetable and implement the following operations: (i)List all the trains (sorted according to train number) that depart from a particular section. (ii)List all the trains that depart from a particular station at a particular time. (iii)List all he trains that depart from a particular station within the next one hour of a given time. (iv)List all the trains between a pair of start station and end station. **Note: For Source code see the Appendix-1**

Experiment No.	**9**
Theme	**File Handling**
Students are required to implement/execute/draw/make document	1. Input Window & Output Window 2. Algorithm 3. Flowchart 4. Program 5. Observation (Error Listing) 6. Final document with evaluator signature.
List of Programs to be implemented	1. WAP to swap two elements using the concept of pointers. 2. WAP to compare the contents of two files and determine whether they are same or not. **Note: For Source code see the Appendix-1**

Experiment No.	**10**
Theme	**File Handling**
Students are required to implement/execute/draw/make document	1. Input Window & Output Window 2. Algorithm 3. Flowchart 4. Program 5. Observation (Error Listing) 6. Final document with evaluator signature.
List of Programs to be implemented	1. WAP to check whether a given word exists in a file or not. If yes then find the number of times it occurs. **Note: For Source code see the Appendix-1**

Appendix-1

THEME :GENERAL PROGRAMS

1] Program to find sum of two numbers.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,s;
clrscr();
printf("Enter two no: ");
scanf("%d%d",&a,&b);
s=a+b;
printf("sum=%d",s);
getch();
}
```

Output:

```
Enter two no: 5
6
sum=11
```

2] Program to find area and circumference of circle.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int r;
float pi=3.14,area,ci;
clrscr();
printf("enter radius of circle: ");
scanf("%d",&r);
area=pi*r*r;
printf("area of circle=%f ",area);
ci=2*pi*r;
printf("circumference=%f ",ci);
getch();
}
```

Output:

```
enter radius of a circle: 5
area of circle=78.000
circumference=31.4
```

3] Program to find the simple interest.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int p,r,t,si;
clrscr();
printf("enter principle, Rate of interest & time to find simple interest: ");
scanf("%d%d%d",&p,&r,&t);
si=(p*r*t)/100;
printf("simple intrest= %d",si);
getch();
}
```

Output:
```
enter principle, rate of interest & time to find simple interest: 500
5
2
simple interest=50
```

4] Program to convert temperature from degree centigrade to Fahrenheit.
```
#include<stdio.h>
#include<conio.h>
void main()
{
float c,f;
clrscr();
printf("enter temp in centigrade: ");
scanf("%f",&c);
f=(1.8*c)+32;
printf("temp in Fahrenheit=%f ",f);
getch();
}
```
Output:
```
enter temp in centigrade: 32
temp in Fahrenheit=89.59998
```

5] Program to calculate sum of 5 subjects and find percentage.
```
#include<stdio.h>
#include<conio.h>
void main()
{
int s1,s2,s3,s4,s5,sum,total=500;
float per;
clrscr();
printf("enter marks of 5 subjects: ");
scanf("%d%d%d%d%d",&s1,&s2,&s3,&s4,&s5);
sum=s1+s2+s3+s4+s5;
printf("sum=%d",sum);
per=(sum*100)/total;
printf("percentage=%f",per);
getch();
}
```
Output:
```
enter marks of 5 subjects: 60
65
50
60
60
sum=300
percentage=60.000
```

THEME:IF-ELSE

C program to find maximum between two numbers

```
 #include <stdio.h>
int main()
{    int num1, num2;
    printf("Enter any two numbers:\n");
    scanf("%d %d", &num1, &num2);
    if(num1 > num2)
    {printf("%d is maximum", num1);
    }
    else
    {printf("%d is maximum", num2);
    }
    return 0;
}
```

C program to find maximum between three numbers

```
#include <stdio.h>
int main()
{   int num1, num2, num3, maximum;
     printf("Enter three numbers to find maximum: \n");
    scanf("%d%d%d", &num1, &num2, &num3);
     if((num1>num2) && (num1>num3))
    {maximum = num1;
    }
    else if(num2>num3)
    {maximum = num2;
    }
    else
    {maximum = num3;
    }
     printf("\nMaximum among all three numbers = %d\n",
     maximum); return 0;
}
```

Program to find maximum between three numbers using nested if

```
#include <stdio.h>
int main()
{   int num1, num2, num3, maximum;
     printf("Enter three numbers to find maximum: \n");
    scanf("%d%d%d", &num1, &num2, &num3);
     if(num1>num2)
    {if(num1>num3)
        {maximum = num1;
        }
        else
        {maximum = num3;
        }
    }
    else
    {if(num2>num3)
        {maximum = num2;
        }
        else
        {maximum = num3;
        }

    }
     printf("\nMaximum among all three numbers = %d", maximum);
     return 0;
}
```

C program check even or odd using if else #include <stdio.h>

```
int
main()
{ int
num;
     printf("Enter any number to check even or odd: ");
    scanf("%d", &num);
     if(num%2 == 0)
    {printf("Number is Even.\n");
    }
    else
    {printf("Number is Odd.\n");
    }
     return 0;
}
```

C program to check Leap Year

It is a special year containing one extra day i.e. total 366 days in ayear. If the year is exactly divisible by 4 and not divisible by 100 then its Leap Year Else if the year is exactly divisible by 400 then its Leap Year Else its a common year. #include <stdio.h>

```
int main()
{ int year;
    printf("Enter
    year : ");
    scanf("%d",
    &year);
    if(((year%4 == 0) && (year%100 !=0)) || (year%400==0))
    {printf("LEAP YEAR");
    }
    else
    {printf("COMMON YEAR");
    }
    return 0;
}
```

C program to check whether a number is positive, negative or zero #include <stdio.h>

```
int
main()
{ int
num;
     printf("Enter any number: ");
   scanf("%d", &num);
     if(num > 0)
   {printf("Number is POSITIVE");
   }
   else if(num < 0)
   {printf("Number is NEGATIVE");
   }
   else
   {printf("Number is ZERO");
   }
    return 0;
}
```

C program to check whether a number is divisible by 5 and 11 or not #include <stdio.h>

```
 int main()
{    int num;
    printf("Enter any number: ");
   scanf("%d", &num);
    if((num%5 == 0) && (num%11 == 0) )
   {printf("Number is divisible by 5 and 11");
   }
   Else
   {printf("Number is not divisible by 5 and 11");
   }
    return 0;
}
```

C program to check whether a character is alphabet or not

character i.e. it must lie between a-z or A-Z (Note: a and A both are different and have different ASCII values).

```
#include <stdio.h>
int main()
{   char ch;
     printf("Enter any character: ");
   scanf("%c", &ch);
     if((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z'))
   {printf("Character is an ALPHABET.\n");
   }
   else
   {printf("Character is NOT ALPHABET.\n");
   }
    return 0;
}
```

C program to check vowel or consonant

In C programming to check if a character is vowel, you just need to check if it is one of the characters a e i o u or A E I O U.
If it is not vowel and entered i.e. it is in range of a-z or A-Z then it is consonant.

```
#include <stdio.h>
int main()
{   char ch;
     printf("Enter any character: ");
    scanf("%c", &ch);
     if(ch=='a' || ch=='e' || ch=='i' || ch=='o' || ch=='u' || ch=='A' || ch=='E' || ch=='I' || ch=='O' || ch=='U')
    {printf("%c is VOWEL.\n", ch);
    }
    else if((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z'))
    {printf("%c is CONSONANT.\n", ch);
    }
     return 0;
}
```

Program to check uppercase or lowercase characters

```
#include <stdio.h>
int
main() {
char ch;
     printf("Enter any character: ");
    scanf("%c", &ch);
     if(ch >= 'A' && ch <= 'Z')
    {printf("%c is uppercase alphabet.\n", ch);
    }

    else if(ch >= 'a' && ch <= 'z')
    {printf("%c is lowercase alphabet.\n", ch);
    }
    else
    {printf("%c is not an alphabet.\n", ch);
    }
     return 0;
}
```

Program to print day of week

```
      #include <stdio.h>
int main()
{   int week;
     printf("Enter week number (1-7): ");
    scanf("%d", &week);
     if(week == 1)
    {printf("MONDAY\n");
    }
    else if(week == 2)
    {printf("TUESDAY\n");
    }
    else if(week == 3)
    {printf("WEDNESDAY\n");
    }
    else if(week == 4)
    {printf("THURSDAY\n");
    }
    else if(week == 5)
    {printf("FRIDAY\n");
    }
    else if(week == 6)
    {printf("SATURDAY\n");
    }
    else if(week == 7)
    {printf("SUNDAY\n");
    }
    else
    {printf("Invalid Input! Please enter week number between 1-7.\n");
    }
     return 0;
}
```

C program to enter month number and print number of days in month

```
#include <stdio.h>
int main()
{   int month;
 printf("Enter month number (1-12): ");
    scanf("%d", &month);
     if(month == 1)
    {printf("31 days");
    }
    else if(month == 2)
    {printf("28 or 29 days");
    }
    else if(month == 3)
    {printf("31 days");
    }
    else if(month == 4)
    {printf("30 days");
    }
    else if(month == 5)
    {printf("31 days");
    }
    else if(month == 6)
    {printf("30 days");
    }
    else if(month == 7)
    {printf("31 days");
    }
    else if(month == 8)
    {printf("31 days");
    }
    else if(month == 9)
    {printf("30 days");
    }
    else if(month == 10)
    {printf("31 days");
    }
    else if(month == 11)
    {printf("30 days");
    }
    else if(month == 12)
    {printf("31 days");
    }
    else
    {printf("Invalid input! Please enter month number between (1-12).\n");
    }
     return 0;
}
```

C program to find all roots of a quadratic equation

```
 #include <stdio.h>
#include <math.h> //Used for sqrt()
int main()
{   float a, b, c;
    float root1, root2, imaginary;
    float discriminant;
      printf("Enter values of a, b, c of quadratic equation (aX^2 + bX + c): ");
    scanf("%f%f%f", &a, &b, &c);
    discriminant = (b*b) - (4*a*c);
    if(discriminant > 0)
    {root1 = (-b + sqrt(discriminant)) / (2*a); root2
       = (-b - sqrt(discriminant)) / (2*a);
       printf("Two distinct and real roots exists: %.2f and %.2f\n", root1, root2);
    }
    else if(discriminant == 0)
    {root1 = root2 = -b / (2*a);
       printf("Two equal and real roots exists: %.2f and %.2f\n", root1, root2);
    }
    else if(discriminant < 0)
    {root1 = root2 = -b / (2*a); imaginary =
       sqrt(-discriminant) / (2*a);
       printf("Two distinct complex roots exists: %.2f + i%.2f and %.2f - i%.2f\n", root1, imaginary, root2, imaginary);
```

```
   }
   return 0;



}
```

C program to check whether triangle is valid or not if angles are given We know that a triangle is valid if and only if sum of all three angles is 180°. a triangle. #include <stdio.h>

```
int main()
{   int a, b, c, sum; //a, b, c are three angles of a triangle
     printf("Enter three angles of triangle: \n");
    scanf("%d%d%d", &a, &b, &c);
     sum = a + b + c;
    if(sum == 180)
    { printf("Triangle is valid.\n"); }
    else
    { printf("Triangle is not valid.\n"); }
    return 0;}
```

THEME :SWITCH –CASE

C program to check vowel or consonant using switch case

```
#include <stdio.h>
int main() {
char ch;
     printf("Enter any alphabet: ");
    scanf("%c", &ch);
     switch(ch)
    {case 'a': printf("VOWEL");
           break;
        case 'e': printf("VOWEL");
           break;
        case 'i': printf("VOWEL");
           break;
        case 'o': printf("VOWEL");
           break;
        case 'u': printf("VOWEL");
           break;
        case 'A': printf("VOWEL");
           break;
        case 'E': printf("VOWEL");
           break;
        case 'I': printf("VOWEL");
           break;
        case 'O': printf("VOWEL");
           break;
        case 'U': printf("VOWEL");
           break;
        default: printf("CONSONANT");
    }   return 0;
}
```

C program to enter week number and print day of week name using switch case

```
#include <stdio.h>
int main()
{   int week;
        printf("Enter week number(1-7): ");
    scanf("%d", &week);
        switch(week)
    {case 1: printf("MONDAY");
            break;
        case 2: printf("TUESDAY");
            break;
        case 3: printf("WEDNESDAY");
            break;
        case 4: printf("THURSDAY");
            break;
        case 5: printf("FRIDAY");
            break;
        case 6: printf("SATURDAY");
            break;
        case 7: printf("SUNDAY");
            break;
        default: printf("Invalid input! Please enter week number between 1-7");
    } return 0;
}
```

C program to create calculator using switch case and functions

```
#include <stdio.h>
int main()
{   char op;
    float num1, num2, result=0;
     printf("WELCOME TO SIMPLE CALCULATOR\n");
    printf("----------------------------\n");
    printf("Enter [number 1] [+ - * /] [number 2]\n");
     scanf("%f %c %f", &num1, &op, &num2);
     switch(op)
    {case '+': result = num1 + num2;
            break;
        case '-': result = num1 - num2;
            break;
        case '*': result = num1 * num2;
            break;
        case '/': result = num1 / num2;
            break;
        default: printf("Invalid operator");
            return 0;
    }
     printf("%.2f %c %.2f = %.2f\n", num1, op, num2, result);
     return 0;
}
```

C program to print all natural numbers from 1 to n

```
#include <stdio.h>
int main() {
int i, n;
   printf("Enter any number: ");
   scanf("%d", &n);
    printf("Natural numbers from 1 to %d : \n", n);
    for(i=1; i<=n; i++)
    {printf("%d\n", i);
    }
    return 0;
}
```

Program to print alphabets

```
#include <stdio.h>
int main() {
char ch;
    printf("Alphabets from a - z are: \n");
   for( ch='a' ; ch<='z' ; ch++)
    {printf("%c\n", ch);
    }
    return 0;
}
```

C program to print all even numbers from 1 to n

```
#include <stdio.h>
int main() {
int i, n;
  printf("Print all even numbers till: ");
   scanf("%d", &n);
    printf("All even numbers from 1 to %d are: \n", n);
    for(i=1; i<=n; i++)
    {
              if(i%2==0)
       {printf("%d\n", i);
       }
    }   return 0;
}
```

Program to display even numbers without using if statement

```
#include <stdio.h>
int main() {
int i, n;
  printf("Print all even numbers till: ");
   scanf("%d", &n);
    printf("All even numbers from 1 to %d are: \n", n);
   for(i=2; i<=n; i+=2)
    {printf("%d\n",i);
    }
   return 0;
}
```

C program to print all odd numbers from 1 to n

```
#include <stdio.h>
int main()
{   int i, n;
     printf("Print odd numbers till: ");
    scanf("%d", &n);
     printf("All odd numbers from 1 to %d are: \n", n);
     for(i=1; i<=n; i++)
    {if(i%2!=0)
        {printf("%d\n", i);
        }
    }
     return 0;
}
```

Program to display odd numbers without using if

```
#include <stdio.h>
 int  main()
{ int i, n;
     printf("Print odd numbers till: ");
    scanf("%d", &n);
     printf("All odd numbers from 1 to %d are: \n",
 n); for(i=1; i<=n; i+=2)
    {printf("%d\n", i);
    }
     return 0;}
```

C program to find sum of even numbers between 1 to n using loop

```
#include <stdio.h>
int main()
{ int i, n, sum=0; printf("Enter
   any number: "); scanf("%d",
   &n);
    for(i=2; i<=n; i+=2)
    {sum += i;
    }
    printf("\nSum of all even number between 1 to %d = %d", n, sum);
    return 0;
}
```

C program to print table of a given number

```
#include <stdio.h>
int main()
{   int i, num;
    printf("Enter number to print the table: ");
    scanf("%d", &num);
     for(i=1; i<=10; i++)
    {printf("%d * %d = %d\n", num, i, (num*i));
    }
     return 0;
}
```

C program to count number of digits in an integer

```
#include <stdio.h>
 int main()
{ long long num; int count = 0;
   printf("Enter any number: ");

   scanf("%lld", &num);
    while(num != 0)
   {count++;
      num /= 10; // num = num / 10
   }
    printf("Total digits: %d", count);
    return 0;
}
```

C program to find sum of digits of a number

logic of this program can be divided in three basic steps:

Find the last digit of number by performing modular division.
Add the last digit just found above to sum.
Remove the last digit from number by dividing the number by 10.

Repeat the above three steps till the number becomes 0 and you will be left with the sum of digits.

```
#include <stdio.h>
int main()
{    int num, sum=0;
   printf("Enter any number to find sum of its digit: ");
   scanf("%d", &num);
      while(num!=0)
   {sum += num % 10;
      num = num / 10;
   }
   printf("\nSum of digits = %d", sum);
   return 0;
}
```

C program to calculate product of digits of a number

logic of this program can be divided in three basic steps:

Find the last digit of number by performing modular division.
Multiply the last digit just found above to product.
Remove the last digit from number by dividing the number by 10.

Repeat the above three steps till the number becomes 0 and you will be left with the product of digits.

```
#include <stdio.h>
int main()
{    int n;
   long product=1;
    printf("Enter any number to calculate product of digit: ");
   scanf("%d", &n);
    while(n!=0)
   {product = product * (n % 10); n =
       n / 10;
   }
   printf("\nProduct of digits = %ld", product);
    return 0;
}
```

C program to find reverse of any number

The process of reversing involves four basic steps:

Mltiply the rev variable by 10.

Find the last digit of the given number.

Add the last digit just found to rev.

Divide the original number by 10 to eliminate the last digit, which is not needed anymore.

Repeat the above four steps till the original number becomes 0 and finally we will be left with the reversed number in rev variable.

```
#include <stdio.h>
 int main()
{    int n, rev = 0;
     /* Reads the number from user */
    printf("Enter any number to find reverse: ");
    scanf("%d", &n);
     /* Repeats the steps till n becomes 0 */
    while(n!=0)
    {
       /* Multiples rev by 10 and adds the last digit to it*/
       rev = (rev * 10) + (n % 10);
         /* Eliminates the last digit from num */
       n = n/10;
    }
     printf("Reverse = %d", rev);
     return 0;
}
```

C program to check whether a number is palindrome or not

After retrieving the reverse of number we just need to make a conditional check that whether the given number is equal to its reverse or not. If the given number and its reverse are same then the number is palindrome otherwise not.

```
#include <stdio.h>
 int main()
{    int n, num, rev = 0;
     /* Reads a number from user */
    printf("Enter any number to check palindrome: ");
    scanf("%d", &n);
     num = n; //Copies original value to num.
     /* Finds reverse of n and stores in rev */
    while(n!=0)
    {rev = (rev * 10) + (n %
       10); n = n/10;
    }
     /* Check if reverse is equal to original num or not */
    if(rev==num)
    {printf("%d is palindrome.", num);
    }
    else
    {printf("%d is not palindrome.", num);
    }
     return 0;
}
```

C program to display number in words Example:
Input: 1234
Output: One Two Three Four
Logic to convert any number in words is pretty simple and straightforward. To convert any number into words involves basic three steps:

Find the last digit of the number by performing modular division. If you don't know how check
Switch the appropriate word for the last digit just found.
Remove last digit from the number as it isn't useful anymore.

Repeat all three steps till the number becomes 0 and finally you will get the desired output. But one problem with the above explained
logic is that suppose the number is 1234 so, if you follow the above logic you will get output as "Four Three Two One" instead of
"One Two Three Four". To overcome this what we need to do is reverse the number before performing above mentioned three tasks
and you will get the correct output.

```
#include <stdio.h>
 int main()
{    int n, num = 0;
     /* Reads a number from user */
    printf("Enter any number to print in words: ");
    scanf("%d", &n);
     /* Finds reverse of the number */
    while(n!=0)
    {num = (num * 10) + (n %
       10); n /=10;
    }
     /*
       Finds last digit of the number and print corresponding digit in words
       till num becomes 0
     */
    while(num!=0)
    {
       switch(num%10)
       {
          case 0: printf("Zero ");
             break;
          case 1: printf("One ");
             break;
          case 2: printf("Two ");
             break;
          case 3: printf("Three ");
             break;
```

```
        case 4: printf("Four ");
            break;
        case 5: printf("Five ");
            break;
        case 6: printf("Six ");
            break;
        case 7: printf("Seven ");
            break;
        case 8: printf("Eight ");
            break;
        case 9: printf("Nine ");
            break;
    }

    num = num/10;
  }
  return 0;
}
```

C program to find power of a number using for loop #include <stdio.h>

```
 int main()
{ int base,
   exponent;
   long long
   power = 1;
   int i;
    /* Reads base and exponent from user */
   printf("Enter base: ");
   scanf("%d", &base);
   printf("Enter exponent: ");
   scanf("%d", &exponent);
    /* Multiples base, exponent times*/
   for(i=1; i<=exponent; i++)
   {
      power = power * base;
   }
    printf("\n%d ^ %d = %lld", base, exponent, power);
    return 0;
}
```

C program to find factorial of any number Example:
Input number: 5
Output
factorial: 120
Logic to find
factorial
Finding factorial is really easy you just need to know how to iterate over a loop. Finding factorial can be basically divided two steps:

Run a loop from 1 to n (where n is the number whose factorial is to be found).

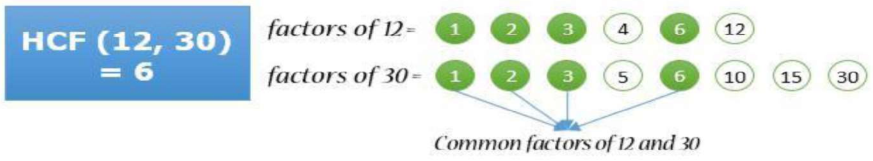
Multiply the current loop counter value with fact (where fact is a variable that stores factorial and is initially initialized with 1).

```
#include <stdio.h>
 int main()
{
   int i, num;
   long long fact=1;
    /* Reads a number from user */
   printf("Enter any number to calculate factorial: ");
   scanf("%d", &num);
    /* Runs a loop from 1 to n */
   for(i=1; i<=num; i++)
   {
      fact = fact * i;
   }
   printf("Factorial of %d = %lld", num, fact);
    return 0;
}
```

C **program to find GCD (HCF) of any two numbers** Example:HCF of 12 and 30 is 6

HCF(Highest Common Factor)

HCF is the greatest number that divides exactly two or more numbers. HCF is also known as GCD(Greatest Common Divisor) or GCF(Greatest Common Factor).
Example:



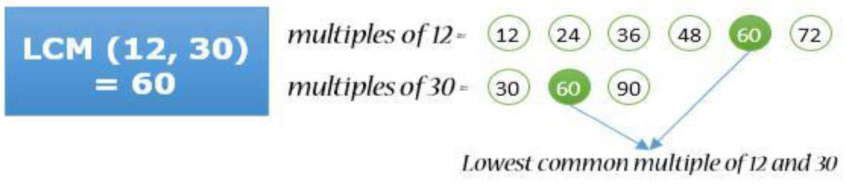
```
#include <stdio.h>
 int main()
{    int i, num1, num2, min, hcf=1;



    /*
      Reads two numbers from user
    */
   printf("Enter any two numbers to find HCF: ");
   scanf("%d %d", &num1, &num2);
    min = (num1<num2) ? num1 : num2;
    for(i=1; i<=min; i++)
   {
      /*
```

```
        If i is factor of both
      number */
     if(num1%i==0 && num2%i==0)
     {
        hcf = i;
     }
  }

  printf("HCF of %d and %d = %d\n", num1, num2, hcf);
  return 0;
}
```

C program to find LCM of any two numbers Example:
Input num1: 12
Input num2: 30
Output LCM: 60

LCM (Lowest Common Multiple)

LCM is the smallest positive integer that exactly divides two or more numbers.
For Example:



LCM of two numbers

```
#include <stdio.h>
 int main()
{
  int i, num1, num2, max, lcm=1;
   /*
     Reads two numbers from user
   */
  printf("Enter any two numbers to find LCM: ");
  scanf("%d %d", &num1, &num2);
   max = (num1>num2) ? num1 : num2;
   i = max;
      //Loop runs forever
  while(1)
  {
     /* If i is a multiple of both numbers */
     if(i%num1==0 && i%num2==0)
     {
        lcm = i;
        break;
        }

        i += max;
    }
     printf("\nLCM of %d and %d = %d\n", num1, num2, lcm);
     return 0;
}
```

C program to check whether a number is prime number or not
Example:
Enter any number: 17
Output: Number is Prime
number Prime numbers
Prime numbers are the positive integers greater than 1 that is only divisible by 1 and self. For example: 2, 3, 5, 7, 11 etc...

```
#include <stdio.h>
 int main()
{
    int i, n, flag;
     //Flag is used as notification. Initially we have supposed that the number is prime.
    flag = 1;

    /* Reads a number from user */
    printf("Enter any number to check prime: ");
    scanf("%d", &n);

    for(i=2; i<=n/2; i++)
    {
        /*
            If the number is divisible by any number
            other than 1 and self then it is not
         prime */
        if(n%i==0)
        {
            flag = 0;
            break;
        }
    }

    /*
       If flag contains 1 then it is prime
     */
    if(flag==1)
    {
        printf("\n%d is prime number", n);
    }
    else
    {
        printf("\n%d is not prime number", n);
    }
    return 0;
}
```

C program to check whether a number is Armstrong number or not.

Example:
Input: 371
Output: Armstrong number
Armstrong number
Armstrong number is a special number whose sum of cube of its digits is equal to the original number. For example: 371 is an
Armstrong number because
33 + 73 + 13 = 371
Logic to check armstrong number
Checking of armstrong number can be divided into basic three steps:

- Find last digit of the given number simply by performing modular division by 10.
- Find cube of the last digit just found and add it to a variable called sum.
- Repeat above two steps till the number becomes 0. And at the completion of the iteration check whether sum equals to original number or not. If it does then the given number is armstrong number otherwise not.

```
#include <stdio.h>
 int main()
{
    int n, num, lastDigit, sum = 0;

    /* Reads a number from user */
    printf("Enter any number to check Armstrong number: ");
    scanf("%d", &n);

    /* Copies the original value of n to num */
    num = n;

    /*
       Finds the sum of cubes of each digit of
     number */
    while(n != 0)
    {
        /* Finds last digit of number */
        lastDigit = n % 10;

        /* Finds cube of last digit and adds to sum */
        sum += (lastDigit * lastDigit * lastDigit);

        n = n / 10;
    }
    /* Checks if sum of cube of digits is equal to original value or not. */
    if(num == sum)
    {
        printf("\n%d is Armstrong number.", num);
    }
    else
    {
        printf("\n%d is not Armstrong number.", num);
    }
     return 0;
}
```

C program to check whether a number is perfect number or not
Example:
Input number: 6
Output: 6 is perfect number

Perfect number

A perfect number is a positive integer which is equal to the sum of its proper positive divisors. For example: 6 is the first perfect number
Proper divisors of 6 are 1, 2, 3
And 1+2+3 = 6. Hence 6 is a perfect number

```
#include <stdio.h>
 int main()
{    int i, num, sum = 0;

    /* Reads a number from user */
    printf("Enter any number to check perfect number: ");
    scanf("%d", &num);

    /* Finds the sum of all proper divisors */
    for(i=1; i<num; i++)
    {
        /* If i is a divisor of num */
        if(num%i==0)
        {
            sum += i;
        }
    }

    /* Checks whether the sum of proper divisors is equal to num */
    if(sum == num)
    {
        printf("\n%d is a Perfect number", num);
    }
    else
    {
        printf("\n%d is not a Perfect number", num);
    }
    return 0;
}
```

C program to print all prime numbers between 1 to n
Example:
Input lower limit: 1
Input upper limit: 10
Output prime numbers between 1-10: 2, 3, 5, 7
Prime number is a positive integer greater than 1 that is only divisible by 1 and self. Example: 2, 3 , 5, 7, 11 are the first five prime numbers

```
#include <stdio.h>
 int main()
{
    int i, j, n, isPrime; //isPrime is used as flag variable

    /* Reads upper limit to print prime */
    printf("Find prime numbers between 1 to : ");
    scanf("%d", &n);
    printf("\nAll prime numbers between 1 to %d are:\n", n);

    /* Finds all Prime numbers between 1 to n */
    for(i=2; i<=n; i++)

    {
        /* Assume that the current number is Prime */
        isPrime = 1;

        /* Check if the current number i is prime or not */
        for(j=2; j<=i/2; j++)
        {
            /*
                If i is divisible by any number other than 1 and self
                then it is not prime number
             */
            if(i%j==0)
            {
                isPrime = 0;
                break;
            }
        }

        /* If the number is prime then print */
        if(isPrime==1)
        {
            printf("%d is Prime number\n", i);
        }
    }
    return 0;
}
```

C program to find sum of all prime numbers between 1ton
Example:
Input: n=10
Output: Sum of all prime numbers between 1 to 10 = 2 + 3 + 5 + 7 = 17

```
#include <stdio.h>
 int main()
{
   int i, j, n, isPrime,
    sum=0; /*
      Reads a number from user
    */
   printf("Find sum of all prime between 1 to : ");
   scanf("%d", &n);

   /*
     Finds all prime numbers between 1 to n
    */
   for(i=2; i<=n; i++)
   {

      /*
         Checks if the current number i is Prime or
       not */
      isPrime = 1;
      for(j=2; j<=i/2 ;j++)
      {
         if(i%j==0)
         {
            isPrime = 0;
            break;

         }
      }

      /*
         If i is Prime then add to
       sum */
      if(isPrime==1)
      {
         sum += i;
      }
   }
    printf("Sum of all prime numbers between 1 to %d = %d", n,
    sum); return 0;
}
```

C program to find fibonacci series upto n terms
Example:
Input upper limit: 10
Output: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34

Fibonacci series

Fibonacci series is a series of numbers where the current number is the sum of previous two terms.
For Example: 0, 1, 1, 2, 3, 5, 8, 13, 21...

Fibonacci series algorithm
If you look to the given series very carefully you will find a specific pattern i.e. the current number is the sum of previous two numbers.
Fibonacci series : 0, 1, 1, 2, 3, 5, 8, 13, 21...
1 is the sum of 0 + 1
2 is the sum of 1 + 1
3 is the sum of 2 + 1
5 is the sum of 3 + 2
and so on....

Lets suppose the first number as a i.e. a = 0
Second number as b i.e. b = 1
And lets suppose c as our current number in Fibonacci series also initialize the value with 0 i.e. c = 0

Now Fibonacci series algorithm is simple and contains only four steps.
Step 1: Print the value of c.
Step 2: a = b.
Step 3: b = c.
Step 4: c = a + b.

```
#include <stdio.h>
int main()
{
   int a, b, c, i, n;
   printf("Enter value of n to print Fibonacci series : ");
   scanf("%d", &n);
   a = 0;
   b = 1;
   c = 0;

   for(i=1; i<=n; i++)
   {



      printf("%d, ", c);

      a=b;
      b=c;
      c=a+b;
   }

   return 0;
}
```

THEME-MODULAR PROGRAMMING

C program to find cube of a number using function

Example:
Input any number: 5
Output: 125 #include
<stdio.h> double
cube(double num)

```
{
   return (num * num * num);
}

int main()
{
   int num;
   double c;
      printf("Enter any number: ");
   scanf("%d", &num);
      c = cube(num);
    printf("Cube of %d is %.2f\n", num, c);
    return 0;
}
```

C program to check prime, Armstrong, perfect number using functions

Example:
Input any number: 11
Output: 11 is prime number
11 is not a armstrong number
11 is not a perfect number

```
#include <stdio.h>
 /* Function declarations */
int isPrime(int num);
int isArmstrong(int num);
int isPerfect(int num);

int main()
{
   int num;

   printf("Enter any number: ");
   scanf("%d", &num);

   //Calls the isPrime() functions
   if(isPrime(num))
   {
      printf("%d is Prime number.\n", num);
   }
   else
   {
      printf("%d is not Prime number.\n", num);
   }
```

```
    //Calls the isArmstrong() function
    if(isArmstrong(num))
    {
        printf("%d is Armstrong number.\n", num);
    }
    else
    {
        printf("%d is not Armstrong number.\n", num);
    }

    //Calls the isPerfect() function
    if(isPerfect(num))
    {
        printf("%d is Perfect number.\n", num);
    }
    else
    {
        printf("%d is not Perfect number.\n", num);
    }

    return 0;
}


/**
    Checks whether a number is prime or not. Returns 1 if the number is prime
    otherwise returns 0.
 */
int isPrime(int num)
{
    int i;

    for(i=2; i<=num/2; i++)
    {
        /*
            If the number is divisible by any number
            other than 1 and self then it is not
           prime */
        if(num%i == 0)
        {
            return 0;
        }
    }
       return 1;
}

 /**
```

```
    Checks whether a number is Armstrong number or not. Returns 1 if the number
    is Armstrong number otherwise returns 0.
 */
int isArmstrong(int num)
{
    int lastDigit, sum, n;
    sum = 0;
    n = num;

    while(n!=0)
    {
        /* Finds last digit of number */
        lastDigit = n % 10;


        /* Finds cube of last digit and adds to sum */
        sum += lastDigit * lastDigit * lastDigit;

        n = n/10;
    }

    return (num == sum);
}

/**
    Checks whether the number is perfect number or not. Returns 1 if the number
    is perfect otherwise returns 0.
 */
int isPerfect(int num)
{
    int i, sum, n;
    sum = 0;
    n = num;

    for(i=1; i<n; i++)
    {
        /* If i is a divisor of num */
        if(n%i == 0)
        {
            sum += i;
        }
    }
         return (num == sum);
}
```

C program to print all Armstrong numbers between given interval using function
Example:
Input lower limit: 1
Input upper limit: 1000
Output armstrong numbers: 1, 153, 370, 371,
407 Logic to print Armstrong numbers using
function
Basically printing Armstrong numbers in a given range includes basic three steps:
Find last digit of the given number.
Cube the last digit just found and add it to some variable sum. Repeat the above two steps till the number becomes 0.
Check if the sum equals to original number then the given number is Armstrong otherwise not.
Hence to code down all three steps into function we will define two functions namely isArmstrong and printArmstrong which will check for Armstrong number and print all Armstrong numbers in range respectively.

```
#include <stdio.h>

/* Function declarations */
int isArmstrong(int num);
void printArmstrong(int start, int end);
 int main()
{
   int start, end;

   /* Reads lowe and upper limit to of armstrong numbers */
   printf("Enter lower limit to print armstrong numbers: ");
   scanf("%d", &start);
   printf("Enter upper limit to print armstrong numbers: ");
   scanf("%d", &end);
   printf("All armstrong numbers between %d to %d are: \n", start, end);
   printArmstrong(start, end);

   return 0;
}

/**
   Checks whether the given number is armstrong number or not.
   Returns 1 if the number is armstrong otherwise 0.
 */
int isArmstrong(int num)
{
   int temp, lastDigit, sum;

   temp = num;
   sum = 0;

   /* Finds sum of cube of digits */
   while(temp != 0)
   {
      lastDigit = temp % 10;
      sum += lastDigit * lastDigit * lastDigit;
      temp /= 10;
   }




   /* Checks if sum of cube of digits equals
```

```
      to original number.
     */
    if(num == sum)
        return 1;
    else
        return 0;
}

/**
    Prints all armstrong numbers between start and
 end. */
void printArmstrong(int start, int end)
{
    /* Iterates from start to end and print the current number
       if it is armstrong
     */
    while(start <= end)
    {
        if(isArmstrong(start))
        {
            printf("%d, ", start);
        }

        start++;
    }
}
```

C program to find reverse of a number using recursion
Example:
Input number: 12345
Output reverse: 54321

Logic to find reverse of number using recursion
basic steps i.e.
Multiply the rev variable by 10.
Find the last digit of the given number.
Add the last digit just found to rev.
Divide the original number by 10 to eliminate the last digit, which is not needed anymore.
And we repeat the above four steps till the number becomes 0 and we are left with the reversed number in rev variable. Here also we will use the above four steps to find the reverse using recursive approach with the given base condition: reverse(0) = 0 {Base condition}
reverse(n) = (n%10 * pow(10, digits)) + reverse(n/10) {where digit is the total number of digits in number}

```
#include <stdio.h>
#include <math.h>
  /* Fuction declaration */
int reverse(int num);

int main()
{
    int num, rev;

    /* Reads number from user */
    printf("Enter any number: ");
    scanf("%d", &num);

    /*Calls the function to reverse number */
    rev = reverse(num);
    printf("Reverse of %d = %d", num, rev);

    return 0;
}

/**
    Recursive function to find reverse of any
 number */
int reverse(int num)
{
    int digit;

    //Base condition
    if(num==0)
        return 0;

    //Finds total number of digits
    digit = (int)log10(num);

    return ((num%10 * pow(10, digit)) + reverse(num/10));
}
```

C program to find sum of even or odd number from 1 to n using recursion
Example:
Input range: 100
Output: Sum of even numbers between 1 to 100 =
2550 Logic to find sum of even or odd numbers
recursively Below is the recursive conditions for
sum of even numbers: sum(0) = 0 {Base condition}
sum(n) = n + sum(n-2) {Where n is always even}

Recursive condition for sum of all odd numbers using recursion would be:
sum(1) = 1 {Base condition}
sum(n) = n + sum(n-2) {Where n is always odd}

Here in the below program what I have tried is to, embed the logic of both even and odd recursive condition in a single function.

```
#include <stdio.h>
 int sum(int num);
 int main()
{
    int num, lastEven, lastOdd;

    //Reads the upper limit to find sum from user
    printf("Enter the upper limit to find sum: ");
    scanf("%d", &num);

    lastEven = (num & 1) ? num-1 : num;   //Finds last even number in range
    lastOdd  = (num & 1) ? num : num-1; //Finds last odd number in range

    printf("Sum of all even numbers from 1 to %d = %d\n", num,
    sum(lastEven)); printf("Sum of all odd numbers from 1 to %d = %d\n",
    num, sum(lastOdd));

    return 0;
}
```

```
/**
    Finds the sum of all even or odd numbers
 recursively. */
int sum(int num)
{
    //Base condition
    if(num <= 0)
        return 0;

    /* Recursively calls sum() to add previous even or odd
    number */ return (num + sum(num-2));
}
```

C program to find factorial of a number using recursion

Example:
Input any number: 5
Output: 120
The factorial of any number can be recursively given
by: fact(0) = 1 {Base case}
fact(num) = num * fact(num-1) {Numbers greater than 1}

```
#include <stdio.h>

 long long fact(int num);

 int main()
{
    int num;
    long long factorial;

    /*
       Reads an integer from user
     */
    printf("Enter any number: ");
    scanf("%d", &num);
    factorial = fact(num); //Calls factorial function

    printf("Factorial of %d is %lld\n", num, factorial);

    return 0;
}

/**
    Function to compute and return factorial of any number
 recursively. */
long long fact(int num)
{
    if(num == 0) //Base condition
        return 1;
    else
        return num * fact(num-1);
}
```

C program to generate nth fibonacci term using recursion
Example:
Input any number: 10
Output 10th fibonacci term: 55
Logic to find nth Fibonacci term
base condition of the recursion which is explained below.

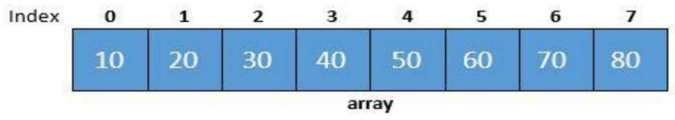
fibo(0) = 0 {Base condition}
fibo(1) = 1 {Base condition}
fibo(num) = fibo(num-1) + fibo(num+2)

```
#include <stdio.h>

//Function declaration
long long fibo(int num);
 int main()
{
    int num;
    long long fibonacci;

    //Reads number from user to find nth fibonacci term
    printf("Enter any number to find nth fiboacci term: ");
    scanf("%d", &num);

    fibonacci = fibo(num);

    printf("%dth fibonacci term is %lld", num, fibonacci);

    return 0;
}

/**


    Recursive function to find nth Fibonacci
 term */
long long fibo(int num)
{
    if(num == 0) //Base condition
        return 0;
    else if(num == 1) //Base condition
        return 1;
    else
        return fibo(num-1) + fibo(num-2); //Recursively calls fibo() to find nth fibonacci term.
}
```

THEME:ARRAY

C program to input and print n elements in an array Example:
Input size: 10
Input
elements:
1 2 3 4 5
6 7 8 9
10

Output: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Logic to input and display array elements
Array uses an index based mechanism for fast and easy accessing of elements. Array index starts from 0 to N - 1 (where N is the total number of elements in the array). Below diagram explains the arrangement of elements in an array.



Here, we access the array elements in following manner
array[0] = 10
array[1] = 20
array[2] = 30
...
array[7] = 70
You can replace the index value within the big brackets [ ] with an integer variable. Like the above statements can be replaced by array[i] = some_value; (where i is an integer variable). Let's demonstrate this through a program. Program to input and display array elements

```
#include <stdio.h>
#define MAX_SIZE 1000 //Maximum size of the array

int main()
{
    int arr[MAX_SIZE]; //Declares an array of MAX_SIZE
    int i, N;




    /*
       Reads size and elements
     in array */
    printf("Enter size of array: ");
    scanf("%d", &N);
    printf("Enter %d elements in the array : ", N);
    for(i=0; i<N; i++)
    {
        scanf("%d", &arr[i]);
    }

    /*
       Prints all elements
     of array */
    printf("\nElements in array are: ");
    for(i=0; i<N; i++)
    {


       printf("%d, ", arr[i]);
    }
     return 0;
}
```

C program to print all negative elements in an array
Example:
Input
array
: -1 -
10
100 5
61 -2
-23 8
-90
51

Output: -1, -10, -2, -23, -90
Logic to display negative elements in array
Displaying negative, positive, prime, even, odd or any special number doesn't requires special skills. You only need to know how to display array elements and how to check that special number.

Now, considering our case where we need to display only negative elements of array. Below is a descriptive logic to display all negative elements of array.

Input elements in array.
Check if the current element is negative or not. If the current array element is negative then print the current element otherwise skip. To check you simple use if(array[i] < 0) (where i is the current array index).
Repeat step 2 till the last element of array.

Let's now implement this.
Program to print negative elements in array

```
#include <stdio.h>

#define MAX_SIZE 100

int main()
{
    int arr[MAX_SIZE]; //Declares an array of MAX_SIZE
    int i, n;

    /*
       Reads size and elements of
     array */
    printf("Enter size of the array : ");
    scanf("%d", &n);

    printf("Enter elements in array : ");
    for(i=0; i<n; i++)
    {
       scanf("%d", &arr[i]);
    }

    printf("\nAll negative elements in array are : ");
    for(i=0; i<n; i++)
    {
        //Print current element if it is negative
        if(arr[i] < 0)
        {
            printf("%d\t", arr[i]);
        }
    }

    return 0;
}
```

C program to find sum of all elements of an array
Example:
Input elements: 10, 20, 30, 40, 50
Sum of all elements = 150

```
#include <stdio.h>
#define MAX_SIZE 100

int main()
{
    int arr[MAX_SIZE];
    int i, n, sum=0;

    /*
     Reads size and elements in array from user
     */
    printf("Enter size of the array: ");
    scanf("%d", &n);
    printf("Enter %d elements in the array: ", n);
    for(i=0; i<n; i++)
    {
        scanf("%d", &arr[i]);
    }

    /*
     Adds each element of array to sum
     */
    for(i=0; i<n; i++)
    {
        sum = sum + arr[i];
    }

    printf("Sum of all elements of array = %d", sum);

    return 0;
}
```

C program to find maximum and minimum element in array
Example: If the elements of the array are: 10, 50, 12, 16, 2
Maximum = 50
Minimum = 2

Logic to find maximum/minimum in
array Step 1: Read element in array.
Step 2: Let's suppose the first element of array as maximum. Set max=array[0]
Step 3: Set i=0
Step 4: If array[i] > max then Set max=array[i]
Step 5: Increment i by 1. Set i=i+1
Step 6: Repeat Step 4-5 till i<size (Where size is the size of array).

```
#include <stdio.h>

int main()
{
    int arr[100];
    int i, max, min, size;

    /*
       Reads size array and elements in the array
     */
    printf("Enter size of the array: ");
    scanf("%d", &size);
    printf("Enter elements in the array: ");
    for(i=0; i<size; i++)
    {
        scanf("%d", &arr[i]);
    }

    /* Supposes the first element as maximum and minimum */
    max = arr[0];
    min = arr[0];

    /*
       Finds maximum and minimum in all array elements.
     */
    for(i=1; i<size; i++)
    {
        /* If current element of array is greater than max */
        if(arr[i]>max)
        {
            max = arr[i];
        }

        /* If current element of array is smaller than min */
        if(arr[i]<min)
        {
            min = arr[i];
        }
    }




    /*
       Prints the maximum and minimum element
     */
    printf("Maximum element = %d\n", max);
    printf("Minimum element = %d", min);

    return 0;
}
```

C program to count even and odd elements in an array
Example:
Input array: 1 2 3 4 5 6 7 8 9
Output:
Total even elements: 4
Total odd elements: 5
Logic to count total even/odd elements in array
Below is a basic logic to count total even or odd elements in an array.
Run a loop from the starting to end index of the array 0-N (Where N is the total number of elements in array).
Check the current element of array. If it is even then, increment the even count by 1. Otherwise increment the odd count.
Print the values of even and odd after the termination of loop.
And you are done.

```
#include <stdio.h>

#define MAX_SIZE 100 //Maximum size of the array

int main()
{
   int arr[MAX_SIZE];
   int i, N, even, odd;

   /*
     Reads size and elements in
    array */
   printf("Enter size of the array: ");
   scanf("%d", &N);
   printf("Enter %d elements in array: ", N);
   for(i=0; i<N; i++)
   {
      scanf("%d", &arr[i]);
   }

   /* Assuming that there are 0 even and odd elements */
   even = 0;
   odd = 0;

   for(i=0; i<N; i++)
   {
      /* If the current element of array is even then increment even count */
      if(arr[i]%2 == 0)
      {
         even++;
      }
      else
      {
         odd++;
      }
   }

   printf("Total even elements: %d\n", even);
   printf("Total odd elements: %d\n", odd);

   return 0;
}
```

C program to print all unique element in an array

Example: If elements of the array are: 1, 2, 3, 5, 1, 5, 20, 2, 12, 10 All unique elements in the array are: 3, 20, 12, 10

```
#include <stdio.h>

int main()
{
   int arr[100], size, isUnique;
   int i, j, k; //Used for loops
/*
   Reads size of the array
 */
printf("Enter size of array: ");
scanf("%d", &size);

/*
   Reads elements in array
 */
printf("Enter elements in array: ");
for(i=0; i<size; i++)
{
   scanf("%d", &arr[i]);
}

/*
   Removes all duplicate elements from array
 */
for(i=0; i<size; i++)
{



   /* Assumes that the current array element is unique */
   isUnique = 1;

   for(j=i+1; j<size; j++)
   {









       /*
          If any duplicate element is found
        */
       if(arr[i]==arr[j])
       {
          /* Remove the duplicate element */
          for(k=j; k<size-1; k++)
          {
             arr[k] = arr[k+1];
          }
```

```
                size--;
                j--;
                isUnique = 0;
            }
        }

        /*
          If array element is not unique
          then also remove the current element */
        if(isUnique != 1)
        {
            for(j=i; j<size-1; j++)
            {
                arr[j] = arr[j+1];
            }

            size--;
            i--;
        }
    }


    /*
      Prints all unique elements in array
     */
    printf("\nAll unique elements in the array are: ");
    for(i=0; i<size; i++)
    {
        printf("%d\t", arr[i]);
    }

    return 0;
}
```

C program count total number of duplicate elements in an array
Example:
If the elements of array are: 1, 10, 20, 1, 25, 1, 10, 30, 25,
1 Total number of duplicate elements = 5
Duplicate elements in array are : 1, 1, 1, 10, 25
Algorithm to count total duplicate elements in
array Step 1: Set count = 0
Step 2: Set i = 0
Step 3: Set j = i+1
Step 4: Check if array[i] == array[j] then increment count by 1 i.e. count = count + 1
Step 5: Increment j by 1 i.e. j = j + 1 and repeat Step 4-5 till j<n (Where n is the total number of elements in array.
Step 6: Increment i by 1 i.e. i = i + 1 and repeat Step 3-5 till i<n.

```
#include <stdio.h>

int main()
{
    int arr[100];
    int i,j, n, count = 0;

    /*
      Reads size and elements of array
     */
    printf("Enter size of the array : ");
    scanf("%d", &n);

    printf("Enter elements in array : ");
    for(i=0; i<n; i++)
    {
        scanf("%d", &arr[i]);
    }

    /*
      Finds all duplicate elements in array
     */
    for(i=0; i<n; i++)
    {
        for(j=i+1; j<n; j++)
        {
            /* If duplicate found then increment count by 1 */
            if(arr[i]==arr[j])
            {
                count++;
                break;
            }
        }
    }

    printf("\nTotal number of duplicate elements found in array = %d", count);

    return 0;
}
```

C program to count frequency of each element in an array

Example: If elements of array are: 5, 10, 2, 5, 50, 5, 10, 1, 2, 2
Frequency of 5 = 3
Frequency of 10 = 2
Frequency of 2 = 3
Frequency of 50 = 1
Frequency of 1 = 1

```
#include <stdio.h>

int main()
{
    int arr[100], freq[100];
    int size, i, j, count;

    /*
       Read size of array and elements in array
     */
    printf("Enter size of array: ");
    scanf("%d", &size);

    printf("Enter elements in array: ");
    for(i=0; i<size; i++)
    {
        scanf("%d", &arr[i]);
        freq[i] = -1;
    }
    *
      Counts frequency of each element
     */
    for(i=0; i<size; i++)
    {
        count = 1;
        for(j=i+1; j<size; j++)
        {
            if(arr[i]==arr[j])
            {
                count++;
                freq[j] = 0;
            }
        }

        if(freq[i]!=0)
        {
            freq[i] = count;
        }
    }


    /*
      Prints frequency of each element
     */
    printf("\nFrequency of all elements of array : \n");
    for(i=0; i<size; i++)
    {
        if(freq[i]!=0)
        {
            printf("%d occurs %d times\n", arr[i], freq[i]);
        }
    }

    return 0;
}
```

C program to merge two sorted array
Example: If elements of array A = 1, 4, 6, 9, 15
Elements of array B = 2, 5, 8, 10
Merged array in ascending order = 1, 2, 4, 5, 6, 8, 9, 10,
15 #include <stdio.h>

```
#define MAX_SIZE 100 //Maximum size of the array

int main()
{
    int arr1[MAX_SIZE], arr2[MAX_SIZE], arr3[MAX_SIZE];
    int size1, size2, size3;
    int i, j, k;

    /*
       Read size of first array and elements in first array
     */
    printf("Enter the size of first array : ");
    scanf("%d", &size1);
    printf("Enter elements in first array : ");
    for(i=0; i<size1; i++)
    {
        scanf("%d", &arr1[i]);
    }

    /*
       Reads size of second array and elements in second array
     */
    printf("\nEnter the size of second array : ");
    scanf("%d", &size2);
    printf("Enter elements in second array : ");
    for(i=0; i<size2; i++)
    {
        scanf("%d", &arr2[i]);
    }

    /* size of merged array is size_of_first_array + size_of_second_array
    */ size3 = size1 + size2;

    /*
       Merge two array in ascending order
     */
    for(i=0, j=0, k=0; i<size3; i++)
    {
        /*
            If all elements of one array
            is merged to final array
         */
        if(j >= size1 || k >= size2)
        {
            break;
        }

         if(arr1[j] < arr2[k])
        {
            arr3[i] = arr1[j];
            j++;
        }
        else
        {
            arr3[i] = arr2[k];
            k++;
        }
    }

    /*
       Merge the remaining elements of array
     */
    while(j < size1)
    {
        arr3[i] = arr1[j];
        i++;
        j++;
    }
    while(k < size2)
    {
        arr3[i] = arr2[k];
        i++;
        k++;
    }

    /*
       Prints the merged array
     */
    printf("\nArray merged in ascending order : ");
    for(i=0; i<size3; i++)
    {
        printf("%d\t", arr3[i]);
    }

    return 0;
}
```

C program to find reverse of an array
Example:

If the elements of the array are: 10, 5, 16, 35, 500
Then its reverse would be: 500, 35, 16, 5, 10
That is if
array[0] = 10
array[1] = 5
array[2] = 16
array[3] = 35
array[4] = 500

Then after reversing array elements should be
array[0] = 500
array[1] = 35
array[2] = 16
array[3] = 5
array[4] = 10
Algorithm:
Basic algorithm for reversing any array.
Step 1: Take two array say A and B. A will hold the original values and B will hold the reversed values.
Step 2: Read elements in array A.
Step 3: Set i=size - 1, j=0 (Where size is the size of array).
Step 4: Set B[j] = A[i]
Step 5: Set j = j + 1 and i = i - 1.
Step 6: Repeat Step 4-5 till i>=0.
Step 7: Print the reversed array in B.
Program:

```
#include <stdio.h>

int main()
{
   int arr[100], reverse[100];
   int size, i, j;

   /*
     Reads the size of array and elements in array
    */
   printf("Enter size of the array: ");
   scanf("%d", &size);
   printf("Enter elements in array: ");
   for(i=0; i<size; i++)
   {
     scanf("%d", &arr[i]);
   }











   /*
```

```
        Reverse the array
     */
    j=0;
    for(i=size-1; i>=0; i--)
    {
        reverse[j] = arr[i];
        j++;
    }

    /*
        Prints the reversed array
     */
    printf("\nReversed array : ");
    for(i=0; i<size; i++)
    {
        printf("%d\t", reverse[i]);
    }

    return 0;
}
```

C program to search an element in the array

Example: If the elements of array are: 10, 12, 20, 25, 13, 10, 9, 40, 60, 5

Element to search is: 25
Output: Element exists
Algorithm:

```
Step 1: Read elements in array A.
Step 2: Read element to be searched in num.
Step 3: Set i=0, flag=0. We have initially supposed that the number doesn't exists in the array hence we set flag=0.
Step 4: If A[i]==num then set flag=1 and print "Element found" and goto Step 6.
Step 5: Set i=i+1 and repeat Step 4 till i<size (Where size is the size of array).
Step 6: If the value of flag=0 then print "Element not found".
```

Program:

```
#include <stdio.h>

int main()
{
   int arr[100];
   int size, i, num, flag;

   /*
     Read size of array and elements in array
    */
   printf("Enter size of array: ");
   scanf("%d", &size);

   printf("Enter elements in array: ");
   for(i=0; i<size; i++)
   {
      scanf("%d", &arr[i]);
   }

   printf("\nEnter the element to search within the array: ");
   scanf("%d", &num);

   /* Supposes that element is not in the array */
   flag = 0;
   for(i=0; i<size; i++)
   {
      /*
         If element is found in the
       array */
      if(arr[i]==num)
      {
         flag = 1;
         printf("\n%d is found at position %d", num, i+1);
         break;
      }
   }

   /*
    * If element is not found in array
```

```
     */
    if(flag==0)
    {
        printf("\n%d is not found in the array", num);
    }

    return 0;
}
```

C program to sort an array in ascending order
Example:
If the elements of array are: 20, 2, 10, 6, 52, 31, 0, 45, 79, 40
Array sorted in ascending order: 0, 2, 6, 10, 20, 31, 40, 45, 52, 79
Algorithm:
Here we will use basic algorithm to sort arrays in ascending order:

```
Step 1: Read elements in array.
Step 2: Set i=0
Step 3: Set j=i+1
Step 4: If array[i] > array[j] then swap value of array[i] and array[j].
Step 5: Set j=j+1
Step 6: Repeat Step 4-5 till j<n (Where n is the size of the array)
Step 7: Set i=i+1
Step 8: Repeat Step 3-7 till i<n
```

Program:

```
#include <stdio.h>

int main()
{
    int arr[100];
    int size, i, j, temp;

    /*
      Read size of array and elements in array
     */
    printf("Enter size of array: ");
    scanf("%d", &size);

    printf("Enter elements in array: ");
    for(i=0; i<size; i++)
    {
        scanf("%d", &arr[i]);
    }
    /*
      Array sorting code
     */
    for(i=0; i<size; i++)
    {
       for(j=i+1; j<size; j++)
       {
          /*
             If there is a smaller element towards right of the array then swap
           it. */
          if(arr[j] < arr[i])
          {
             temp = arr[i];
             arr[i] = arr[j];
             arr[j] = temp;
          }
       }
    }

    /*
      Prints the sorted array
     */
    printf("\nElements of array in sorted ascending order: ");
    for(i=0; i<size; i++)
    {
       printf("%d\t", arr[i]);
    }

    return 0;
}
```

C program to sort an array in descending order
Example:
If the elements of array are: 20, 2, 10, 6, 52, 31, 0, 45, 79, 40

Array sorted in descending order: 79, 52, 45, 40, 31, 20, 10, 6, 2, 0

Algorithm:
Here we will use basic algorithm to sort arrays in descending order:

```
Step 1: Read elements in array.
Step 2: Set i=0
Step 3: Set j=i+1
Step 4: If array[i] < array[j] then swap value of array[i] and array[j].
Step 5: Set j=j+1
Step 6: Repeat Step 4-5 till j<n (Where n is the size of the array)
Step 7: Set i=i+1
Step 8: Repeat Step 3-7 till i<n
```

Program:

```
#include <stdio.h>

int main()
{
    int arr[100];
    int size, i, j, temp;

    /*
       Read size of array and elements in array
     */
    printf("Enter size of array: ");
    scanf("%d", &size);

    printf("Enter elements in array: ");
    for(i=0; i<size; i++)
    {
        scanf("%d", &arr[i]);
    }

    /*
       Array sorting code
     */
    for(i=0; i<size; i++)
    {
        for(j=i+1; j<size; j++)
        {
          /*
             If there is a smaller element towards left of the array then swap it. */
          if(arr[i] < arr[j])
          {
             temp = arr[i];
             arr[i] = arr[j];
             arr[j] = temp;
          }
        }
    }

    /*
       Prints the sorted array */
    printf("\nElements of array in sorted descending order: ");
    for(i=0; i<size; i++)
    {
       printf("%d\t", arr[i]);
    }

    return 0;
}
```

C program to add two matrices Example:
If matrix 1: 1
2 3 4 5 6 7 8 9

And matrix 2:
9 8 7
6 5 4
3 2 1

Sum of both matrix =
10 10 10
10 10 10
10 10 10

Matrix Addition

Matrix addition is a simple process. Addition of two matrices can be done only and only if both matrices are of same size.

Matrix addition is done element wise (entry wise) i.e. Sum of two matrices A and B of size mXn is defined by (A + B) = Aij + Bij (Where $1 \leq i \leq m$ and $1 \leq j \leq n$)

Program:
```
#include <stdio.h>
```

```
int main()
{
    int A[3][3], B[3][3], C[3][3];
    int row, col;

    /*
      Reads elements in first matrix */
    printf("Enter elements in matrix A of size 3x3: \n");
    for(row=0; row<3; row++)
    {
        for(col=0; col<3; col++)
        {
            scanf("%d", &A[row][col]);
        }
    }

    /*
      Reads elements in second matrix */
    printf("\nEnter elements in matrix B of size 3x3: \n");
    for(row=0; row<3; row++)
    {
        for(col=0; col<3; col++)
        {
            scanf("%d", &B[row][col]);
        }
    }

    /*
      Adds both matrices A and B entry wise or element wise
      And stores result in matrix C
     */
    for(row=0; row<3; row++)
    {
        for(col=0; col<3; col++)
        {
            /* Cij = Aij + Bij */
            C[row][col] = A[row][col] + B[row][col];
        }
    }

    /*
     * Prints the sum of both matrices A and B

     */
    printf("\nSum of matrices A+B = \n");
    for(row=0; row<3; row++)
    {
        for(col=0; col<3; col++)
        {
            printf("%d ", C[row][col]);
        }
        printf("\n");
    }

    return 0;
}
```

C program to subtract two matrices
Example:
If matrix
1: 1 2 3 4
5 6 7 8 9

And matrix 2:
9 8 7
6 5 4
3 2 1

Difference of both matrices =
-8 -6 -4
-2 0 2
 4 6 8

Matrix Subtraction:

Matrix subtraction is a simple and easy process. Elements of two matrices can only be subtracted if and only if both matrices are of same size.

Matrix subtraction is done element wise (entry wise) i.e. Difference of two matrices A and B of size mXn is defined by (A - B) = Aij - Bij (Where 1 ≤ i ≤ m and 1 ≤ j ≤ n)

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} - \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1-9 & 2-8 & 3-7 \\ 4-6 & 5-5 & 6-4 \\ 7-3 & 8-2 & 9-1 \end{bmatrix}$$

$$= \begin{bmatrix} -8 & -6 & -4 \\ -2 & 0 & 2 \\ 4 & 6 & 8 \end{bmatrix}$$

Program:

```
#include <stdio.h>

int main()
{
    int A[3][3], B[3][3], C[3][3];
    int row, col;

    /*
      Reads elements in first matrix */
    printf("Enter elements in matrix A of size 3x3: \n");
    for(row=0; row<3; row++)
    {
        for(col=0; col<3; col++)
        {
            scanf("%d", &A[row][col]);
        }
    }

    /*
      Reads elements in second matrix */
    printf("\nEnter elements in matrix B of size 3x3: \n");
    for(row=0; row<3; row++)
    {
        for(col=0; col<3; col++)
        {
            scanf("%d", &B[row][col]);
        }
    }

    /*
      Subtracts both matrices and stores the result in matrix C */
    for(row=0; row<3; row++)
    {
        for(col=0; col<3; col++)
        {
            C[row][col] = A[row][col] - B[row][col];
        }
    }

    /*
      Prints the difference of both matrices A and B */
    printf("\nDifference of two matrices A-B = \n");
    for(row=0; row<3; row++)
    {
        for(col=0; col<3; col++)
        {
            printf("%d ", C[row][col]);
        }
        printf("\n");
    }

    return 0;
}
```

C program to multiply two matrices

Example:
If matrix 1 = 1
2 3
4 5 6
7 8 9

And matrix 2 =
9 8 7
6 5 4
3 2 1

Product of both matrices =
30 24 18
84 69 54
138 114 90
Required knowledge:
Basic C programming, For loop, Array, Matrix

Matrix Multiplication

Multiplication of matrices is different from simple Scalar multiplication of matrix in C. Two matrices can be multiplied only and only if number of columns in the first matrix is same as number of rows in second matrix. Multiplication of two matrices can be defined as

$$\begin{bmatrix} a\ b \\ c\ d \end{bmatrix} \begin{bmatrix} e\ f \\ g\ h \end{bmatrix} = \begin{bmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{bmatrix}$$

```
#include <stdio.h>

int main()
{
   int A[3][3], B[3][3], C[3][3];
   int row, col, i, sum;

   /*
      Reads elements in first matrix from
    user */
   printf("Enter elements in matrix A of size 3x3: \n");
   for(row=0; row<3; row++)
   {
      for(col=0; col<3; col++)
      {
         scanf("%d", &A[row][col]);
      }
   }




   /*
      Reads elements in second matrix from
    user */
   printf("\nEnter elements in matrix B of size 3x3: \n");
   for(row=0; row<3; row++)
```

```
    {
        for(col=0; col<3; col++)
        {
            scanf("%d", &B[row][col]);
        }
    }


    /*
       Multiplies both matrices
     A*B */
    for(row=0; row<3; row++)
    {
        for(col=0; col<3; col++)
        {
            sum = 0;
            /*
                Multiplies row of first matrix to column of second matrix
                And stores the sum of product of elements in sum.
             */
            for(i=0; i<3; i++)
            {
                sum += A[row][i] * B[i][col];
            }

            C[row][col] = sum;
        }
    }

    /*
       Prints the product of
     matrices */
    printf("\nProduct of Matrix A * B = \n");
    for(row=0; row<3; row++)
    {
        for(col=0; col<3; col++)
        {
            printf("%d ", C[row][col]);
        }
        printf("\n");
    }

    return 0;
}
```

C program to check whether two matrices are equal or not
Example:

If
matrix 1
= 1 2 3
4 5 6 7 8
9

And matrix 2 =
1 2 3
4 5 6
7 8 9

Output: Both matrices are equal.

Equality of matrix

Checking whether two matrices are equal or not is very simple and fundamental. Two matrices are said to be equal if and only if they are of same size and they have equal corresponding entries.

Equality of two matrices A and B can be defined as

$A_{ij} = B_{ij}$ (Where $1 \le i \le m$ and $1 \le j \le n$).

$$\underbrace{\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}}_{\text{Matrix A}} = \underbrace{\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}}_{\text{Matrix B}}$$

Both the matrices are of same dimension and also their corresponding elements are equal. Hence both Matrix A and Matrix B are equal.
Program:

```
#include <stdio.h>

int main()
{
    int A[3][3], B[3][3];
    int row, col, isEqual;





    /*
      Reads elements in first matrix from user
     */
    printf("Enter elements in matrix A of size 3x3: \n");
```

```
for(row=0; row<3; row++)
{
    for(col=0; col<3; col++)
    {
        scanf("%d", &A[row][col]);
    }
}

/*
   Reads elements in second matrix from user
 */
printf("\nEnter elements in matrix B of size 3x3: \n");
for(row=0; row<3; row++)
{
    for(col=0; col<3; col++)
    {
        scanf("%d", &B[row][col]);
    }
}

/* Assumes that the matrices are equal */
isEqual = 1;

for(row=0; row<3; row++)
{
    for(col=0; col<3; col++)
    {
        /*
            If the corresponding entries of matrices are not
         equal */
        if(A[row][col] != B[row][col])
        {
            isEqual = 0;
            break;
        }
    }
}
```

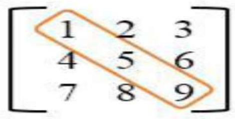
```
    /*
      Checks the value of isEqual
      As per our assumption if isEqual contains 1 means both are equal
      If it contains 0 means both are not equal
     */
    if(isEqual == 1)
    {
       printf("\nMatrix A is equal to Matrix B");
    }
    else
    {
       printf("\nMatrix A is not equal to Matrix B");
    }

    return 0;
}
```

C program to find sum of main diagonal elements of a matrix Example:
If the array elements are: 1 2 3 4
5 6 7 8 9

Output: Sum of main diagonal elements = 15

Main diagonal of matrix

Main diagonal of a matrix A is a collection of elements Aij Such that i = j.



Program:

```
#include <stdio.h>

int main()
{
    int A[3][3];
    int row, col, sum = 0;




    /*
      Reads elements in matrix from user */
    printf("Enter elements in matrix of size 3x3: \n");
    for(row=0; row<3; row++)
    {
        for(col=0; col<3; col++)
        {
            scanf("%d", &A[row][col]);
        }
    }

    /*
     * Finds the sum of main diagonal elements


     */
    for(row=0; row<3; row++)
    {
        sum = sum + A[row][row];
    }

    printf("\nSum of main diagonal elements = %d", sum);

    return 0;
}
```

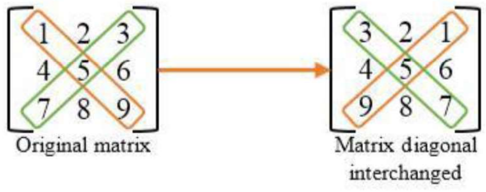
C program to interchange diagonals of a matrix Example: If elements of matrix are:

1 2 3
4 5 6
7 8 9

Matrix after interchanging its diagonal:

3 2 1
4 5 6
9 8 7

Algorithm:



Logic to interchange diagonals of matrix A:
Step 1: temp = A[row][col]
Step 2: A[row][col] = A[row][(N-col)-1] (Where N is the size of array).
Step 3: A[row][(N-col)-1] = temp.

Program:
```
#include <stdio.h>
#define MAX_ROWS 3
#define MAX_COLS 5

int main()
{
    int A[MAX_ROWS][MAX_COLS];
    int row, col, square, temp;

    /*
      Reads elements in matrix from user */
    printf("Enter elements in matrix of size %dx%d: \n", MAX_ROWS, MAX_COLS);
    for(row=0; row < MAX_ROWS; row++)
    {
        for(col=0; col < MAX_COLS; col++)
        {
            scanf("%d", &A[row][col]);
        }
    }

    square = (MAX_ROWS < MAX_COLS) ? MAX_ROWS : MAX_COLS;



    /*
      Interchanges diagonal of the matrix
     */
    for(row=0; row < square; row++)
    {
        col = row;

        temp = A[row][col];
        A[row][col] = A[row][(square - col)-1];
        A[row][(square - col)-1] = temp;
    }

    /*
      Prints the interchanged diagonals matrix
     */
    printf("\nMatrix after diagonals interchanged: \n");
    for(row=0; row < MAX_ROWS; row++)
    {
        for(col=0; col < MAX_COLS; col++)
        {
            printf("%d ", A[row][col]);
        }

        printf("\n");
    }

    return 0;
}
```

C program to find transpose of a matrix

Example: If elements of the matrix are:

```
1 2 3
4 5 6
7 8 9
```

Then its transpose is :
1 4 7
2 5 8
3 6 9

Transpose of a matrix:

Transpose of a matrix A is defined as converting all rows into columns and columns into rows. Transpose of a matrix A is written as AT.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}^T = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

Transpose of a matrix

Algorithm to find transpose of a matrix:
To find transpose of a matrix A.
Step 1: Read elements in matrix A

Step 2: For each element in matrix A.
$B_{ij} = A_{ji}$ (Where $1 \le i \le m$ and $1 \le j \le n$).
Program:

```
#include <stdio.h>

int main()
{
   int A[3][3], B[3][3];
   int row, col;

   /*
     Reads elements in matrix A from user */
   printf("Enter elements in matrix of size 3x3: \n");
   for(row=0; row<3; row++)
   {
      for(col=0; col<3; col++)
      {
         scanf("%d", &A[row][col]);
      }
   }

   /*
     Finds the transpose of matrix A */
   for(row=0; row<3; row++)
   {
      for(col=0; col<3; col++)
      {
         /* Stores each row of matrix A to each column of matrix B */
         B[row][col] = A[col][row];
```

```
        }
    }

    /*
       Prints the original matrix A */
    printf("\nOriginal matrix: \n");
    for(row=0; row<3; row++)
    {
        for(col=0; col<3; col++)
        {
            printf("%d ", A[row][col]);
        }

        printf("\n");
    }

    /*
       Prints the transpose of matrix A */
    printf("Transpose of matrix A: \n");
    for(row=0; row<3; row++)
    {
        for(col=0; col<3; col++)
        {
            printf("%d ", B[row][col]);



        }

        printf("\n");
    }

    return 0;
}
```

C program to count frequency of digits in an integer
Example:

Input num: 116540
Output: Frequency
of 0 = 1 Frequency
of 1 = 2 Frequency
of 2 = 0 Frequency
of 3 = 0 Frequency
of 4 = 1 Frequency
of 5 = 1 Frequency
of 6 = 1 Frequency
of 7 = 0 Frequency
of 8 = 0 Frequency
of 9 = 0
You must have a good understanding of below three concepts
Here goes the step-by-step detailed explantion about how you gonna find the frequency of digits in an integer.

First of all we need a storage where we can store frequencies of each digits. For that we will be using an integer array of size 10 call it as freq[10]. We have used an array of size 10 because decimal has base 10. There are only 10 digits that makes up any decimal number.

Next, we need to initialize every element of the array with 0. Assuming that every digit has occurred 0 times.

Now comes the main logic. Find the last digit of the given number. For that we need to perform modular division by 10 i.e. lastDigit = num % 10 (where num is the number whose frequency of digits is to be found).

Increment the freq[lastDigit]++ by one.

Now remove the last digit from the num as it isn't needed anymore. Perform num = num / 10.

Repeat steps 3-5 till num != 0. Finally you will be left with an array freq having the frequency of each digits in that number.

```
#include <stdio.h>
#define BASE 10
 int main()
{ long long num, n;
   int i, lastDigit; int
   freq[BASE];

   printf("Enter any number: ");
   scanf("%lld", &num);
       Initializes frequency array with
   0 for(i=0; i<BASE; i++)
   {freq[i] = 0;
   }
    n = num; //Copies the value of num to n
    while(n != 0)
   {// Gets the last digit
      lastDigit = n % 10;
          Increments the frequency
      array freq[lastDigit]++;
          Removes the last digit from
      n n /= 10;
   }
   printf("Frequency of each digit in %lld is: \n", num);
```

```
    for(i=0; i<BASE; i++)
    {
        printf("Frequency of %d = %d\n", i, freq[i]);
    }
     return 0;
}
```

C program to check symmetric matrix

Example: If elements of the matrix is:
1 2 3
2 4 5
3 5 8

Output: Given matrix is symmetric matrix.
Before checking whether a matrix is symmetric or not you must know how to find transpose of a matrix in C.

Symmetric Matrix:

Symmetric matrix is a square matrix which is equal to its transpose. All symmetric matrix must be a square matrix.

A symmetric matrix A is defined as: A = AT

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 8 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 8 \end{bmatrix}^T$$

Symmetric matrix

Algorithm to check symmetric matrix:
To check whether a matrix A is symmetric or not we need to check whether A = AT or not.

Step 1: Read elements in matrix A.
Step 2: Find transpose of matrix A.
Step 3: If matrix A is equal to its transpose AT then it is symmetric matrix.
If Aij = ATij (Where $1 \leq i \leq m$ and $1 \leq j \leq n$) then the matrix is symmetric.
Program:

```
#include <stdio.h>

int main()
{
    int A[3][3], B[3][3];
    int row, col, isSymmetric;

    /*
       Reads elements in matrix A from user */
    printf("Enter elements in matrix of size 3x3: \n");
    for(row=0; row<3; row++)
    {
        for(col=0; col<3; col++)
        {
            scanf("%d", &A[row][col]);
        }
    }
    /*
      Finds the transpose of matrix A */
    for(row=0; row<3; row++)
    {
        for(col=0; col<3; col++)
        {
            /* Stores each row of matrix A to each column of matrix B */
            B[row][col] = A[col][row];
        }
    }

    /*
      Checks whether matrix A is equal to its transpose or not */
    isSymmetric = 1;
    for(row=0; row<3 && isSymmetric; row++)
    {
        for(col=0; col<3; col++)
        {
            /* If matrix A is not equal to its transpose */
            if(A[row][col] != B[row][col])
            {
                isSymmetric = 0;
                break;
            }
        }
    }

    /*
      If the given matrix is symmetric. */
    if(isSymmetric==1)
    {
        printf("\nThe given matrix is Symmetric matrix: \n");

        for(row=0; row<3; row++)
        {
            for(col=0; col<3; col++)
            {
                printf("%d ", A[row][col]);
            }

            printf("\n");
        }
    }
    else
    {
        printf("\nThe given matrix is not Symmetric matrix.");
    }

    return 0;
}
```

C program to find length of a string

Example:
Input string: I love programming. I love CodeforWin.
Output length of string: 38

Logic to find length of a string
In C there is a general concept that every string must terminate with a special character that is NULL character which is also escaped as \0. Hence we use a this basic concept to find length of string. What we need to do is traverse entire string character-by-character till a NULL \0 character is found and in each iteration increment the counter value.

```
#include <stdio.h>

int main()
{
    char text[100]; /* Declares a string of size 100 */
    int index= 0;

    printf("Enter any string: ");
    gets(text);

    while(text[index]!='\0')
    {
        index++;
    }

    printf("Length of '%s' = %d", text, index);

    return 0;
}
```

C program to compare two strings
Example:
Input string1: CodeforWin
Input string2: CodeforWin
Output: Both strings are equal

```
#include <stdio.h>

#define MAX_SIZE 100 //Maximum size of the string


/* Function declarations */
int compare(char * string1, char * string2);



int main()
{
    char string1[MAX_SIZE], string2[MAX_SIZE];
    int res;
    /* Reads two strings from user */
    printf("Enter first string: ");
    gets(string1);
    printf("Enter second string: ");
    gets(string2);


    /* Calls function to compare both strings and stores result in res */
    res = compare(string1, string2);

    if(res == 0)
    {
        printf("Both strings are equal.\n");
    }
    else if(res == -1)
    {
        printf("First string is lexicographically smaller than second.\n");
    }
    else
    {
        printf("First string is lexicographically greater than second.\n");
    }

    return 0;
}

/**
    Compares two strings lexicographically.
    Returns 0 if both strings are equal,
        -1 if first string is smaller
        otherwise returns 1
 */
int compare(char * string1, char * string2)
{
    int i=0;

    /* Runs till both strings are equal */
    while(string1[i] == string2[i])
    {
        if(string1[i] == '\0' || string2[i] == '\0')
            break;

        i++;
    }

    if(string1[i-1] == '\0' && string2[i-1]=='\0')
        return 0;
    else if(string1[i] > string2[i])
        return 1;
    else if(string1[i] < string2[i])
        return -1;
}
```

C program to convert lowercase string to uppercase
Example:
Input string: I love programming!

Output string: I LOVE
PROGRAMMING! Logic to convert
lowercase string to uppercase
Internally characters in C are represented as an integer value known as ASCII value. Which means if we write A or any other character it is translated into a numeric value in our case it is 65 as ASCII value of A = 97.

Here what we need to do is first we need to check whether the given character is lowercase alphabet or not. If it is lowercase alphabet just subtract 32 from it which will result in uppercase alphabet (Since ASCII value of A=65, a=97 their difference is 97-65 = 32).
Algorithm to convert lowercase to uppercase

```
%%Input : text {Array of characters / String}
        N {Size of the String}

Begin:
    For i ← 0 to N do
        If (text[i] >= 'a' and text[i] <= 'z') then
            text[i] ← text[i] - 32;
        End if
    End for
End
```

```c
#include <stdio.h>

#define MAX_SIZE 100 //Maximum size of the string


int main()
{
    char string[MAX_SIZE];
    int i;

    /* Reads a string from user */
    printf("Enter your text : ");
    gets(string);

    for(i=0; string[i]!='\0'; i++)
    {
        /*
            If current character is lowercase alphabet then
            convert it to uppercase.
         */
        if(string[i]>='a' && string[i]<='z')
        {
            string[i] = string[i]-32;
        }
    }

    printf("Uppercase string : %s\n",string);
    return 0;
}
```

C program to find total number of alphabets, digits or special characters in a string Example:
Input string: I love programming.
Output: Alphabets = 16
Digits = 0
Special character = 3
Total length = 19

```
#include <stdio.h>

#define MAX_SIZE 100 //Maximum size of the string
  int main()
{
    char string[MAX_SIZE];
    int alphabets, digits, others, i;

    alphabets = digits = others = i = 0;


    /* Reads a string from user */
    printf("Enter any string : ");
    gets(string);

    /*
       Checks each character of string
     */
    while(string[i]!='\0')
    {
        if((string[i]>='a' && string[i]<='z') || (string[i]>='A' && string[i]<='Z'))
        {
            alphabets++;
        }
        else if(string[i]>='0' && string[i]<='9')
        {
            digits++;
        }
        else
        {
            others++;
        }

        i++;
    }





    printf("Alphabets = %d\n", alphabets);
    printf("Digits = %d\n", digits);
    printf("Special characters = %d\n", others);

    return 0;
}
```

C program to count total number of vowels and consonants in a string

Example:
Input text: I love CodeforWin!
Total Vowels = 7
Total Consonants = 8

Logic to count number of vowels and consonants

Logic to count total number of vowels/consonants uses a basic concept. What we need to do is we need to traverse entire string character by character and increment the vowel counter if the current character is vowel otherwise increment consonant counter. Therefore before counting total number of vowels and consonants in a string we must know how to check whether the current character is vowel or consonant.

Algorithm to count total number of vowels and consonants in a string

```
%%Input : text {Array of characters /String}
        N {Size of the string}
Begin:
    vowel ← 0;
    consonant ← 0;
    For i ← 0 to N do
        If (text[i] == 'a','e','i','o','u','A','E','I','O','U')
            then vowel ← vowel + 1;
        End if
        Else if (text[i] >='a' and <='z') or (text[i] >='A' and <='Z')
            then consonant ← consonant + 1;
        End if
    End for
End
```

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

#define MAX_SIZE 100 //Maximum size of the string

int main()
{
   char string[MAX_SIZE];
   int i, len, vowel, consonant;

   /* Reads string from user */
   printf("Enter any string: ");
   gets(string);

   vowel = 0;
   consonant = 0;
   len = strlen(string);

   for(i=0; i<len; i++)
   {
      /*
         If the current character(string[i]) is a vowel both upper and lowercase
       characters */
      if(string[i] =='a' || string[i]=='e' || string[i]=='i' || string[i]=='o' || string[i]=='u' || string[i]=='A' || string[i]=='E' || string[i]=='I' ||
string[i]=='O' || string[i]=='U')
      {
         vowel++;
      }
      else if((string[i]>='a' && string[i]<='z') || (string[i]>='A' && string[i]<='Z'))
      {
         consonant++;
      }
   }

   printf("Total number of vowel = %d\n", vowel);
   printf("Total number of consonant = %d\n", consonant);

   return 0;
}
```

C program to find reverse of a string
Example:
Input string: Hello
Output reverse string: olleH
Program to find reverse of a string

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

#define MAX_SIZE 100 //Maximum size of the string

int main()
{
   char string[MAX_SIZE], reverse[MAX_SIZE];
   int i, j, len;

   /* Reads string from user */
```

```
    printf("Enter any string: ");
    gets(string);

    len = strlen(string);
    j = 0;

    for(i=len-1; i>=0; i--)
    {
        reverse[j] = string[i];
        j++;
    }
    reverse[j] = '\0';

    printf("\nOriginal string = %s\n", string);
    printf("Reverse string = %s", reverse);

    return 0;
}
```

C program to check whether a string is palindrome or not
Example:
Input String: madam
Output: Palindrome string

Palindromic string

Palindrome string is a special string which reads same from backward or forward such as madam, mom, eye, dad etc.

Logic to check palindrome string
The basic idea behind checking palindrome is if it can be read same from forward and backward then it is palindrome else not. Here in the below algorithm we will traverse the string character by character in both direction at the same time if they are equal then the string is palindrome.
Algorithm to check Palindrome string

```
%%Input : text {Array of characters /String)
        N {Size of the string}
Begin:
    S ← 0; E ← N-1;
    While (S <= E) do
        If (text[S] != text[E])
            break
        End if
        S ← S + 1;
        E ← E - 1;
    End while
    If (S >= E) then
        write ('String is Palindrome')
    End if
    Else then
```

```
        write ('String is Not Palindrome')
    End if
End
```

C program to search all occurrences of a character in a string

Example:
Input string: I love programming. I love CodeforWin.
Input character to search: o
Output 'o' found at index: 3, 9, 23, 28, 32
Program to search occurrence of character in string

```
#include <stdio.h>

#define MAX_SIZE 100 //Maximum size of the string

int main()



{
    char string[MAX_SIZE];
    char search;
    int i;

    /*
       Reads a string and character from user
     */
    printf("Enter any string: ");
    gets(string);
    printf("Enter any character to search: ");
    search = getchar();

    //Runs a loop till the NULL character is found
    for(i=0; string[i]!='\0'; i++)
    {
        /*
           If character is found in string
         */
        if(string[i] == search)
        {
            printf("'%c' is found at index %d\n", search, i);
        }
    }

    return 0;
}
```

C program to remove spaces, blanks from a string
Example:
Input string: "I   love   C   programming!"
Output string: "I love C programming!"
Program to remove extra spaces from string

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

#define MAX_SIZE 100 //Maximum size of the string


/* Function declaration */
char * removeBlanks(const char * string);



int main()
{
    char string[MAX_SIZE];
    char * newString;

    printf("Enter any string: ");
    gets(string);

    printf("\nString before removing blanks: \"%s\"\n", string);

    newString = removeBlanks(string);



    printf("String after removing blanks: \"%s\"\n", newString);

    return 0;
}


/**
    Removes extra blank spaces from the given string
    and returns the new string with single
 blank spaces */
char * removeBlanks(const char * string)
{
    int i, j;
    char * newString;

    newString = (char *)malloc(strlen(string) + 1);
    i = 0;
    j = 0;

    while(string[i] != '\0')
    {
        /* If blank space is found */
        if(string[i] == ' ')
        {
            newString[j] = ' ';
            j++;

            /* Skip all further spaces */
            while(string[i] == ' ')
                i++;
        }

        newString[j] = string[i];

        i++;
        j++;
    }
    //Make sure that the string is NULL terminated
    newString[j] = '\0';

    return newString;
}
```

C program to convert Binary to Decimal number system
Example:
Input binary number: 0011
Output decimal number:
3 Algorithm:
Steps to convert from binary number system to decimal number
system: Step 1: Read binary number in binary.
Step 2: Set decimal = 0 and N = 0.
Step 3: Get Last digit of binary number and store in some variable i.e. rem = binary % 10.
Step 4: If rem == 1 then decimal = decimal + pow(2, N). (Where pow() is a function used to calculate power of any number.
Step 5: N = N + 1 and binary = binary / 10.
Step 6: Repeat Step 3-5 till binary > 0 .
Step 7: Print the value of decimal.

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

#define BASE 2

int main()
```

```
{
   long long binary, decimal=0, tempBinary;
   int N=0;

   printf("Enter any binary number: ");
   scanf("%lld", &binary);

   tempBinary = binary;

   while(tempBinary!=0)
   {
      if(tempBinary%10 == 1)
      {
         decimal += pow(BASE, N);
      }

      N++;
      tempBinary /= 10;
   }

   printf("Binary number = %lld\n", binary);
   printf("Decimal number= %lld", decimal);

   return 0;
}

}
```

C program to convert Decimal to Binary number system

Example:
Input any decimal number: 112
Output binary number: 0111000

Decimal number system:
Decimal number system is a base 10 number system. Decimal number system uses only 10 symbols to represent all number i.e. 0 1 2 3 4 5 6 7 8 9

Binary number system:

Binary number system is a base 2 number system. Binary number system uses only 2 symbols to represent all numbers i.e. 0 and 1

Algorithm to convert from Decimal to Binary number system:

```
Algorithm Decimal to Binary conversion
begin:
read (DECIMAL);
BINARY ← 0; PLACE ← 1; REM ← 0;
while (DECIMAL !=0) do
begin
```

```
    REM ← DECIMAL % 2;
    BINARY ← (REM * PLACE) + BINARY;
    PLACE ← PLACE * 10;
    DECIMAL ← DECIMAL / 2;
end
write('Binary = ' BINARY)
end
```

Program:

```
#include <stdio.h>

int main()
{
    long long decimal, tempDecimal, binary;
    int rem, place = 1;

    binary = 0;

    /*
       Reads decimal number
     from user */
    printf("Enter any decimal number: ");
    scanf("%lld", &decimal);
    tempDecimal = decimal;

    /*
       Converts the decimal number to
     binary number */
    while(tempDecimal!=0)
    {
        rem = tempDecimal % 2;

        binary = (rem * place) + binary;

        tempDecimal /= 2;
        place *= 10;
    }

    printf("\nDecimal number = %lld\n", decimal);
    printf("Binary number = %lld", binary);

    return 0;


}
```

Theme: File Handling-1

WAP to swap two numbers using concept of pointers.

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

```
void swap(int *num1, int *num2) {
  int temp;
  temp = *num1;
  *num1 = *num2;
  *num2 = temp;
}

int main() {
  int num1, num2;

  printf("\nEnter the first number : ");
  scanf("%d", &num1);
  printf("\nEnter the Second number : ");
  scanf("%d", &num2);

  swap(&num1, &num2);

  printf("\nFirst number  : %d", num1);
  printf("\nSecond number : %d", num2);

  return (0);
}
```

WAP to compare the contents of two files and determine whether they are same or not.

```
#include<stdio.h>

int main() {
  FILE *fp1, *fp2;
  int ch1, ch2;
  char fname1[40], fname2[40];

  printf("Enter name of first file :");
  gets(fname1);

  printf("Enter name of second file:");
  gets(fname2);

  fp1 = fopen(fname1, "r");
  fp2 = fopen(fname2, "r");

  if (fp1 == NULL) {
    printf("Cannot open %s for reading ", fname1);
    exit(1);
  } else if (fp2 == NULL) {
    printf("Cannot open %s for reading ", fname2);
    exit(1);
  } else {
    ch1 = getc(fp1);
    ch2 = getc(fp2);
    while ((ch1 != EOF) && (ch2 != EOF) && (ch1 == ch2)) {
      ch1 = getc(fp1);
      ch2 = getc(fp2);
    }

    if (ch1 == ch2)
      printf("Files are identical n");
    else if (ch1 != ch2)
      printf("Files are Not identical n");

    fclose(fp1);
    fclose(fp2);
  }
  return (0);
}
```

Theme: File Handling-2

WAP to check whether a given word exists in a file or not. If yes then find the number of times it occurs.

```
#include "stdafx.h"
# include <stdio.h>
# include <conio.h>
# include<string.h>
# include <stdlib.h>
# include <conio.h>
# include <stdlib.h>


 void checkRepeats( char sentence[], char text[]);

 int _tmain(int argc, _TCHAR* argv[])
 {
    while(1)
    {
                    system("cls");
                    char sentence[500], x, text[10];

                    printf("Enter the string/sentence\n=-=-=->");
                    fgets(sentence, sizeof(sentence), stdin);
                    printf("please enter the search word\n=-=-=->");
                    fgets(text, sizeof(text), stdin);

                    checkRepeats(sentence, text);

                    printf("\n\nPress ENTER to try again, '$' to quit : ");
                    x = getchar();
                    if (x=='$')
                    break;
    }
 }
```

```
void checkRepeats ( char sentence[], char text[])
{
int i=0, j, k, len, chkword , words=0, again=0;
char wrd[10];
len = strlen(sentence);

            while(sentence[i]!='\0')
              {

                    if(sentence[i]==' ')
                    {
                            words++;
                            i++;

                            wrd[i] = sentence[i];

                    }

                    if(strcmp(wrd,text)==1)
                    {

                            again++;
                    }

                    else
                    printf("The search word does not exist in the sentence");
            }

            if (len==0)
              {
                    printf("\n\nRESULTS:\n\nNumber of words in the text: %d",words);
                    printf("\n\nCharacters:%d \n",len);
              }

            else
              {
                    printf("\n\nRESULTS:\n\nNumber of words: %d",words+1);
                    printf("\nCharacters:%d \n",len-1);
                                                                          }

             printf("The search word repeats %d times in the sentence\n", again);

}
```