AMRITA VISHWA VIDYAPEETHAM CHENNAI CAMPUS

Second Year. B.Tech (Computer Science and Engineering)

Problem Solving using ‘C’ programming

Work Book

(From Academic year 2025-26 ODD)

Name: S DEEPAK BHARATHWAJ

College Name: Amrita Vishwa Vidyapeetham, Chennai

Roll No.: CH.SC.U4CSE24015 Department: CSE

Academic Year: 2025-26

Preface to the First Edition

The new syllabus for First Year B.Tech Computer Science is implemented from the academic year 2022.

It is absolutely necessary and essential that all the Computer Science practicals be conducted on Open Source Operating System like Linux or windows. All the practical's related to C needs to be conducted using GCC compiler.

.

It is mandatory to carry the completed and duly signed lab Activity sheets for the practical examination.

Lab Incharge: Dr J.Umamageswaran

Introduction

1. About the work book

This workbook is intended to be used by First Year B.Tech. (Computer Science) students for the Computer Science laboratory courses in their curriculum. In Computer Science, hands-on laboratory experience is critical to the understanding of theoretical concepts studied in the theory courses. This workbook provides the requisite background material as well as numerous computing problems covering all difficulty levels.

The objectives of this book are

1) Defining clearly the scope of the course
2) Bringing uniformity in the way the course is conducted across different colleges
3) Continuous assessment of the course
4) Bring in variation and variety in the experiments carried out by different students in a batch
5) Providing ready reference for students while working in the lab
6) Catering to the need of slow paced as well as fast paced learners

2. How to use this work book?

This workbook is mandatory for the completion of the laboratory course. It is a measure of the performance of the student in the laboratory for the entire duration of the course.

2.1 Instructions to the students

Please read the following instructions carefully and follow them

1) Carry this book every time you come to the lab for computer science practical's
2) You should prepare yourself beforehand for the Exercise by reading the material mentioned under.
3) If the self-activity exercise or assessment work contains any blanks such as _____, get them filled by your instructor.
4) Instructor will specify which problems you are to solve by ticking box
5) You will be assessed for each exercise on a scale of 5

i) Not done	0	ii) Incomplete	1
iii) Late Complete	2	iv) Needs improvement	3
v) Complete	4	vi) Well Done	5

2.2. Instruction to the Instructors

1) Explain the assignment and related concepts in around ten minutes using white board if required or by demonstrating the software
2) Fill in the blanks with different values for each student
3) Choose appropriate problems to be solved by student by ticking box
4) Make sure that students follow the instruction as given above
5) After a student completes a specific set, the instructor has to verify the outputs and sign in the provided space after the activity.
6) You should evaluate each assignment carried out by a student on a scale of 5 as specified above by ticking appropriate box.
7) The value should also be entered on assignment completion page of the respective Lab course

2.3. Instructions to the Lab administrator

You have to ensure appropriate hardware and software is made available to each student.

The operating system and software requirements on server side and also client side are as given below

1) Server Side (Operating System)
    a. * Fedora Core Linux *
    b. Servers Side (software's to be installed)

    In Linux – C, C++, awk, shell, perl, postgresql/Mysql

2) Client Side (Operating System)

    * Red Hat Linux and Fedora Core *

    Client Side (software's to be installed)

    In Linux – C, C++, awk, shell, perl, postgresql/mysql

Information about installation and configuring of the server and client are provided in appendix A

Activity Completion Sheet

Problem Solving and C programming		
Sr.No	**Assignment Name**	**Marks**
1	Problem Solving using Pseudo code and Flowchart, Simple programs, Understanding errors and error handling.	
2	Decision Making Control Structures.	
3	Loop Control Structures	
4	Functions (User Defined functions, Library functions and Recursion).	
5	Arrays (1-D and 2-D)	
6	Strings	
7	Pointers	

CERTIFICATE

This is to certify that, Mr./Miss._____Roll No._____of I year B.Tech. (Computer Science and Engineering) has successfully completed __out of ___ practical's satisfactorily during the academic year 20 - 20 .

Practical In-charge

Chairperson

Problem Solving using ‘C’ programming

You should read following topics before starting this exercise

1. **UNIX and LINUX operating system**

About UNIX and LINUX

The success story of UNIX starts with the failure of the MULTICS project. The project failed and the powerful GE-645 machine was withdrawn by GE. Two scientists at Bell Labs, Ken Thompson and Dennis Ritchie, who were part of the MULTICS team, continued to work and succeeded and named their Operating system UNIX, a pun on MULTICS.

The machine available at Bell Labs was a DEC PDP-7 with only 64 k memory while the Operating system they were developing was meant for a larger machine. The problematic situation was handled with an innovative solution. They developed most part of the software in a higher-level language, C, which helped them in porting their Operating system from one hardware to another.

With the growing popularity of UNIX, it was available on a variety of machines, from personal computers to mainframes. The most popular amongst them was UNIX System V from AT&T.

Each big player in the market came up with their own versions of UNIX. IBM had its own version of UNIX called AIX, which was used on high-end servers. Sun's version of UNIX called Solaris was used on Sun workstations. Novell marketed UnixWare along with Netware, its Network operating system.

LINUX is a version of UNIX, which though it resembles UNIX in looks and feels but differs from other versions in the way it was developed and distributed. In contrast to large proprietary UNIX versions, Linux was developed by Linus Torvalds, a Finnish student. He made the source code available and invited partners via the internet in his development effort. He got professional help from all quarters and Linux evolved rapidly. It was made freely available for everyone to use. Linux that was initially meant for Personal computers is now available for a variety of hardware platforms, from mainframes to handheld computers

Linux supports multiple users. Every user needs to have an account in order to use the system. One of the users called system administrator (root) is given the charge of creating user accounts and managing the system normally works on the "#" prompt.

You will be given a username and password, using which you can login into Linux operating system.

For computer users, the operating system provides a user-command interface that is easy to use, usually called the Shell. The user can type commands at the shell prompt and get the services of the operating system. Linux operating system shell has the "$" prompt.

You can open a system terminal that gives you a $ prompt where you can type in various shell commands.

LINUX system will usually offer a variety of shell types:

- sh or Bourne Shell: the original shell still used on UNIX systems and in UNIX-related environments. It is available on every Linux system for compatibility with UNIX programs.
- bash or Bourne Again shell: the standard GNU shell, is the standard shell for common users on Linux and is a superset of the Bourne shell.
- csh or C shell: the syntax of this shell resembles that of the C programming language.
- tcsh or Turbo C shell: a superset of the common C shell, enhancing user-friendliness and speed.
- ksh or the Korn shell: A superset of the Bourne shell

Activity Sheet 1 **Date:**

Problem Solving using Pseudo code and Flowchart, Simple programs, understanding errors and error handling.

Objective-To demonstrate the use of data types, simple operators and expressions

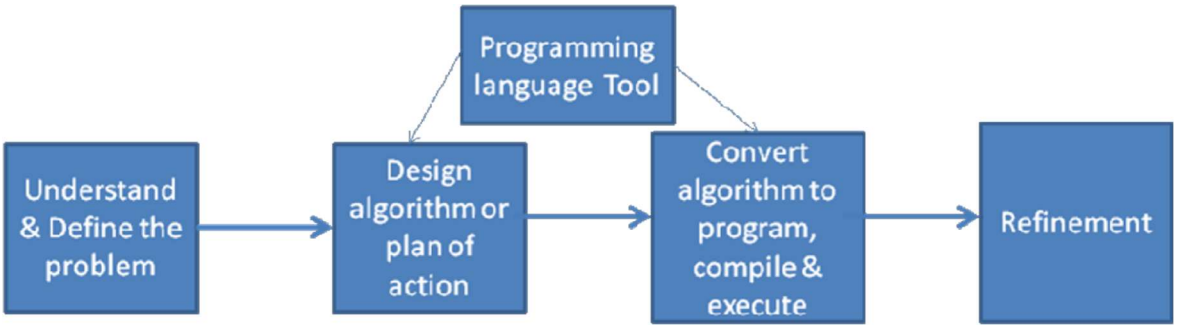
Reading-

You should read following topics before starting this exercise

1. Problem solving steps- writing algorithms and flowcharts
2. Different basic data types in C and rules of declaring variables in C
3. Different operators and operator symbols in C
4. How to construct expressions in C, operator precedence

Program solving using Computers

The steps in solving a problem using computers are shown below



Pseudo Language conventions

1. Appropriate names should be given to variables – the algorithm itself is given a name

   Algorithm FindingMaximum

2. Organized sequence of data values is given a name and each data value is indicated by giving the index in brackets

   Values[position]

3. Expressions containing Arithmetic operators +, -, * , / relational operators <, ≤ , >, ≥, ≠, = and logical and, or , not can be used

   Example: length * breadth, previous < next

4. Assignment statement variable = expression is used to assign data values to variables

5. Conditional constructs for making decisions

   if condition then statements

   Statements are to be carried out only if condition is true

   if condition then statements1 else statements2

   Statements1 are to be carried out only if condition is true and statements2 are to be carried out if condition is false

6. Loops

   i. while condition do statements

      Statements are to be repeated while condition is true

   ii. for variable = value1 to value2 do statements

      Statements are to be repeated moving from value1 to value2, with an increment one step.

   iii. for variable = value1 downto value2 do { statements }

      Statements are to be repeated moving from value1 down to value2, with a decrement one step.

   iv. repeat { statements } until condition

      Statements are to be repeated until condition is true

7. Input /output statements

   i. read value For taking value as input for the algorithm

   ii. write value For printing value as output of the algorithm

8. Algorithm name(value1 , valu2) – taking input for the algorithm

9. return value1 , value2 – for giving output of the algorithm

Examples:

Problem Statement: Accept radius and calculate area and circumference of a circle

```
Algorithm AreaCircumference
Begin
        Input radius pi=3.142
        area = piXradiusXradius
        circum= 2XpiXradius  Output
area
        Output circum
End
```

Problem Statement: Check if a number is even

```
Algorithm Even
Begin Input m if m mod 2 = 0 then
        output “m is even”
End
```

Problem Statement: Find maximum of two numbers

```
Algorithm Maximum
Begin
        Input m Input n if
        m > n then  output
        m
        else  output n
End
```

Problem Statement: Give a discount of 15 % when purchase amount exceeds 5000, otherwise give a discount of 10%

```
Algorithm Discount Input
amount
if amount exceeds  5000 then  compute discount as 25 times
        amount divided by 100
else compute discount as 15 times amount divided by 100
Subtract discount from amount
Output amount
End
```

Problem Statement: Given a set of 100 values representing marks of students, count the total students that have passed. (A score of 40 is required for passing.)

```
Algorithm PassCount1
Begin
         count = 0
        n = 1
         While n <= 100 do
                        Input marks
                       If marks >= 40 then
                                increment count by 1
                          n = n+1
        Output count
 End
```

The same can be done using another loop construct as shown below:

```
Algorithm PassCount1
Begin
         count = 0
        for n = 1 to 100 do
                        Input marks
                       If marks >= 40 then  increment
        count by 1 Output count
 End
```

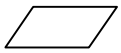
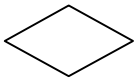
Problem Statement: Accept characters till a * is entered from the keyboard and count the number of characters entered.

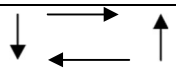
```
Algorithm CharacterCount
Begin
        count = 0
        Input char
        while char ≠ * do
count = count + 1
                        Input char
        Output count
 End
```

Problem Statement: Accept a number and calculate the sum of its digits.

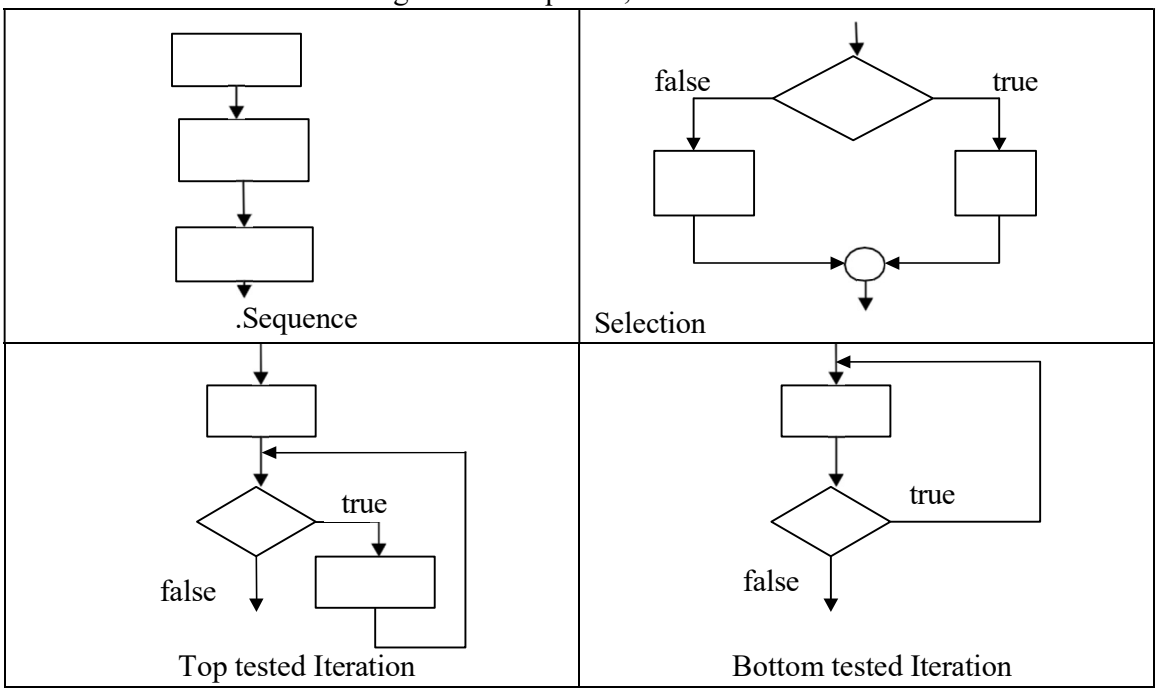
```
Algorithm SumDigits
Begin
        Input num
        sum=0
        repeat
                        digit = num mod 10
                        sum=sum+digit
        num = num/10
        until num>0 Output
        sum
 End
```

FLOWCHART SYMBOLS

Start Statement	
Input Statement	
Statement or procedure	
Decision, choice or Selection	
Connectors	

Control flow	

Basic control structures in an algorithm: Sequence, Selection and Iteration are shown below



1. Data type Table

Data	Data Format	C Data Type	C Variable declaration	Input Statement	Output statement

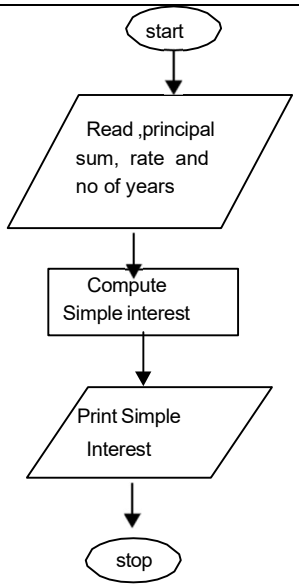
quantity
month
creditcard number
	Numeric	int
		Short int
		long int
			int quantity;
			short month;
			long ccno;
				scanf("%d",&quantity);
				scanf("%d",&month);
				scanf("%ld", &ccno);
					printf("The quantity is %d", quantity);
					printf("Credit card number is %ld, ccno);

price π	real	float double	float price; const double pi=3.141593 ;	scanf("%f",&price);	printf("The price is %5.2f", price);
grade	character		char grade;	scanf("%c",&grade)	printf("The grade is %c",grade);

2. Expression Examples

Expression	C expression
Increment by a 3	a = a + 3
Decrement b by 1	b = b-1 or b--
$2\ a^2 + 5\ b/2$	2*a*a + 5*b/2
7/13(x-5)	(float)7/13*(x-5)
5% of 56	(float)5/100*56
n is between 12 to 70	n>=12 && n<=70
$\pi r^2 h$	Pi*r*r*h
n is not divisible by 7	n % 7 != 0
n is even	n%2== 0
ch is an alphabet	ch>='A' && ch<='Z' || ch>='a' && ch<='z'

Step 1: Writing the Pseudocode	Step 2: Draw the flowchart	Step 3: Write the Program

Algorithm SimpleInterest
Begin
Input amount, rate, years
si =amountXyearsXrate / 100
Output si
End
		#include <stdio.h>
		void main( )
		{
		/* variable declarations */
		float amount, rateOfInterest, simpleInterest;
		int noOfYears;
		/* prompting and accepting input */
		printf("Give the Principal Sum");
		scanf("%f",&amount);
		printf("Give the Rate of Interest");
		scanf("%f",&rateOfInterest);
		printf("Give the Number of years");
		scanf("%d",&noOfYears);
		/* Compute the simple Interest*/
		simpleInterest=amount*noOfYears*rateOfInterest / 100 ;
		/* Print the result*/
		printf("The simple Interest on amount %7.2f for %d years at the rate %4.2f is %6.2f", amount, noOfYears, rateOfInterest, simpleInterest);
		}

Self-Activity

1. Type the sample program given above. Execute it for the different values as given below and fill the last column from the output given by the program. Follow the following guidelines

a. At $ prompt type gedit followed by filename. The filename should have .c as extension for example $gedit pnr.c

b. Type the sample program and save it. Compile the program using cc compiler available in Linux
$cc pnr.c
It will give errors if any or it will give back the $ prompt if there are no errors
A executable file a.out is created by the compiler. The program can be executed by typing name of the file as follow $ ./a.out
Alternatively, the executable file can be given name by using –o option while compiling as follows $cc pnr.c –o pnrexec
$./pnrexec

Sr. No	Principal sum	No of years	Rate of interest	Simple Interest
1	2000	3	____	
2	4500	___	4.5	
3	_____	6	8.3	

Problem Solving using Pseudo code and Flowchart, Simple programs, understanding errors and error handling.

Objective-To demonstrate the use of data types, simple operators and expressions

Problem Statement – 1: Accept radius and calculate area and circumference of a circle

CODE:

```c
#include <stdio.h>
void main()
{
    float radius;
    float pi = 3.142;
    printf("Enter the radius of the circle: ");
    scanf("%f", &radius);
    float area = pi * radius * radius;
    float circum = 2 * pi * radius;
    printf("Area of the circle: %.2f\n", area);
    printf("Circumference of the circle: %.2f\n", circum);
}
```

OUTPUT:

```
Enter the radius of the circle: 9
Area of the circle: 254.50
Circumference of the circle: 56.56

--------------------------------
Process exited after 7.183 seconds with return value 35
Press any key to continue . . .
```

Problem Statement - 2: Check if a number is even

CODE:

```c
#include <stdio.h>
int main()
{
    int m;
    printf("Enter the number : ");
    scanf("%d",&m);
    if(m%2 == 0){
        printf("m is even");
    }
    return 1;
}
```

OUTPUT:

```
Enter the number : 243478
m is even
--------------------------------
Process exited after 19.71 seconds with return value 1
Press any key to continue . . .
```

Problem Statement - 3: Find maximum of two numbers

CODE:

```c
#include <stdio.h>
int main(){
    int m,n;
    printf("Enter m: ");
    scanf("%d",&m);
    printf("Enter n: ");
    scanf("%d",&n);
    if (m>n){
        printf("%d",m);
    }
    else{
        printf("%d",n);
    }
    return 0;
}
```

OUTPUT:

```
Enter m: 5
Enter n: 8
8
--------------------------------
Process exited after 2.343 seconds with return value 0
Press any key to continue . . .
```

Problem Statement – 4: Give a discount of 25 % when purchase amount exceeds 5000, otherwise give a discount of 15%

CODE:

```c
#include <stdio.h>
int main(){
    int discount;
    float amount;
    printf("Enter the amount: ");
    scanf("%f",&amount);
    if(amount>5000){
        discount=(15*amount)/100;
    }
    else{
        discount=(10*amount)/100;
    }
    amount=amount-discount;
    printf("%f",amount);
}
```

OUTPUT:

```
Enter the amount: 5678
4827.000000
--------------------------------
Process exited after 3.466 seconds with return value 11
Press any key to continue . . .
```

Problem Statement - 5: Given a set of 5 values representing marks of students, count the total students that have passed. (A score of 40 is required for passing.)

CODE :

```c
#include <stdio.h>
int main(){
    int count=0,marks;
    int n=1;
    while(n<=5){
        printf("Enter the marks of the student: ");
        scanf("%d",&marks);
        if(marks>=40){
            count=count+1;
        }
        n=n+1;
    }
    printf("%d",count);
}
```

OUTPUT:

```
Enter the marks of the student: 40
Enter the marks of the student: 25
Enter the marks of the student: 23
Enter the marks of the student: 59
Enter the marks of the student: 99
3
--------------------------------
Process exited after 9.704 seconds with return value 1
Press any key to continue . . .
```

Problem Statement – 6: Accept characters till a * is entered from the keyboard and count the number of characters entered

CODE:

```c
#include <stdio.h>
int main(){
    char charac;
    int count = 0;
    printf("Enter characters:\n");
    scanf(" %c", &charac);
    while (charac!='*') {
        count++;
        scanf(" %c", &charac);
    }
    printf("%d", count);
    return 0;
}
```

OUTPUT:

```
Enter characters:
^
%
*
2
--------------------------------
Process exited after 6.49 seconds with return value 0
Press any key to continue . . .
```

Problem Statement – 7: Accept a number and calculate the sum of its digits.

INPUT:

```c
#include <stdio.h>
int main(){
    int num, digit, sum = 0;
    printf("Enter a number: ");
    scanf("%d", &num);
    while (num > 0) {
        digit = num % 10;
        sum = sum + digit;
        num = num / 10;
    }
    printf("Sum of digits: %d", sum);
    return 0;
}
```

OUTPUT:

```
Enter a number: 148
Sum of digits: 13
--------------------------------
Process exited after 2.122 seconds with return value 0
Press any key to continue . . .
```

Q1)

Data	Data Format	C Data Type	C Variable declaration	Input Statement	Output statement

quantity
month
creditcard number
	Numeric	int
		Short int
		long int
			int quantity;
			short month;
			long ccno;
				scanf("%d",&quantity);
				scanf("%d",&month);
				scanf("%ld", &ccno);
					printf("The quantity is %d", quantity);
					printf("Credit card number is %ld, ccno);

CODE:

```c
#include <stdio.h>
int main(){
    int quantity;
    short month;
    long long int ccno;
    printf("Enter the quantity : ");
    scanf("%d",&quantity);
    printf("Enter the month : ");
    scanf("%d",&month);
    printf("Enter credit card number : ");
    scanf("%d",&ccno);
    printf("The quantity is %d\n",quantity);
    printf("Credit card number is %lld",ccno);
    return 0;
}
```

OUTPUT:

```
Input quantity:3
Input month:06
Input credit card number: 123456789876
The quantity is 0
Credit card number is 123456789876
--------------------------------
Process exited after 17.33 seconds with return value 1
Press any key to continue . . .
```

Q2)

price π	real	float double	float price; const double pi=3.141593 ;	scanf("%f",&price);	printf("The price is %5.2f", price);

CODE:

```
#include <stdio.h>
int main(){
float price;
const double pi=3.141593;
printf("Enter the price : ");
scanf("%f",&price);
printf("The price is %5.2f", price");
return 1;
}
```

OUTPUT:

```
Enter the price : 982.122
The price is 982.12
Process returned 1 (0x1)   execution time : 18.123 s
Press any key to continue.
```

Q3)

grade	character		char grade;	scanf("%c",&grade)	printf("The grade is %c",grade);

CODE:

```c
#include <stdio.h>
int main(){
char grade;
printf("Enter the grade : ");
scanf("%c",&grade);
printf("The grade is %c",grade);
return 1;
}
```

OUTPUT:

```
Enter the grade:A
The grade is A
Process returned 1 (0x1)   execution time : 12.887 s
Press any key to continue.
```

SELF ACTIVITY:

Sr_no	Principle amount	No of year	Rate of Intrest	Simple intrest
1	2000	3	3.4	204.00
2	4500	2	4.5	405.00
3	7800	6	8.3	3884.40

CODE:

```c
void main( )
{
float amount, rateOfInterest, simpleInterest;
int noOfYears;
printf("Give the Principal Sum : ");
scanf("%f",&amount);
printf("Give the Rate of Interest : ");
scanf("%f",&rateOfInterest);
printf("Give the Number of years : ");
scanf("%d",&noOfYears);
simpleInterest=amount*noOfYears*rateOfInterest / 100 ;
printf("The simple Interest on amount %7.2f for %d years at the rate %4.2f is %6.2f", amount,
noOfYears, rateOfInterest, simpleInterest);
}
```

OUTPUT - 1:

```
Give the Principal Sum : 2000
Give the Rate of Interest : 3.4
Give the Number of years : 3
The simple Interest on amount 2000.00 for 3 years at the rate 3.40 is 204.00
--------------------------------
Process exited after 19.76 seconds with return value 76
Press any key to continue . . .
```

OUTPUT - 2:

```
Give the Principal Sum : 4500
Give the Rate of Interest : 4.5
Give the Number of years : 2
The simple Interest on amount 4500.00 for 2 years at the rate 4.50 is 405.00
--------------------------------
Process exited after 6.466 seconds with return value 76
Press any key to continue . . .
```

OUTPUT - 3:

```
Give the Principal Sum : 7800
Give the Rate of Interest : 8.3
Give the Number of years : 6
The simple Interest on amount 7800.00 for 6 years at the rate 8.30 is 3884.40
--------------------------------
Process exited after 9.907 seconds with return value 77
Press any key to continue . . .
```

Set A. Apply all the three program development steps for the following examples.

1. Accept dimensions of a cylinder and print the surface area and volume
(Hint: surface area = $2\pi r^2 + 2\pi rh$, volume = $\pi r^2 h$)

Program:

```c
// CH.SC.U4CSE24015
//Accept dimensions of a cylinder and print the surface area and volume
#include <stdio.h>
int main(){
    float pi = 3.14;
    int radius; // declaring radius
    int height; // declaring height
    printf("Enter the radius : ");
    scanf("%d",&radius); // getting radius from user
    printf("Enter the height : ");
    scanf("%d",&height); // getting height from user
    float area = 2 * pi * radius * radius;
    float volume = pi * radius * radius * height;
    printf("The area is : %.2f\n",area);
    printf("The volume is %.2f",volume);
    return 1;
}
```

Output:

```
Enter the radius : 2
Enter the height : 3
The area is : 25.12
The volume is 37.68
--------------------------------
Process exited after 1.4 seconds with return value 1
Press any key to continue . . .
```

2. Accept temperatures in Fahrenheit (F) and print it in Celsius(C) and Kelvin (K)
(Hint: C=5/9(F-32), K = C + 273.15)

Program:

```c
// CH.SC.U4CSE24015
//Accept temperature in Fahrenheit(F) and print it in Celsius(C) and Kelvin(K)
#include <stdio.h>
int main(){
    float temp; // declaring temperature
    printf("Enter the temp in Fahrenheit : ");
    scanf("%f",&temp); // getting temperature from user
    float celsius = 5.0/9.0 *(temp - 32);
    float kelvin = celsius + 273.15;
    printf("The temp in celsius is : %.2f\n",celsius);
    printf("The temp in kelvin is %.2f",kelvin);
    return 1;
}
```

Output:

```
Enter the temp in Fahrenheit : 90
The temp in celsius is : 32.22
The temp in kelvin is 305.37
--------------------------------
Process exited after 0.9146 seconds with return value 1
Press any key to continue . . .
```

3. Accept initial velocity (u), acceleration (a) and time (t). Print the final velocity (v) and distance (s) travelled. (Hint: $v = u + at$, $s = u + at^2$)

Program:

```c
// CH.SC.U4CSE24015
//Accept initial velcoity(u) , acceleration(a) and time(t).
//Print final velocity(v) and distance(S) travelled.
#include <stdio.h>
void main(){
    float initial_v; // declaring initial velocity
    float acceleration; // declaring acceleration
    int time; // declaring time
    printf("Enter the initial velocity : ");
    scanf("%f",&initial_v); // getting initial velocity from user
    printf("Enter the acceleration : ");
    scanf("%f",&acceleration); // getting acceleration value from user
    printf("Enter the time : ");
    scanf("%d",&time); // getting time from user
    float final_v = initial_v + (acceleration*time);
    float distance = initial_v + (acceleration*time*time);
    printf("The final velocity is : %.2f\n",final_v);
    printf("The distance is : %.2f",distance);
}
```

Output:

```
Enter the initial velocity : 4
Enter the acceleration : 3
Enter the time : 2
The final velocity is : 10.00
The distance is : 16.00
--------------------------------
Process exited after 11.55 seconds with return value 23
Press any key to continue . . .
```

Accept inner and outer radius of a ring and print the perimeter and area of the ring

(Hint: perimeter = $2\pi(a+b)$, area = $\pi(a^2-b^2)$)

Program:

4.

```c
//CH.SC.U4CSE24015
//Finding perimeter and area of the ring
#include <stdio.h>
void main(){
	float pi=3.142;
	float inner_r,outer_r;
	printf("Enter inner radius: ");
	scanf("%f",&inner_r);
	printf("Enter outer radius: ");
	scanf("%f",&outer_r);
	float perimeter = 2*pi*(inner_r+outer_r);
	float area = pi*((inner_r*inner_r)-(outer_r*outer_r));
	printf("Perimeter: %f\n",perimeter);
	printf("Area: %f\n",area);
}
```

Output:

```
Enter inner radius: 6
Enter outer radius: 5
Perimeter: 69.124001
Area: 34.562000

Process returned 16 (0x10)   execution time : 4.070 s
Press any key to continue.
```

5. Accept two numbers and print arithmetic and harmonic mean of the two numbers (Hint: AM= (a+b)/2 , HM = ab/(a+b))

Program:

```c
//CH.SC.U4CSE24015
//Finding AM and HM of two numbers
#include <stdio.h>
void main(){
	int num1,num2;
	printf("Enter a number: ");
	scanf("%d",&num1);
	printf("Enter another number: ");
	scanf("%d",&num2);
	float AM = (num1+num2)/2;
	float HM = (num1*num2)/(num1+num2);
	printf("AM: %f\n",AM);
	printf("HM: %f\n",HM);
}
```

Output:

```
Enter a number: 101
Enter another number: 56
AM: 78.000000
HM: 36.000000

Process returned 14 (0xE)   execution time : 13.453 s
Press any key to continue.
```

6. Accept three dimensions length (l), breadth(b) and height(h) of a cuboid and print surface area and volume (Hint : surface area=2(lb+lh+bh), volume = lbh)

Program:

```c
//CH.SC.U4CSE24015
//Finding Surface area and volume of a cuboid
#include <stdio.h>
void main(){
	float l,b,h;
	printf("Enter length: ");
	scanf("%f",&l);
	printf("Enter breadth: ");
	scanf("%f",&b);
	printf("Enter height: ");
	scanf("%f",&h);
	float SA = 2*((l*b)+(l*h)+(b*h));
	float Vol = l*b*h;
	printf("Surface Area: %f\n",SA);
	printf("Volume: %f\n",Vol);
}
```

Output:

```
Enter length: 37
Enter breadth: 63
Enter height: 9
Surface Area: 6462.000000
Volume: 20979.000000

Process returned 21 (0x15)   execution time : 36.804 s
Press any key to continue.
```

7. Accept a character from the keyboard and display its previous and next character in order. Ex. If the character entered is ‘d’, display “The previous character is c”, “The next character is e”.

Program:

```c
//CH.SC.U4CSE24015
//Finding Previous and next character
#include <stdio.h>
int main() {
    char ch;
    printf("Enter a character : ");
    scanf(" %c", &ch);
    printf("The previous character is : %c\n", ch - 1);
    printf("The next character is : %c\n", ch + 1);
    return 0;
}
```

Output:

```
Enter a character : P
The previous character is : O
The next character is : Q

Process returned 0 (0x0)   execution time : 1.686 s
Press any key to continue.
```

8. Accept a character from the user and display its ASCII value.

Program:

```c
//CH.SC.U4CSE24015
//Finding ASCII value
#include <stdio.h>
int main() {
    char ch;
    printf("Enter a character: ");
    scanf(" %c", &ch);
    printf("The ASCII value of '%c' is %d\n", ch, ch);
    return 0;
}
```

Output:

```
Enter a character: D
The ASCII value of 'D' is 68

Process returned 0 (0x0)   execution time : 3.303 s
Press any key to continue.
```

Set B . Apply all the three program development steps for the following examples.

1. Accept the x and y coordinates of two points and compute the distance between the two points.

Program:

```c
//CH.SC.U4CSE24015
//Finding Distance between two points
#include <stdio.h>
int main() {
    int x1,y1,x2,y2;
    printf("Enter x1: ");
    scanf(" %d", &x1);
    printf("Enter x2: ");
    scanf(" %d", &x2);
    printf("Enter y1: ");
    scanf(" %d", &y1);
    printf("Enter y2: ");
    scanf(" %d", &y2);
    float distance = (((x1-x2)*(x1-x2))+((y1-y2)*(y1-y2)))^(1/2);
    printf("Distance between two points %f", distance);
    return 0;
}
```

Output:

```
Enter x1: 5
Enter x2: 7
Enter y1: 4
Enter y2: 9
Distance between two points 29.000000
Process returned 0 (0x0)   execution time : 9.389 s
Press any key to continue.
```

2. Accept two integers from the user and interchange them. Display the interchanged numbers.

Program:

```c
//CH.SC.U4CSE24015
//Swapping of two numbers
#include <stdio.h>
int main() {
    int a, b, temp;
    printf("Enter first number: ");
    scanf("%d", &a);
    printf("Enter second number: ");
    scanf("%d", &b);
    temp = a;
    a = b;
    b = temp;
    printf("After swapping:\n");
    printf("First number = %d\n", a);
    printf("Second number = %d\n", b);
    return 0;
}
```

Output:

```
Enter first number: 15
Enter second number: 22
After swapping:
First number = 22
Second number = 15

Process returned 0 (0x0)   execution time : 44.827 s
Press any key to continue.
```

3. A cashier has currency notes of denomination 1, 5 and 10. Accept the amount to be withdrawn from the user and print the total number of currency notes of each denomination the cashier will have to give.

Program:

```c
//CH.SC.U4CSE24015
//Segregating notes
#include <stdio.h>
int main() {
    int amount, tens, fives, ones;
    printf("Enter the amount: ");
    scanf("%d", &amount);
    tens = amount / 10;
    amount = amount % 10;
    fives = amount / 5;
    amount = amount % 5;
    ones = amount;
    printf("%d notes of 10 rupees\n", tens);
    printf("%d notes of 5 rupees\n", fives);
    printf("%d notes of 1 rupee\n", ones);
    return 1;
}
```

Output:

```
Enter the amount: 86359
8635 notes of 10 rupees
1 notes of 5 rupees
4 notes of 1 rupee

Process returned 1 (0x1)   execution time : 3.392 s
Press any key to continue.
```

Assignment Evaluation:

0: Not Done [ ] 1: Incomplete [ ] 2: Late Complete [ ]

3: Needs Improvement [ ] 4: Complete 5: Well Done [ ]

Viva Questions:

1. What is pseudocode and why is it used in problem solving?

Ans:

Pseudocode is an informal, human-readable way of describing the steps of an algorithm without using the strict rules of any specific programming language. It combines plain language with some structured elements like loops and conditionals to outline how a problem should be solved. By writing pseudocode first, you create a blueprint that can be easily translated into actual code later. It also improves collaboration among team members by providing a common understanding of the approach. Overall, pseudocode is a valuable step to organize thoughts and ensure the algorithm is correct and efficient before implementation.

2. Write pseudocode to find the largest of three numbers.

Ans:

```
start
  Input x, y, z
  If x >= y and x >= z then
     largest = x
  else if y >= x and y >= z then
     largest = y
  else
     largest = z
  end if
  Output "The largest number is : ", largest
end
```

3. How would you write a pseudocode for calculating the factorial of a number?

Ans:

```
start
  Input num
  Set fact = 1
  for i from 1 to num do
     fact = fact * i
  end for
  Output "The factorial is : ", fact
end
```


4. Explain the difference between sequence, selection, and iteration in

pseudocode.

Ans:

Sequence means steps are executed one after another in order. Every instruction follows the previous one with no branching or repeating.
Selection is decision-making, where the flow branches based on a condition. It chooses which set of instructions to execute. Usually done with If...Else if...Else statements.
Iteration means repeating a set of steps multiple times, also called looping. Common loops: For, While etc.

5. What are the advantages and limitations of using pseudocode before writing actual code?

Ans:

A
Ps
w
la
st
cc
su
la

LIMITATION
Pseudocode cannot be executed, so it cannot directly test whether the logic works correctly. There is no strict standard format for pseudocode, which can lead to inconsistencies and misunderstandings between different programmers. For complex algorithms, it can still be difficult to express all details clearly in pseudocode alone.

Activity Sheet 2 **Date:**

Decision Making Control Structures

Exercise 1:

Objective:

To demonstrate use of decision-making statements such as if and if-else.

Reading:

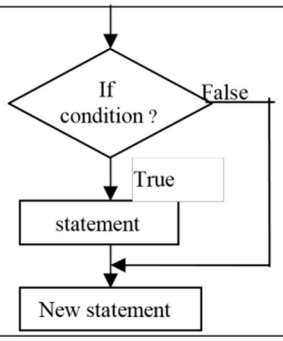
You should read following topics before starting this exercise

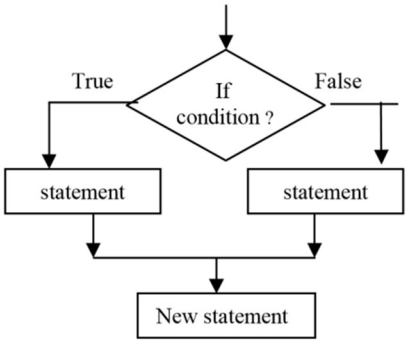
1. Different types of decision-making statements available in C.
2. Syntax for these statements.

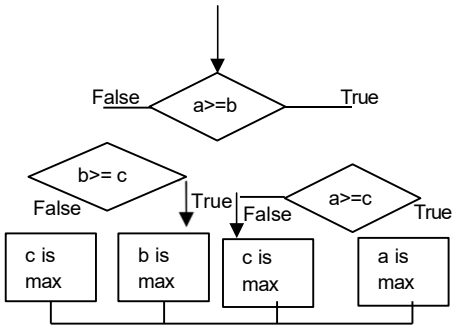
During problem solving, we come across situations when we have to choose one of the alternative paths depending upon the result of some condition. Condition is an expression evaluating to true or false. This is known as the Branching or decision-making statement. Several forms of If and else constructs are used in C to support decision-making.

1) if statements
2) if – else
3) Nested if

Sr. No	Statement Syntax	Flowchart	Example

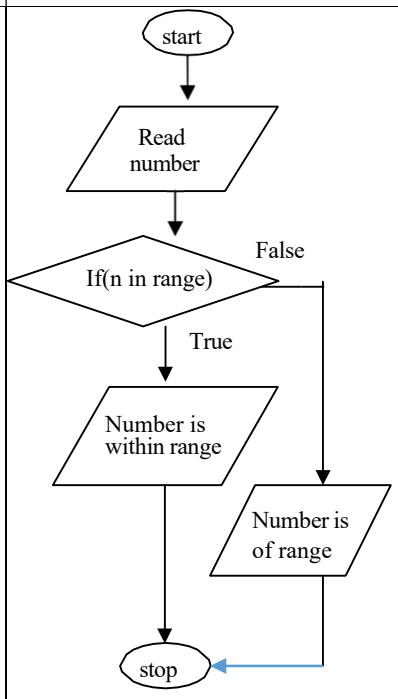
1.	**if statement**
	
	if (condition)
	{
	statement;
	}
			if(n > 0)
			printf("Number is positive");

2.	**if - else statement**
	
	if (condition)
	{
	statement;
	}
	else
	{
	statement;
	}
			if(n % 2 == 0)
			printf("Even");
			else
			printf("Odd");

3.	Nested if
	
	if (condition)
	{
	if
	(condition) {
	statement;}
	else
	{ statement;}
	
	}
	else
	{
	if (condition)
	{ statement;
	} else
	{ statement; }
	
	}
			If (a >= b)
			{
			if (a >= c)
			printf(" %d is maximum",a);
			else
			printf(" %d is maximum",c);
			}
			else
			{
			if (b >= c) printf(" %d is maximum",b);
			else
			printf(" %d is maximum",c);
			}

- Sample program- to check whether a number is within range.

Step 1: Writing the Algorithm	**Step 2: Draw the flowchart**	**Step 3: Writing Program**

Algorithm Range
Begin
Input llimit, ulimit
Input num
If n≥llimit and n≤ulimit then
Output "in range"
Else
Output "not in range"
End
		#include <stdio.h>
		main( )
		{
		/* variable declarations */
		int n;
		int llimit=50, ulimit = 100;
		/* prompting and accepting input */
		printf("Enter the number");
		scanf("%d",&n);
		if(n>=llimit && n <= ulimit)
		printf("Number is within range");
		else
		printf("Number is out of range");
		}

Self Activity

1. Execute the following program for five different values and fill in the adjoining table

```
main()
{
 int n;
  printf("Enter no.");
 scanf("%d",&n);
  if(n%___==0)
     printf("divisible);
 else
   printf("not divisible ");
 }
```

n	output

2. Type the above sample program 4 and execute it for the following values.

n	Output message
50	
100	
65	
____	
____	

3. Using the sample code 3 above write the complete program to find the maximum of three numbers and execute it for different set of values.

Set A: Apply all the three program development steps for the following examples.

1. Write a program to accept an integer and check if it is even or odd.

Program:

```c
// ch.sc.u4cse24015
// Check whether a number is even or odd using if-else
#include <stdio.h>

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

    if (num % 2 == 0)
        printf("%d is even.\n", num);
    else
        printf("%d is odd.\n", num);

    return 0;
}
```

Output:

```
Enter a number: 345
345 is odd.

Process returned 0 (0x0)   execution time : 8.604 s
Press any key to continue.
```

2. Write a program to accept three numbers and check whether the first is between the other two numbers. Ex: Input 20 10 30. Output: 20 is between 10 and 30

Program:

```c
// ch.sc.u4cse24015
// Check whether first number is between the other two
#include <stdio.h>

int main() {
    int a, b, c;
    printf("Enter three numbers (a b c): ");
    scanf("%d %d %d", &a, &b, &c);

    if ((a > b && a < c) || (a > c && a < b))
        printf("%d is between %d and %d.\n", a, b, c);
    else
        printf("%d is not between %d and %d.\n", a, b, c);

    return 0;
}
```

Output:

```
Enter three numbers (a b c): 1 7 86
1 is not between 7 and 86.

Process returned 0 (0x0)   execution time : 6.500 s
Press any key to continue.
```

3. Accept a character as input and check whether the character is a digit.
(Check if it is in the range ‘0’ to ‘9’ both inclusive)

Program:

```c
// ch.sc.u4cse24015
// Check whether character is a digit
#include <stdio.h>

int main() {
    char ch;
    printf("Enter a character: ");
    scanf(" %c", &ch); // space before %c avoids newline issue

    if (ch >= '0' && ch <= '9')
        printf("'%c' is a digit.\n", ch);
    else
        printf("'%c' is not a digit.\n", ch);

    return 0;
}
```

Output:

```
Enter a character: D
'D' is not a digit.

Process returned 0 (0x0)   execution time : 3.789 s
Press any key to continue.
```

4. Write a program to accept a number and check if it is divisible by 5 and 7.

Program:

```c
// ch.sc.u4cse24015
// Check divisibility by both 5 and 7
#include <stdio.h>

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

    if (num % 5 == 0 && num % 7 == 0)
        printf("%d is divisible by both 5 and 7.\n", num);
    else
        printf("%d is not divisible by both 5 and 7.\n", num);

    return 0;
}
```

Output:

```
Enter a number: 47
47 is not divisible by both 5 and 7.

Process returned 0 (0x0)   execution time : 3.985 s
Press any key to continue.
```

5. Write a program, which accepts annual basic salary of an employee and calculates and displays the Income tax as per the following rules. ***

Basic: < 1,50,000	Tax = 0
1,50,000 to 3,00,000	Tax = 20%
> 3,00,000 Tax = 30%	

Program:

```c
// ch.sc.u4cse24015
// Calculate income tax using salary ranges
#include <stdio.h>

int main() {
    float salary, tax;
    printf("Enter your salary: ");
    scanf("%f", &salary);

    if (salary < 150000)
        tax = 0;
    else if (salary <= 300000)
        tax = 0.2 * salary;
    else
        tax = 0.3 * salary;

    printf("Income tax to be paid: ?%.2f\n", tax);

    return 0;
}
```

Output:

```
Enter your salary: 200000
Income tax to be paid: ?40000.00

Process returned 0 (0x0)   execution time : 2.508 s
Press any key to continue.
```

6. Accept a character from the user and check whether the character is a vowel or consonant. (Hint: a,e,i,o,u, A, E, I, O, U are vowels)

Program:

```c
// ch.sc.u4cse24015
// Check if a character is a vowel
#include <stdio.h>

int main() {
    char ch;
    printf("Enter an alphabet: ");
    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 a vowel.\n", ch);
    else
        printf("'%c' is a consonant.\n", ch);

    return 0;
}
```

Output:

```
Enter an alphabet: E
'E' is a vowel.

Process returned 0 (0x0)   execution time : 1.885 s
Press any key to continue.
```

7. Accept any year as input through the keyboard. Write a program to check whether the year is a leap year or not. (Hint leap year is divisible by 4 and not by 100 or divisible by 400)

Program:

```c
// ch.sc.u4cse24015
// Check whether a year is a leap year
#include <stdio.h>

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

    if ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0)
        printf("%d is a leap year.\n", year);
    else
        printf("%d is not a leap year.\n", year);

    return 0;
}
```

Output:

```
Enter a year: 2025
2025 is not a leap year.

Process returned 0 (0x0)   execution time : 2.441 s
Press any key to continue.
```

Set B: Apply all the three program development steps for the following examples.

1. Write a program to check whether given character is a digit or a character in lowercase or uppercase alphabet. (Hint ASCII value of digit is between 48 to 58 and Lowercase characters have ASCII values in the range of 97 to122, uppercase is between 65 and 90)

Program:

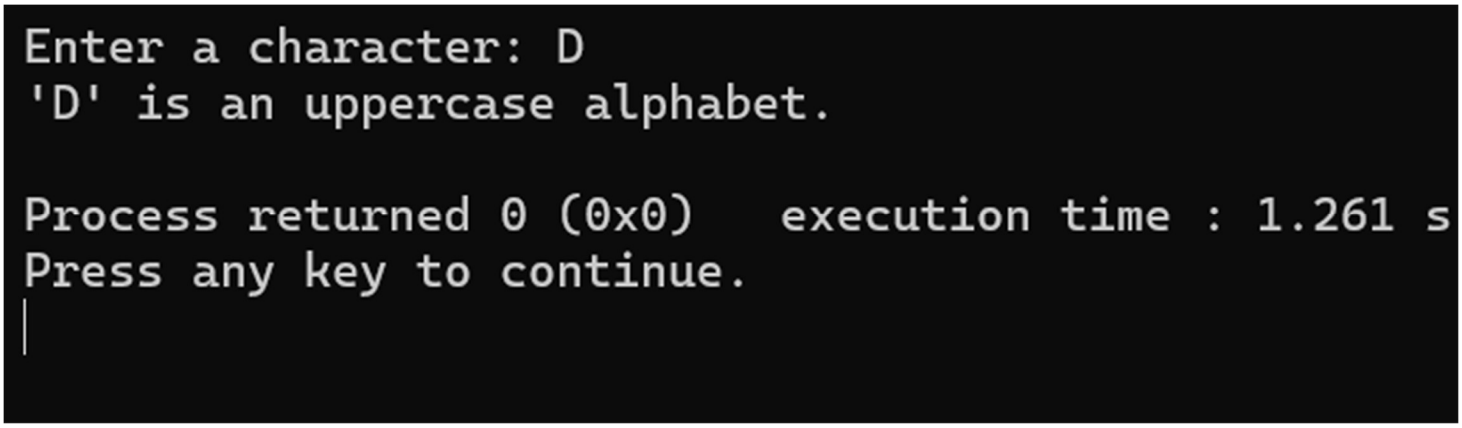
```c
// ch.sc.u4cse24015
// Identify character type using if-else
#include <stdio.h>

int main() {
    char ch;
    printf("Enter a character: ");
    scanf(" %c", &ch);

    if (ch >= '0' && ch <= '9')
        printf("'%c' is a digit.\n", ch);
    else if (ch >= 'a' && ch <= 'z')
        printf("'%c' is a lowercase alphabet.\n", ch);
    else if (ch >= 'A' && ch <= 'Z')
        printf("'%c' is an uppercase alphabet.\n", ch);
    else
        printf("'%c' is a special symbol.\n", ch);

    return 0;
}
```

Output:


Enter a character: D
'D' is an uppercase alphabet.
Process returned 0 (0x0)   execution time : 1.261 s
Press any key to continue.

2. Accept the x and y coordinate of a point and find the quadrant in which the point lies.

Program:

```c
// ch.sc.u4cse24015
// Determine the quadrant of a point
#include <stdio.h>

int main() {
    int x, y;
    printf("Enter x and y coordinates: ");
    scanf("%d %d", &x, &y);

    if (x > 0 && y > 0)
        printf("Point lies in the First quadrant.\n");
    else if (x < 0 && y > 0)
        printf("Point lies in the Second quadrant.\n");
    else if (x < 0 && y < 0)
        printf("Point lies in the Third quadrant.\n");
    else if (x > 0 && y < 0)
        printf("Point lies in the Fourth quadrant.\n");
    else if (x == 0 && y == 0)
        printf("Point is at the Origin.\n");
    else
        printf("Point lies on an axis.\n");

    return 0;
}
```

Output:

```
Enter x and y coordinates: 88 69
Point lies in the First quadrant.

Process returned 0 (0x0)   execution time : 6.321 s
Press any key to continue.
```

3. Write a program to calculate the roots of a quadratic equation. Consider all possible cases. Accept the cost price and selling price from the keyboard. Find out if the seller has made a profit or loss and display how much profit or loss has been made.

Program:

```c
// ch.sc.u4cse24015
// Calculate profit or loss
#include <stdio.h>

int main() {
    float cost, selling;
    printf("Enter cost price: ");
    scanf("%f", &cost);
    printf("Enter selling price: ");
    scanf("%f", &selling);

    if (selling > cost)
        printf("Profit: ?%.2f\n", selling - cost);
    else if (cost > selling)
        printf("Loss: ?%.2f\n", cost - selling);
    else
        printf("No profit, no loss.\n");

    return 0;
}
```

Output:

```
Enter cost price: 2000
Enter selling price: 2200
Profit: ?200.00

Process returned 0 (0x0)   execution time : 7.241 s
Press any key to continue.
```

4. Write a program to accept marks for three subjects. Calculate the average and also display the class obtained. (Distinction – above _____Class I – above _%, class II – ___% to ___%, pass class – ___% to ___% and fail otherwise)

Program:

```c
// ch.sc.u4cse24015
// Calculate average marks and determine class
#include <stdio.h>

int main() {
    float m1, m2, m3, avg;
    printf("Enter marks for 3 subjects: ");
    scanf("%f %f %f", &m1, &m2, &m3);

    avg = (m1 + m2 + m3) / 3;
    printf("Average = %.2f\n", avg);

    if (avg >= 75)
        printf("Class: Distinction\n");
    else if (avg >= 60)
        printf("Class: First Class\n");
    else if (avg >= 50)
        printf("Class: Second Class\n");
    else if (avg >= 40)
        printf("Class: Pass Class\n");
    else
        printf("Class: Fail\n");

    return 0;
}
```

Output:

```
Enter marks for 3 subjects: 100 98 99
Average = 99.00
Class: Distinction

Process returned 0 (0x0)   execution time : 4.992 s
Press any key to continue.
```

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete	5: Well Done [ ]

Viva Questions

1. Give an application of nested if else statements in C Programming.
   ANS:
   An application of nested if-else statements in C programming can be seen in the development of a **student grading system**. In such a system, marks obtained by a student are evaluated to assign a grade based on specific ranges. For example, if the marks are 90 or above, the student receives an 'A'; if the marks are between 75 and 89, they receive a 'B'; and so on. This type of logic requires checking multiple conditions that are dependent on one another, making nested if-else structures ideal.

2. How would you handle missed cases in conditional statements.
   ANS:
   Missed cases in conditional statements can be handled by using a **default or fallback condition**, typically done using the else clause or a default case in switch statements. This ensures that even if none of the specific conditions are met, the program still executes a defined block of code, preventing unexpected behavior or program crashes.

3. Can you describe a situation where the use of a ternary operator would be more appropriate than a traditional if else statement.
   A ternary operator is most appropriate in situations where a simple condition needs to choose between two values, and writing a full if-else block would be unnecessarily verbose.
   Eg:
   age >= 18 ? printf("Eligible") : printf("Not Eligible");

   Need a compact syntax for assignment or output.
   The logic is simple.

4. How do you incorporate error handling within your conditional statement.

   ANS:

Error handling within conditional statements is done by checking for invalid inputs, unexpected conditions, or failures, and then executing appropriate actions such as displaying error messages, exiting the program, or prompting for re-entry.

Date:

Exercise 2:

Objective:

To demonstrate decision making statements (switch case)

Reading:

You should read following topics before starting this exercise

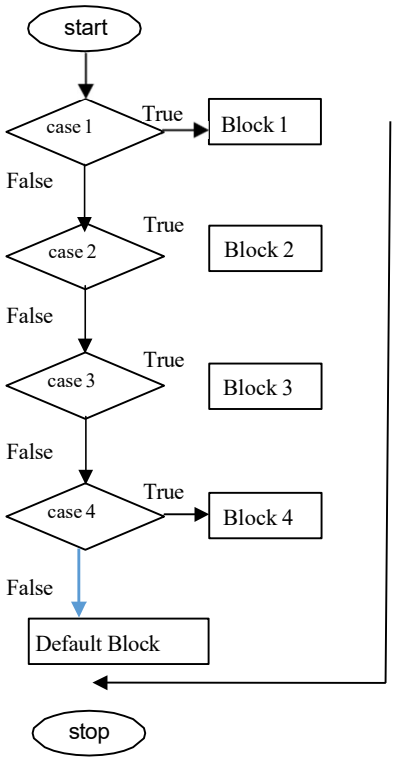
1. Different types of decision-making statements available in C.
2. Syntax for switch case statements.

The control statement that allows us to make a decision from the number of choices is called a switchcase statement. It is a multi-way decision making statement.

1. **Usage of switch statement**

Statement Syntax	Flowchart	Example

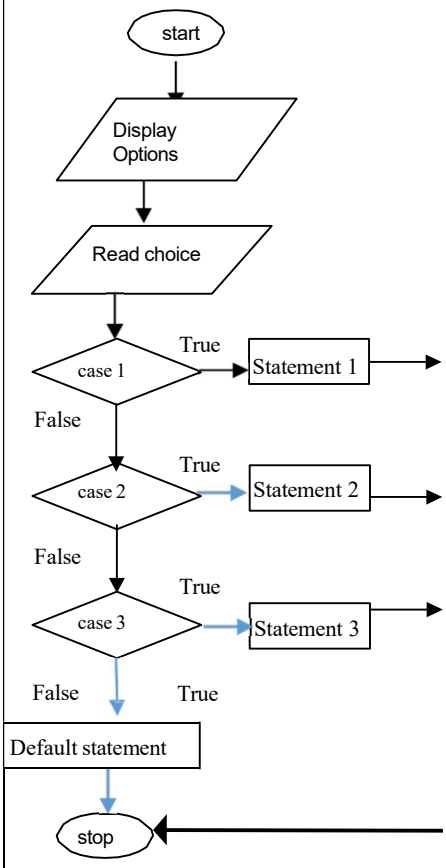
```
switch(expression)
{
case value1: block1;
                break;
case value2: block2;
                break;
.
.
.
default: default block;
break;
}
```




```
switch (color)
{
 case 'r' : case 'R' :
          printf ("RED");
          break;
 case 'g' :    case 'G' :
          printf ("GREEN");
           break;
 case 'b' :    case 'B' :
           printf ("BLUE");
           break;
 default :
       printf ("INVALID COLOR");
}
```

2. The switch statement is used in writing menu driven programs where a menu displays several options and the user gives the choice by typing a character or number. A Sample program to display the selected option from a menu is given below.

Step 1: Writing the Algorithm	**Step 2: Draw the flowchart**	**Step 3: Writing Program**

Algorithm Menu
Begin
Output menu
Read choice
Execute statements depending on choice
End
		#include <stdio.h>
		main( )
		{
		/* variable declarations */
		int choice;
		/* Displaying the Menu */
		printf("\n 1. Option 1\n");
		printf(" 2. Option 2\n");
		printf(" 3. Option 3\n");
		printf("Enter your choice");
		scanf("%d",&choice);
		switch(choice)
		{
		case 1: printf("Option 1 is selected");
		break;
		case 2: printf("Option 2 is selected");
		break;
		case 3: printf("Option 3 is selected");
		break;
		default: printf("Invalid choice");
		}
		}

Self Activity

1. Write the program that accepts a char–type variable called color and displays appropriate message using the sample code 1 above. Execute the program for various character values and fill in the following table. Modify the program to include all rainbow colours

Input character	Output Message
V	
I	
B	
g	
R	
y	

Set A: Apply all the three program development steps for the following examples.

1. Accept a single digit from the user and display it in words.

   For example, if digit entered is 9, display Nine.

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
int main() {
    int digit;
    printf("Enter a single digit (0-9): ");
    scanf("%d", &digit);
    if (digit < 0 || digit > 9) {
        printf("Invalid input! Please enter a digit between 0 and 9.\n");
        return 1;
    }
    printf("The digit in words is: ");
    switch(digit) {
        case 0: printf("Zero\n"); break;
        case 1: printf("One\n"); break;
        case 2: printf("Two\n"); break;
        case 3: printf("Three\n"); break;
        case 4: printf("Four\n"); break;
        case 5: printf("Five\n"); break;
        case 6: printf("Six\n"); break;
        case 7: printf("Seven\n"); break;
        case 8: printf("Eight\n"); break;
        case 9: printf("Nine\n"); break;
    }
    return 1;
}
```

Output:

```
Enter a single digit (0-9): 8
The digit in words is: Eight

Process returned 1 (0x1)   execution time : 1.040 s
Press any key to continue.
```

2. Write a program, which accepts two integers and an operator as a character (+ - * /), performs the corresponding operation and displays the result.

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
int main() {
    int num1, num2;
    char op;
    printf("Enter first integer: ");
    scanf("%d", &num1);
    printf("Enter an operator (+, -, *, /): ");
    scanf(" %c", &op);
    printf("Enter second integer: ");
    scanf("%d", &num2);
    switch(op) {
        case '+':
            printf("Result: %d\n", num1 + num2);
            break;
        case '-':
            printf("Result: %d\n", num1 - num2);
            break;
        case '*':
            printf("Result: %d\n", num1 * num2);
            break;
        case '/':
            if (num2 == 0) {
                printf("Error: Division by zero is not allowed.\n");
            } else {
                printf("Result: %d\n", num1 / num2);
            }
            break;
        default:
            printf("Invalid operator!\n");
    }
    return 0;
}
```

Output:

```
Enter first integer: 9
Enter an operator (+, -, *, /): +
Enter second integer: 6
Result: 15

Process returned 0 (0x0)   execution time : 7.948 s
Press any key to continue.
```

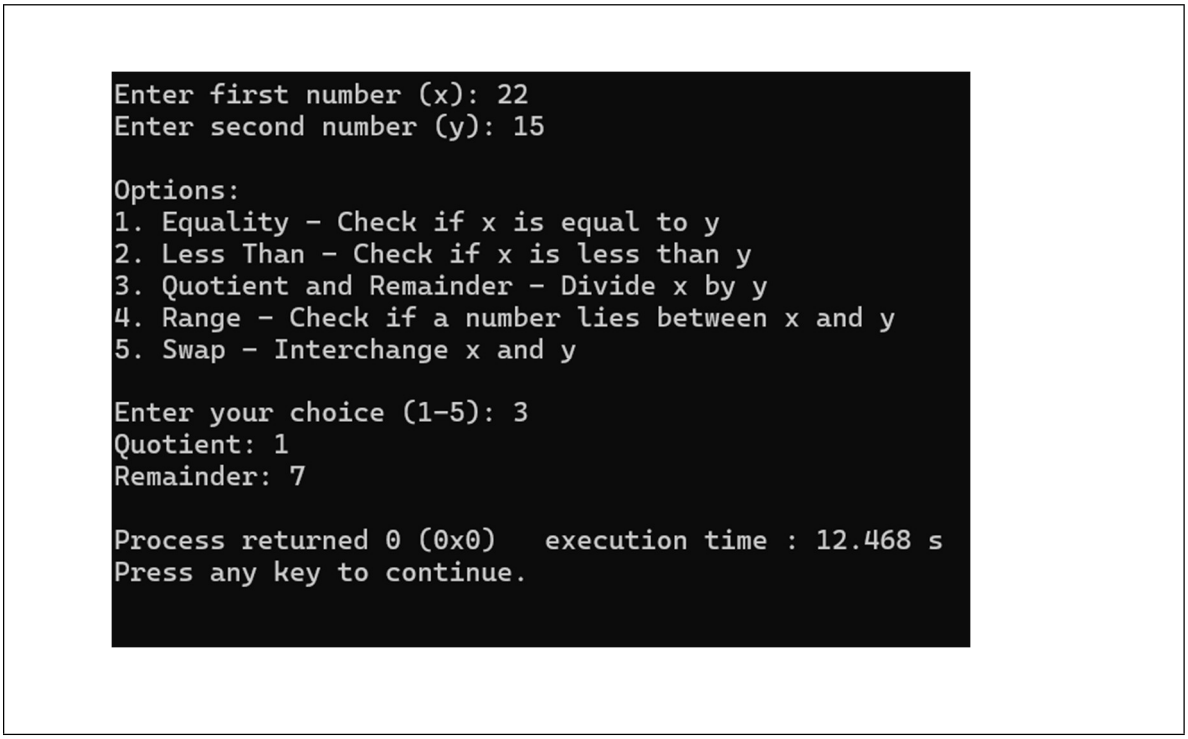
3. Accept two numbers in variables x and y from the user and perform the following operations

Options	Actions
1. Equality	Check if x is equal to y
2. Less Than	Check if x is less than y
3. Quotient and Remainder	Divide x by y and display the quotient and remainder
4. Range	Accept a number and check if it lies between x and y (both inclusive)
5. Swap	Interchange x and y

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
int main() {
    int x, y, choice, num;
    printf("Enter first number (x): ");
    scanf("%d", &x);
    printf("Enter second number (y): ");
    scanf("%d", &y);
    printf("\nOptions:\n");
    printf("1. Equality - Check if x is equal to y\n");
    printf("2. Less Than - Check if x is less than y\n");
    printf("3. Quotient and Remainder - Divide x by y\n");
    printf("4. Range - Check if a number lies between x and y\n");
    printf("5. Swap - Interchange x and y\n");
    printf("\nEnter your choice (1-5): ");
    scanf("%d", &choice);
    switch (choice) {
        case 1:
            if (x == y) {
                printf("x is equal to y.\n");
            } else {
                printf("x is not equal to y.\n");
            }
            break;
        case 2:
            if (x < y) {
                printf("x is less than y.\n");
            } else {
                printf("x is not less than y.\n");
            }
            break;
        case 3:
            if (y == 0) {
                printf("Error: Division by zero is not allowed.\n");
            } else {
                printf("Quotient: %d\n", x / y);
                printf("Remainder: %d\n", x % y);
            }
            break;
        case 4:
            printf("Enter the number to check range: ");
            scanf("%d", &num);
            if ((num >= x && num <= y) || (num >= y && num <= x)) {
                printf("%d lies between %d and %d (inclusive).\n", num, x, y);
            } else {
                printf("%d does not lie between %d and %d.\n", num, x, y);
            }
            break;
        case 5: {
            int temp = x;
            x = y;
            y = temp;
            printf("After swapping: x = %d, y = %d\n", x, y);
            break;
        }
        default:
            printf("Invalid choice.\n");
    }
    return 0;
}
```

Output:


Enter first number (x): 22
Enter second number (y): 15
Options:
1. Equality - Check if x is equal to y
2. Less Than - Check if x is less than y
3. Quotient and Remainder - Divide x by y
4. Range - Check if a number lies between x and y
5. Swap - Interchange x and y
Enter your choice (1-5): 3
Quotient: 1
Remainder: 7
Process returned 0 (0x0) execution time : 12.468 s
Press any key to continue.

Set B: Apply all the three program development steps for the following examples

1. Accept radius from the user and write a program having menu with the following options and corresponding actions

Options	Actions
1. Area of Circle	Compute area of circle and print
2. Circumference of Circle	Compute Circumference of circle and print
3. Volume of Sphere	Compute Volume of Sphere and print

Program:

```c
//CH.SC.U4CSE2015
#include <stdio.h>
int main() {
    float PI = 3.14;
    int choice;
    float radius, area, circumference, volume;
    printf("Enter the radius: ");
    scanf("%f", &radius);
    if (radius < 0) {
        printf("Radius cannot be negative.\n");
        return 1;
    }
    printf("\nOptions:\n");
    printf("1. Area of Circle\n");
    printf("2. Circumference of Circle\n");
    printf("3. Volume of Sphere\n");
    printf("\nEnter your choice (1-3): ");
    scanf("%d", &choice);
    switch (choice) {
        case 1:
            area = PI * radius * radius;
            printf("Area of Circle: %.2f\n", area);
            break;
        case 2:
            circumference = 2 * PI * radius;
            printf("Circumference of Circle: %.2f\n", circumference);
            break;
        case 3:
            volume = (4.0/3.0) * PI * radius * radius * radius;
            printf("Volume of Sphere: %.2f\n", volume);
            break;
        default:
            printf("Invalid choice.\n");
    }
    return 0;
}
```

Output:

```
Enter the radius: 15

Options:
1. Area of Circle
2. Circumference of Circle
3. Volume of Sphere

Enter your choice (1-3): 2
Circumference of Circle: 94.20

Process returned 0 (0x0)   execution time : 17.115 s
Press any key to continue.
```

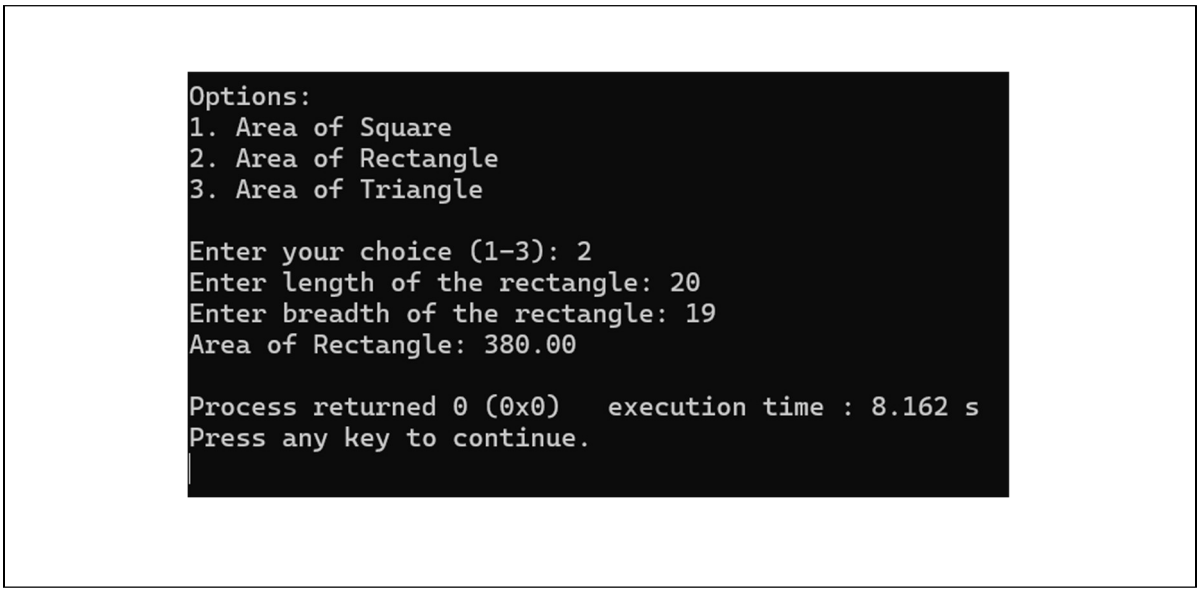
2. Write a program having a menu with the following options and corresponding actions

Options	Actions
1. Area of square	Accept length, Compute area of square and print
2. Area of Rectangle	Accept length and breadth, Compute area of rectangle and print
3. Area of triangle	Accept base and height, Compute area of triangle and print

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
int main() {
    int choice;
    float length, breadth, base, height, area;
    printf("Options:\n");
    printf("1. Area of Square\n");
    printf("2. Area of Rectangle\n");
    printf("3. Area of Triangle\n");
    printf("\nEnter your choice (1-3): ");
    scanf("%d", &choice);
    switch (choice) {
        case 1:
            printf("Enter length of the square: ");
            scanf("%f", &length);
            area = length * length;
            printf("Area of Square: %.2f\n", area);
            break;
        case 2:
            printf("Enter length of the rectangle: ");
            scanf("%f", &length);
            printf("Enter breadth of the rectangle: ");
            scanf("%f", &breadth);
            area = length * breadth;
            printf("Area of Rectangle: %.2f\n", area);
            break;
        case 3:
            printf("Enter base of the triangle: ");
            scanf("%f", &base);
            printf("Enter height of the triangle: ");
            scanf("%f", &height);
            area = 0.5 * base * height;
            printf("Area of Triangle: %.2f\n", area);
            break;
        default:
            printf("Invalid choice.\n");
    }
    return 0;}
```

Output:



Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete	5: Well Done [ ]

Activity Sheet 3 **Date:**

Loop Control Structures

Exercise 1:

Objective:

To demonstrate use of simple loops.

Reading:

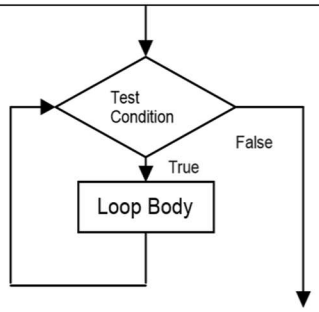
You should read following topics before starting this exercise

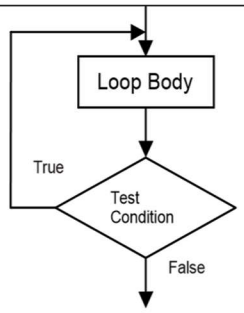
1. Different types of loop structures in C.
2. Syntax and usage of these statements.

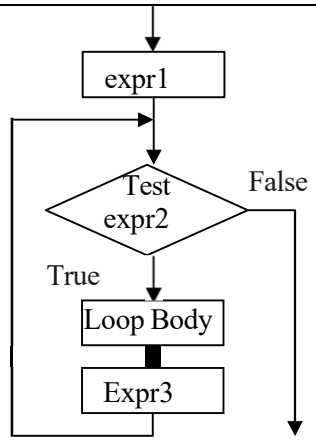
We need to perform certain actions repeatedly for a fixed number of times or till some condition holds true. These repetitive operations are done using loop control statements. The types of loop structures supported in C are

1. while statement
2. do-while statement
3. for statement

Sr. No	Statement Syntax	Flowchart	Example

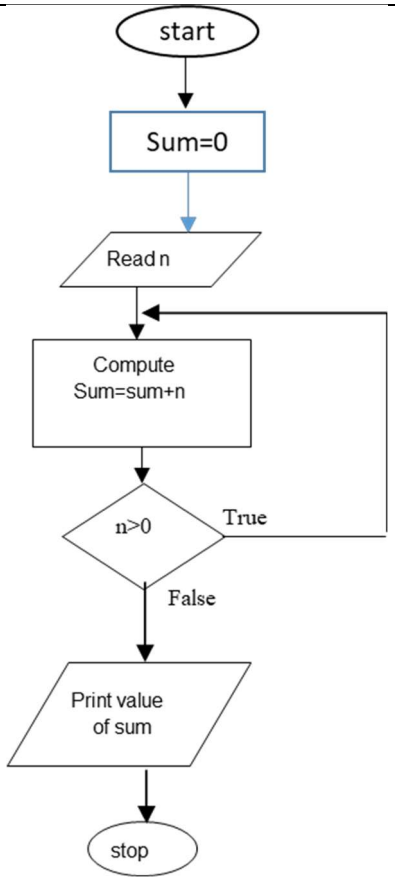
1.	**while statement**
	
	while (condition)
	{
	statement 1;
	statement 2;
	.
	.
	}
			/* accept a number*/
			scanf("%d", &n);
			/* if not a single digit */
			while (n > 9)
			{
			/* remove last digit
			n = n /10;
			}

2.	**do-while statement**
	
	do
	{
	statement 1;
	statement 2;
	.
	.
	} while (condition);
			/*initialize sum*/
			sum =0;
			do
			{/* Get a number */
			printf(" give number"};
			scanf("%d",&n);
			/* add number to sum*/
			sum=sum+n;
			} while (n>0);
			printf ("sum is %d", sum);

3.	**for statement**
	
	for (expr1; expr2; expr3)
	{
	statement 1
	.
	.
	}
	expr1 = initialization expression
	expr2 = loop condition
	expr3 = alteration expression which alters the loop variable
			/* display first 10 multiples of 2 */
			
			{
			for(i=1; i <= 10; i++)
			printf ("2 X %d = %d\n", i, 2*i);
			}

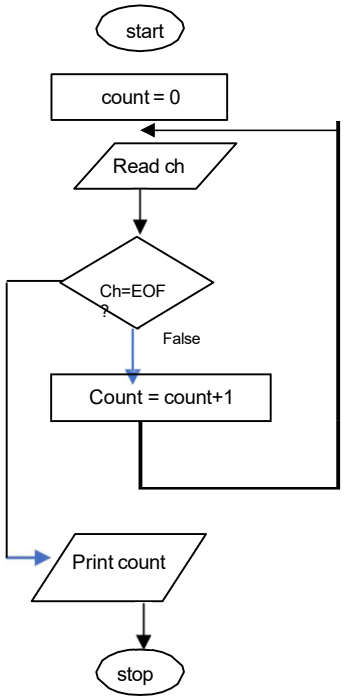
3. Sample program- to print sum of 1+2+3+….n.

Step 1: Writing the Algorithm	**Step 2: Draw the flowchart**	**Step 3: Writing Program**

Algorithm SumofN
Begin
Input n
Sum=0
While n>0 do
sum-sum+n
n=n-1
Output sum
End
		#include <stdio.h>
		main( )
		{
		/* variable declarations */
		int sum = 0, n;
		printf("enter the value of n : ");
		scanf("%d",&n);
		while (n>0)
		{
		sum = sum + n;
		n--;
		}
		printf("\n The sum of numbers is %d", sum);
		}

4. Sample program- To read characters till EOF (Ctrl+Z) and count the total number of characters entered.

Step 1 : Writing the Algorithm	Step 2 : Draw the flowchart	Step 3 : Writing Program

Algorithm CharCount
Begin
Count=0
Input ch
While ch ≠EOF do
Count=count+1
Input ch
Output count
End
		#include <stdio.h> main( )
		{
		char ch; int count=0;
		while((ch=getchar())!=EOF)
		count++;
		printf("Total characters =%d", count);
		}

Self-Activity

1. Execute example 1 given above. Execute the program for different values.
2. Write a program that accepts numbers continuously as long as the number is positive and prints the sum of the numbers read. Refer example code 2 given above.
   Execute the program for different values.
3. Write a program to accept n and display its multiplication table. Refer to sample code 3 given above.
4. Type the sample program to print sum of first n numbers and execute the program for different values of n.
5. Write a program to accept characters till the user enters EOF and count number of times 'a' Is entered. Refer to sample program 5 given above.

Set A . Apply all the three program development steps for the following examples.

1. Write a program to accept an integer n and display all even numbers upto n.

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
int main(){
    int i,n;
    scanf("%d",&n);
    for(i = 0 ; i<n ; i+=2){
        printf("\n%i",i);
    }
    return 1;
}
```

Output:

```
10

0
2
4
6
8
--------------------------------
Process exited after 6.21 seconds with return value 1
Press any key to continue . . .
```

2. Accept two integers x and y and calculate the sum of all integers between x and y (both inclusive)

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
int main(){
    int i,x,y,sum = 0;
    scanf("%d%d",&x,&y);
    for(i=x;i<=y;i++){
        sum=sum+i;
    }
    printf("%i",sum);
    return 0;
}
```

Output:

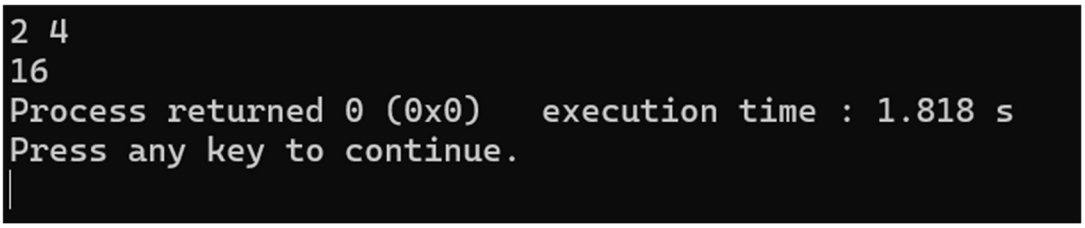
```
1 10
55
Process returned 0 (0x0)   execution time : 5.104 s
Press any key to continue.
```

3. Write a program to accept two integers x and n and compute x^n

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
int main() {
    int i, n, x, exp = 1;
    scanf("%d%d", &x, &n);
    for(i = 0; i < n; i++) {
        exp *= x;
    }
    printf("%i", exp);
    return 0;
}
```

Output:



4. Write a program to accept a character, an integer n and display the next n characters.

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
int main(){
    char chr;
    int i,n;
    scanf("%c%d",&chr,&n);
    for(i = 1 ; i<=n ; i++){
        printf("%c\n", chr+i);
    }
    return 1;
}
```

Output:

```
a 5
b
c
d
e
f

Process returned 1 (0x1)   execution time : 1.641 s
Press any key to continue.
```

5. Write a program to accept an integer and check if it is prime or not.

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
int main() {
    int i,n,prime=1;
    scanf("%d", &n);
    if(n == 1) {
        printf("Neither Prime nor Composite");
    }
    else {
        for(i = 2; i <= n / 2; i++) {
            if(n % i == 0) {
                prime = 0;
                break;
            }
        }
        if(prime)
            printf("prime number");
        else
            printf("Not a prime number");
    }
    return 0;
}
```

Output:

```
9
Not a prime number
Process returned 0 (0x0)   execution time : 2.448 s
Press any key to continue.
```

6. Write a program to accept an integer, count number of digits and calculate sum of digits in the number. Example: Number = 1234 Output: Digits = 4, Sum = 10

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
int main() {
    int n, digit = 0, sum = 0;
    scanf("%d", &n);
    if (n == 0) {
        digit = 1;
        sum = 0;
    } else {
        while (n > 0) {
            sum += n % 10;
            digit++;
            n = n / 10;
        }
    }
    printf("Digit count is : %d\n",digit);
    printf("Sum of digits is : %d\n",sum);
    return 1;
}
```

Output:

```
1234
Digit count is : 4
Sum of digits is : 10

Process returned 1 (0x1)   execution time : 3.720 s
Press any key to continue.
```

7. Write a program to accept an integer and reverse the number. Example: Input:

546, Reverse = 645.

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
int main(){
    int num,rev=0,temp;
    printf("Enter the number : ");
    scanf("%d",&num);
    temp = num;
    while(num>0){
        int digit = num % 10;
        rev = rev*10 + digit;
        num = num/10;
    }
    printf("Original number : %i\n",temp);
    printf("Reverse : %i",rev);
    return 1;
}
```

Output:

```
Enter the number : 456
Original number : 456
Reverse : 654
Process returned 1 (0x1)   execution time : 2.544 s
Press any key to continue.
```

Set B. Apply all the three program development steps for the following examples.

1. Write a program to display the first n Fibonacci numbers. (1 1 2 3 5 ……)

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
int main() {
    int n, i;
    int a=1,b=1,c;
    printf("Enter no of terms : ");
    scanf("%d", &n);
    for (i=1;i<=n;i++) {
        if(i==1||i==2) {
            c=1;
        }
        else{
            c=a+b;
            a=b;
            b=c;
        }
        printf("%d ",c);
    }
    return 1;
}
```

Output:

```
Enter no of terms : 10
1 1 2 3 5 8 13 21 34 55
Process returned 1 (0x1)   execution time : 13.330 s
Press any key to continue.
```

2. Write a program to accept real number x and integer n and calculate the sum of first n terms of the series x+ 3x+5x+7x+…

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
int main(){
    float x,sum=0;
    int i,n;
    printf("Enter the number : ");
    scanf("%f",&x);
    printf("Enter no of terms : ");
    scanf("%d",&n);
    for (i=0;i<n;i++) {
        sum+=(2*i+1)*x;
    }
    printf("Sum : %.2f\n",sum);
    return 1;
}
```

Output:

```
Enter the number : 5
Enter no of terms : 3
Sum : 45.00

Process returned 1 (0x1)   execution time : 2.138 s
Press any key to continue.
```

3. Write a program to accept real number x and integer n to calculate sum of first n terms of the Series $\frac{1}{1}x + \frac{2}{2}x^2 + \frac{3}{3}x^3 + \ldots\ldots$

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
#include <math.h>
int main(){
    float x,sum=0;
    int i,n;
    printf("Enter the number : ");
    scanf("%f",&x);
    printf("Enter no of terms : ");
    scanf("%d",&n);
    for (i=1;i<=n;i++){
        sum += pow(x,i);
    }
    printf("Sum : %.2f\n",sum);
    return 1;
}
```

Output:

```
Enter the number : 2
Enter no of terms : 4
Sum : 30.00

Process returned 1 (0x1)   execution time : 2.490 s
Press any key to continue.
```

4. Write a program to accept characters till the user enters EOF and count number of alphabets and digits entered. Refer to sample program 5 given above.

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
int main(){
    char ch;
    int alphabets = 0, digits = 0;
    printf("Enter characters : ");
    while (scanf("%c", &ch) == 1){
        if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z')) {
            alphabets++;
        }
        else if (ch >= '0' && ch <= '9'){
            digits++;
        }
    }
    printf("Number of alphabets: %d\n", alphabets);
    printf("Number of digits: %d\n", digits);
    return 0;
}
```

Output:

```
Enter characters : ofj125gqeg3

^Z

Number of alphabets: 7
Number of digits: 4

Process returned 0 (0x0)   execution time : 6.347 s
Press any key to continue.
```

5. Write a program, which accepts a number n and displays each digit in words. Example: 6702

Output = Six-Seven-Zero-Two.

(Hint: Reverse the number and use a switch statement)

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>

int main() {
    int num,rev = 0,digit;
    printf("Enter the number : ");
    scanf("%d",&num);
    int temp = num;
    while (temp != 0) {
        rev = rev * 10 + temp % 10;
        temp /= 10;
    }
    while (rev != 0) {
        digit = rev % 10;
        switch (digit) {
            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;
        }
        rev /= 10;
        if (rev != 0) {
            printf("-");
        }
    }
    printf("\n");
    return 1;
}
```

Output:

```
Enter the number : 4736
Four-Seven-Three-Six

Process returned 1 (0x1)   execution time : 19.333 s
Press any key to continue.
```

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Exercise 2 **Date:**

Objective:

To demonstrate use of nested loops

Reading

In the previous exercise, you used while, do-while and for loops. You should read following topics before starting this exercise

1. Different types of loop structures in C.
2. Syntax for these statements.
3. Usage of each loop structure

Nested loop means a loop that is contained within another loop. Nesting can be done upto any levels. The inner loop has to be completely enclosed in the outer loop. No overlapping of loops is allowed.

Sr. No	Format	Sample Program

1.	Nested for loop
	
	for(exp1; exp2 ; exp3)
	{ ……
	for(exp11; exp12 ; exp13)
	{ ……
	}
	……..
	}
		#include <stdio.h>
		void main()
		{
		int n , line_number , number;
		printf("How many lines: ");
		scanf("%d",&n);
		for(line_number =1 ;line_number <=n; line_number++)
		{
		for(number = 1; number <= line_number; number++)
		printf ("%d\t", number);
		printf ("\n");
		}
		}

2.	Nested while loop / do while loop
	
	while(condition1)
	{ ……
	while(condition2)
	{ ……
	}
	…….
	}
	
	do
	{ ……
	while(condition1)
	{ ……..
	}
	…….
	} while (condition2);
		#include <stdio.h>
		void main( )
		{
		int n , sum;
		printf(“Give any number ”);
		scanf(“%d”,&n);
		do
		{
		sum =0;
		printf(“%d --->”,n);
		while (n>0)
		{
		sum +=n%10;
		n= n/10;
		}
		n=sum;
		} while(n >9);
		printf (“ %d” , n);
		}

Self-Activity

1. The Sample program 1 displays n lines of the following triangle. Type the program and execute it for different values of n.

```
1
1       2
1       2       3
1       2       3       4
```

CODE:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
int main() {
    int n, line, num;
    printf("How many lines: ");
    scanf("%d", &n);
    for(line = 1; line <= n; line++) {
        for(num = 1; num <= line; num++) {
            printf("%d\t", num);
        }
        printf("\n");
    }
    return 1;
}
```

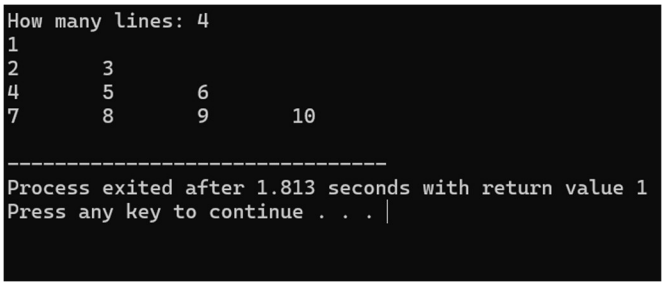
```
How many lines: 4
1
1       2
1       2       3
1       2       3       4

--------------------------------
Process exited after 0.899 seconds with return value 1
Press any key to continue . . .
```

2. Modify the sample program 1 to display n lines of the Floyd's triangle as follows (here n=4).

```
1
2       3
4       5       6
7       8       9       10
```

```c
//CH.SC.U4CSE24015
#include <stdio.h>
int main() {
    int n, line, count = 1, num;
    printf("How many lines: ");
    scanf("%d", &n);
    for(line = 1; line <= n; line++) {
        for(num = 1; num <= line; num++) {
            printf("%d\t", count);
            count++;
        }
        printf("\n");
    }
    return 1;
}
```



3. The sample program 2 computes the sum of digits of a number and the process is repeated till the number reduces to a single digit number. Type the program and execute it for different values of n and give the output

Input number	Output
6534	
67	
8	
___	

Set A . Write C programs for the following problems.

1. Write a program to display all prime numbers between _____and ____.

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
int main() {
    int first,last,prime,i,j;
    printf("Enter the starting number: ");
    scanf("%d", &first);
    printf("Enter the ending number: ");
    scanf("%d", &last);
    printf("Prime numbers between %d and %d are:\n",first,last);
    for (i = first; i <= last; i++) {
        if (i < 2)
        continue;
        prime = 1;
        for (j=2;j*j<=i;j++) {
            if (i % j == 0) {
                prime = 0;
                break;
            }
        }
        if (prime) {
            printf("%d ",i);
        }
    }
    printf("\n");
    return 1;
}
```

Output:

```
Enter the starting number: 2 20
Enter the ending number: Prime numbers between 2 and 20 are:
2 3 5 7 11 13 17 19

Process returned 1 (0x1)   execution time : 5.810 s
Press any key to continue.
```

2. Write a program to display multiplication tables from __to __having n multiples each. The output should be displayed in a tabular format. For example, the multiplication tables of 2 to 9 having 10 multiples each is shown below.

2 × 1 = 2	3 × 1 = 3	9 × 1 = 9
2 × 2 = 4	3 × 2 = 6.....	9 × 2 = 18
.....	.....	
2 × 10 = 20	3 × 10 = 30.....	9 × 10 = 90

Program:

```c
//CH.SC.U4CSE2015
#include <stdio.h>
int main() {
    int i,j,first,last,n;
    printf("Enter the first number : ");
    scanf("%d",&first);
    printf("Enter the last number : ");
    scanf("%d",&last);
    printf("Enter the no of multiples : ");
    scanf("%d",&n);
    printf("\n");
    for(i = 1; i <= n; i++){
        for (j = first; j <= last; j++) {
            printf("%d * %d = %-4d\t", j, i, j * i);
        }
        printf("\n");
    }
    return 1;
}
```

Output:

```
Enter the first number : 2
Enter the last number : 9
Enter the no of multiples : 10

2 * 1 = 2       3 * 1 = 3       4 * 1 = 4       5 * 1 = 5       6 * 1 = 6       7 * 1 = 7       8 * 1 = 8       9 * 1 = 9
2 * 2 = 4       3 * 2 = 6       4 * 2 = 8       5 * 2 = 10      6 * 2 = 12      7 * 2 = 14      8 * 2 = 16      9 * 2 = 18
2 * 3 = 6       3 * 3 = 9       4 * 3 = 12      5 * 3 = 15      6 * 3 = 18      7 * 3 = 21      8 * 3 = 24      9 * 3 = 27
2 * 4 = 8       3 * 4 = 12      4 * 4 = 16      5 * 4 = 20      6 * 4 = 24      7 * 4 = 28      8 * 4 = 32      9 * 4 = 36
2 * 5 = 10      3 * 5 = 15      4 * 5 = 20      5 * 5 = 25      6 * 5 = 30      7 * 5 = 35      8 * 5 = 40      9 * 5 = 45
2 * 6 = 12      3 * 6 = 18      4 * 6 = 24      5 * 6 = 30      6 * 6 = 36      7 * 6 = 42      8 * 6 = 48      9 * 6 = 54
2 * 7 = 14      3 * 7 = 21      4 * 7 = 28      5 * 7 = 35      6 * 7 = 42      7 * 7 = 49      8 * 7 = 56      9 * 7 = 63
2 * 8 = 16      3 * 8 = 24      4 * 8 = 32      5 * 8 = 40      6 * 8 = 48      7 * 8 = 56      8 * 8 = 64      9 * 8 = 72
2 * 9 = 18      3 * 9 = 27      4 * 9 = 36      5 * 9 = 45      6 * 9 = 54      7 * 9 = 63      8 * 9 = 72      9 * 9 = 81
2 * 10 = 20     3 * 10 = 30     4 * 10 = 40     5 * 10 = 50     6 * 10 = 60     7 * 10 = 70     8 * 10 = 80     9 * 10 = 90

Process returned 1 (0x1)   execution time : 3.053 s
Press any key to continue.
```

3. Modify the sample program 1 to display n lines as follows (here n=4).

A

B C

D E F

G H I J

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
void main() {
    int n,line_number,number;
    char ch = 'A';
    printf("How many lines: ");
    scanf("%d",&n);
    for (line_number = 1; line_number <= n; line_number++) {
        for (number = 1; number <= line_number; number++) {
            printf("%c\t", ch);
            ch++;
        }
        printf("\n");
    }
}
```

Output:

```
How many lines: 4
A
B       C
D       E       F
G       H       I       J

Process returned 4 (0x4)   execution time : 4.112 s
Press any key to continue.
```

Set B. Write C programs for the following problems.

1. Write a program to display all Armstrong numbers between 1 and 500. (An Armstrong number is a number such that the sum of cube of digits = number itself. Ex. 153 = 1*1*1 + 5*5*5 + 3*3*3

Program:

```c
//CH.SC.U4CSE2015
#include <stdio.h>
int main() {
    int num,temp,digit,sum;
    printf("Armstrong numbers are :\n");
    for (num = 1; num <= 500; num++) {
        sum = 0;
        temp = num;
        while (temp != 0) {
            digit = temp % 10;
            sum += digit * digit * digit;
            temp /= 10;
        }
        if (sum == num) {
            printf("%d\n", num);
        }
    }
    return 1;
}
```

Output:

```
Armstrong numbers are :
1
153
370
371
407

Process returned 1 (0x1)   execution time : 0.272 s
Press any key to continue.
```

2. Accept n numbers and display the number having the maximum sum of digits.

Program:

```c
//CH.SC.U4CSE2015
#include <stdio.h>
int main(){
    int i,n,num,temp,digit,sum;
    int max_sum = -1;
    int max_num = 0;
    printf("How many numbers : ");
    scanf("%d",&n);
    for (i = 1; i <= n; i++) {
        scanf("%d", &num);
        sum = 0;
        temp = num;
        while (temp != 0) {
            digit = temp % 10;
            sum += digit;
            temp /= 10;
        }
        if (sum > max_sum) {
            max_sum = sum;
            max_num = num;
        }
    }
    printf("The number with maximum sum of digits is : %d",max_num);
    return 1;
}
```

Output:

```
How many numbers : 3
153
225
47
The number with maximum sum of digits is : 47
Process returned 1 (0x1)   execution time : 10.156 s
Press any key to continue.
```

3. Display all perfect numbers below 500. [A perfect number is a number, such that the sum of its factors is equal to the number itself]. Example: 6 (1 + 2 + 3), 28 (1+2+4+7+14)

Program:

```c
//CH.SC.U4CSE2015
#include <stdio.h>
int main(){
    int i,num,sum;
    printf("Perfect numbers below 500 are : \n");
    for (num = 1; num < 500; num++){
        sum = 0;
        for (i = 1; i <= num / 2; i++){
            if (num % i == 0) {
                sum += i;
            }
        }
        if (sum == num) {
            printf("%d\n", num);
        }
    }
    return 1;
}
```

Output:

```
Perfect numbers below 500 are :
6
28
496

Process returned 1 (0x1)   execution time : 0.295 s
Press any key to continue.
```

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Activity Sheet 4 **Date:**

Functions (User Defined functions, Library functions and Recursion).

Exercise 1:

To demonstrate menu driven programs and use of standard library functions

Reading

You should read following topics before starting this exercise

1. Use of switch statement to create menus as in exercise3
2. Use of while and do while loops as in

ctype.h : contains function prototypes for performing various operations on characters.

Function	Purpose	Example
isalpha()	Check whether a character is a alphabet	if (isalpha(ch))
isalnum()	Check whether a character is alphanumeric	if (isalnum(ch))
isdigit()	Check whether a character is a digit	if (isdigit(ch))
isspace()	Check whether a character is a space	if (isspace(ch))
ispunct()	Check whether a character is a punctuation symbol	if (ispunct(ch))
isupper()	Check whether a character is uppercase alphabet	if (isupper(ch))
islower()	Check whether a character is lowercase alphabet	if (isupper(ch))
toupper()	Converts a character to uppercase	ch = toupper(ch)
tolower()	Converts a character to lowercase	ch = tolower(ch)

math.h : contains function prototypes for performing various mathematical operations on numeric data.

Name	Description
ceil	smallest integer not less than parameter
cos	cosine
cosh	hyperbolic cosine
exp(double x)	exponential function, computes e^x
fabs	absolute value
floor	largest integer not greater than parameter
fmod	floating point remainder
log	natural logarithm
log10	base-10 logarithm
pow(x,y)	compute a value taken to an exponent, x^y
sin	sine
sinh	hyperbolic sine
sqrt	square root
tan	tangent
tanh	hyperbolic tangent

A program that does multiple tasks, provides a menu from which user can choose the appropriate task to be performed. The menu should appear again when the task is completed so that the user can choose another task. This process continues till the user decides to quit. A menu driven program can be written using a combination of do-while loop containing a switch statement. One of the options provided in a menu driven program is to exit the program.

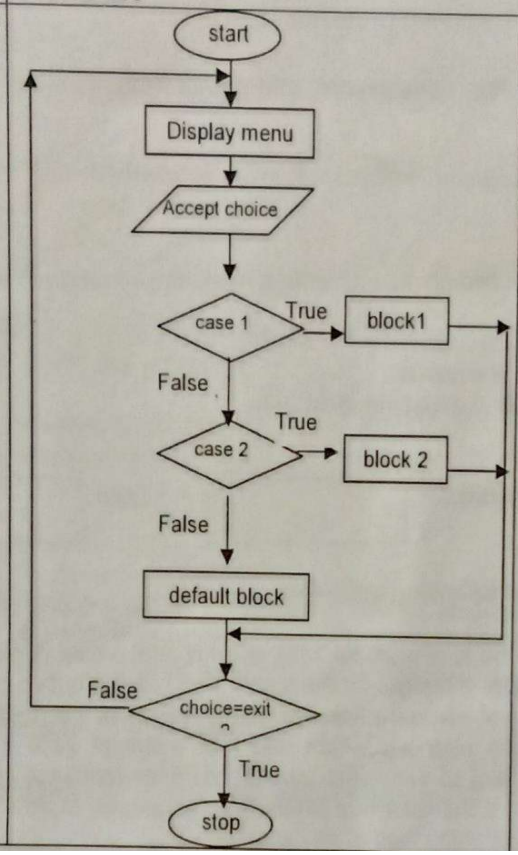
Statement Syntax

```
Do
{
 display menu;
 accept choice;
switch(choice)
{
case value1:
block1;

break;
case value2:
block2;

break;
.
.
.
default : default
block;
  }
}while(choice !=
exit);
```

Flowchart



Example

```
ch = getchar();
do
{
 printf("\n 1: ISUPPER ");
 printf("\n 2: ISLOWER ");
 printf("\n 3: ISDIGIT ");
 printf("\n 4: EXIT");

 printf("Enter your choice :");
 scanf("%d", &choice);

 switch (choice)
 {
  case 1: if(isupper(ch))
            printf("Uppercase");
          break;
  case 2: if(islower(ch))
            printf("Lowercase");
          break;
  case 3: if(isdigit(ch))
            printf("Digit");
          break;
 }
}while (choice!=4);
```

Self-Activity

1. Write a menu driven program to perform the following operations on a character type variable.
    i. Check if it is an alphabet
    ii. Check if it is a digit.
    iii. Check if it is lowercase.
    iv. Check if it is uppercase.
    v. Convert it to uppercase.
    vi. Convert it to lowercase.

   Refer to the sample code given above and use standard functions from ctype.h

Set A . Write C programs for the following problems

1. Write a program, which accepts a character from the user and checks if it is an alphabet, digit or punctuation symbol. If it is an alphabet, check if it is uppercase or lowercase and then change the case.

Program:

Output:

2. Write a menu driven program to perform the following operations till the user selects Exit. Accept appropriate data for each option. Use standard library functions from math.h
i. Power ii. Square Root iii. Floor iv. Ceiling v. Exit

Program:

Output:

Set B. Write C programs for the following problems

1. Accept two fractions (numerator, denominator) and perform the following operations till the user selects Exit.
   i. Addition
   ii.Subtraction
   iii.Multiplication
   iv.EXIT

Program:

Output:

2. Accept x and y coordinates of two points and write a menu driven program to perform the Following operations till the user selects Exit.
   iv. Distance between points
   v. Slope of line between the points.
   vi. Check whether they lie in the same quadrant.
   vii. EXIT

Program:

Output:

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Exercise 2: **Date:**

Objective:

To demonstrate writing C programs in modular way (use of user defined functions)

You should read following topics before starting this exercise

1. Declaring and Defining a function
2. Function call
3. Passing parameters to a function
4. Function returning a value

A function is a named sub-module of a program, which performs a specific, well-defined task. It can accept information from the calling function and return only 1 value. In C, every program has a function named main. main in turn calls other functions.

Sr. No	Actions involving functions	Syntax	Example
1.	Function declaration	returntype function(type arg1, type arg2 …);	void display(); int sum(int x, int y);

2.	Function definition	returntype function(type arg1, type arg2 …)
		{
		/* statements*/
		}
			float calcarea (float r)
			{
			float area = Pi *r*r ;
			return area;
			}

3.	Function call	function(arguments); variable = function(arguments);	display(); ans = calcarea(radius);

1. **Sample code**

The program given below calculates the area of a circle using a function and uses this function to calculate the area of a cylinder using another function.

```
main()
{
float areacircle (float r);
float areacylinder(float r, int h);
float area, r;
printf("\n Enter Radius: ");
scanf("%f",&r);
area=areacircle(r);
printf("\n Area of circle =%6.2f", area);
printf("\n Enter Height: ");
scanf("%d",&h);
area=areacylinder(r,h);
    printf("\n Area of cylinder =%6.2f", area);
}
```

```
float areacircle (float r)
{
      const float pi=3.142;
      return(pi * r*r );
}
float areacylinder (float r, int h)
{
      return 2*areacircle(r)*h;
}
```

2. **Sample code**

The function iswhitespace returns 1 if its character parameter is a space, tab or newline character. The program accepts characters till the user enters EOF and counts the number of white spaces.

```
main()
{
  int iswhitespace (char ch);
  char ch;
  int count=0;
  printf("\n Enter the characters. Type CTRL +Z to terminate: ");
  while((ch=getchar())!=EOF)
   if(iswhitespace(ch))
   count++;
   printf("\n The total number of white spaces =%d", count);
}
int iswhitespace (char ch)
{
     switch(ch)
     {
      case ' ':
      case '\t' :
      case '\n' : return 1;
      default : return 0;
     }
}
```

Self-Activity

1. Type the program given in sample code 1 above and execute the program. Add another function to calculate the volume of sphere and display it.
2. Type the program given in sample code 2 above and execute the program. Modify the function such that it returns 1 if the character is a vowel. Also count the total number of vowels entered.

Set A.Write C programs for the following problems

1. Write a function isEven, which accepts an integer as parameter and returns 1 if the number is even, and 0 otherwise. Use this function in main to accept n numbers and check if they are even or odd.

Program:

Output:

2. Write a function, which accepts a character and integer n as parameter and displays the next n characters.

Program:

Output:

Set B . Write C programs for the following problems

1. Write a function isPrime, which accepts an integer as parameter and returns 1 if the number is prime and 0 otherwise. Use this function in main to display the first 10 prime numbers.

Program:

Output:

2. Write a function that accepts a character as parameter and returns 1 if it is an alphabet, 2 if it is a digit and 3 is it is a special symbol. In main, accept characters till the user enters EOF and use the function to count the total number of alphabets, digits and special symbols entered.

Program:

Output:

3. Write a function power, which calculates x^y. Write another function, which calculates n! Using For loop. Use these functions to calculate the sum of first n terms of the Taylor series:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \text{+ ……}$$

Program:

Output:

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Exercise 3: **Date:**

Objective:

To demonstrate Recursion.

You should read the following topics before starting this exercise

1. Recursive definition
2. Declaring and defining a function
3. How to call a function
4. How to pass parameters to a function

Recursion is a process by which a function calls itself either directly or indirectly. The points to be remembered when recursive functions

i. Each time the function is called recursively it must be closer to the solution.

ii. There must be some terminating condition, which will stop recursion.

iii. Usually the function contains an if –else branching statement where one branch makes recursive call while other branch has non-recursive terminating condition

Expressions having recursive definitions can be easily converted into recursive functions

Sr. No	Recursive definition	Recursive Function	Sample program

1.	The recursive definition for factorial is given below:
	
	n!= 1 if n = 0 or 1
	= n * (n-1)! if n > 1
		long int factorial (int n)
		{
		If (n==0)||(n==1))
		/* terminating condition */
		return(1);
		else
		return(n* factorial(n-1));
		/* recursive call */
		}
			#include <stdio.h>
			main()
			{
			int num;
			/* function declaration */
			long int factorial(int n);
			printf("\n enter the number:");
			scanf("%d",&num);
			printf("\n The factorial of %d is %ld",num,factorial(num));
			}
			/* function code*/

2.	The recursive definition for nCr (no of combinations of r objects out of n objects) is as follows
	nCn = 1
	nC0 = 1
	nCr = n-1Cr + nCr-1
		long int nCr(int n, int r)
		{
		if(n==r || r==0)
		/* terminating condition */
		return(1);
		else
		return(nCr(n-1,r)+nCr(n, r-1));
		/* recursive call */
		}
			#include <stdio.h>
			/* function code*/
			main()
			{
			int n,r; ;
			printf("\n enter the total number of objects:");
			scanf("%d",&n);
			printf("\n enter the number of objects to be selected");
			scanf("%d",&r);
			printf("\n The value %dC%d is %ld",n, r, nCr(n,r));
			}

Self Activity

1. Write the sample program 1 given above and execute the program. Modify the program to Define a global integer variable count and increment it in factorial function. Add a printf statement in main function for variable count. Execute the program for different values and fill in the following table.

Sr. No.	num	factorial	count
1.	0		
2	1		
3	5		
4	__		
5	___		

2. Write the sample program 2 given above and execute the program for different values of n and r. Modify the program to define a global integer variable count and increment it in nCr function.

   Add a print statement in main function for variable count. Execute the program for different values and fill in the following table

Sr. No.	n	r	nCr	count
1.	5	0		
2	5	5		
3	5	2		
4	5	__		
5	___	__		

Set A . Write C programs for the following problems

1. Write a recursive C function to calculate the sum of digits of a number. Use this function in main to accept a number and print sum of its digits.

Program:

Output:

2. Write a recursive C function to calculate the GCD of two numbers. Use this function in main.
   The GCD is calculated as : gcd(a,b) = a if b = 0 = gcd (b, a mod b) otherwise

Program:

Output:

3. Write a recursive C function to calculate x^y. (Do not use standard library function)

Program:

Output:

Set B . Write C programs for the following problems

1. Write a recursive function to calculate the nth Fibonacci number. Use this function in main to Display the first n Fibonacci numbers. The recursive definition of nth Fibonacci number is as follows:

   fib(n) = 1 if n = 1 or 2

   = fib(n-2) + fib(n-1) if n>2

Program:

Output:

2. Write a recursive function to calculate the sum of digits of a number till you get a single digit number.

   Example: 961 -> 16 -> 5. (Note: Do not use a loop)

Program:

Output:

3. Write a recursive C function to print the digits of a number in reverse order. Use this function in main to accept a number and print the digits in reverse order separated by tab.

Example: 3456 6 4 5 3

(Hint: Recursiveprint(n) = print n if n is single digit number
= print n % 10 + tab + Recursiveprint(n/10)

Program:

Output:

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Activity Sheet 5 **Date:**

Arrays (1-D and 2-D)

Exercise 1

Objective

To demonstrate use of 1-D arrays and functions.

Reading

You should read the following topics before starting this exercise

1. What are arrays and how to declare an array?
2. How to enter data in to array and access the elements of an array.
3. How to initialize an array and how to check the bounds of an array?
4. How to pass an array to a function

An array is a collection of data items of the same data type referred to by a common name. Each element of the array is accessed by an index or subscript. Hence, it is also called a subscripted variable.

Actions involving arrays	syntax	Example
Declaration of array	data-type array_name[size];	int temperature[10]; float pressure[20];
Initialization of array	data-type array_name[]={element1, element2, ……, element n}; data-type array_name[size]={element-1, element-2, ……, element-size};	int marks[]={45,57,87,20,90}; marks[3] refers to the fourth element which equals 20 int count[3]={4,2,9}; count[2] is the last element 9 while 4 is count[0]
Accessing elements of an array	The array index begins from 0 (zero) To access an array element, we need to refer to it as array_name[index].	Value = marks[3]; This refers to the 4th element in the array
Entering data into an array.		for(i=0; i<=9; i++) scanf("%d", &marks[i]);
Printing the data from an array		for(i=0; i<=9; i++) printf("%d", marks[i]);

Arrays and function	We can pass an array to a function using two methods.
	Pass the array element by element
	Pass the entire array to the function
		/* Passing the whole array*/ void modify(int a[5])
		{
		int i;
		for(i=0; i<5 ; i++)
		a[i] = i;
		}

Sample program to find the largest element of an array

```
#include<stdio.h> int
main()
{
  int arr[20]; int n;
  void accept(int a[20], int n);
  void display(int a[20], int n);
  int maximum(int a[20], int n);

  printf("How many numbers :");
  scanf("%d", &n);
  accept(arr,n);
  display(arr,n);
  printf("The maximum is :%d" , maximum(arr,n));
}
void accept(int a[20], int n)
{
  int i;
  for(i=0; i<n ; i++)
  scanf("%d", &a[i]);
}
void display(int a[20], int n)
{
  int i;
  for(i=0; i<n ; i++)
  printf("%d\t", a[i]);
}
int maximum(int a[20], int n)
{
  int i, max = a[0];
  for(i=1; i<n ; i++)
  if(a[i] > max)
  max = a[i];
  return max;
}
```

Self Activity

1. Write a program to accept n numbers in an array and display the largest and smallest number. Using these values, calculate the range of elements in the array. Refer to the sample code given above and make appropriate modifications.

```c
//CH.SC.U4CSE24015
#include <stdio.h>
void accept(int a[20], int n);
void display(int a[20], int n);
int maximum(int a[20], int n);
int minimum(int a[20], int n);
int main() {
    int arr[20];
    int n, max, min, range;
    printf("How many numbers: ");
    scanf("%d", &n);
    accept(arr, n);
    display(arr, n);
    max = maximum(arr, n);
    min = minimum(arr, n);
    range = max - min;
    printf("\nThe maximum is: %d", max);
    printf("\nThe minimum is: %d", min);
    printf("\nThe range is: %d\n", range);
    return 1;
}

void accept(int a[20], int n) {
    int i;
    printf("Enter %d numbers:\n", n);
    for(i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }
}
void display(int a[20], int n) {
    int i;
    printf("Array elements: ");
    for(i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");
}
int maximum(int a[20], int n) {
    int i, max = a[0];
    for(i = 1; i < n; i++) {
        if(a[i] > max)
            max = a[i];
    }
    return max;
}
int minimum(int a[20], int n) {
    int i, min = a[0];
    for(i = 1; i < n; i++) {
        if(a[i] < min)
            min = a[i];
    }
    return min;
}
```

```
How many numbers: 5
Enter 5 numbers:
1 2 3 4 5
Array elements: 1 2 3 4 5

The maximum is: 5
The minimum is: 1
The range is: 4

Process returned 1 (0x1)   execution time : 4.806 s
Press any key to continue.
```

2. Write a program to accept n numbers in an array and calculate the average. Refer to the sample code given above and make appropriate modifications.

CODE:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
void accept(int a[20], int n);
void display(int a[20], int n);
float average(int a[20], int n);
int main() {
    int arr[20], n;
    float avg;
    printf("How many numbers: ");
    scanf("%d", &n);
    accept(arr, n);
    display(arr, n);
    avg = average(arr, n);
    printf("\nThe average is: %.2f\n", avg);
    return 1;
}
void accept(int a[20], int n) {
    int i;
    printf("Enter %d numbers:\n", n);
    for(i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }
}
void display(int a[20], int n) {
    int i;
    printf("Array elements: ");
    for(i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");
}
float average(int a[20], int n) {
    int i, sum = 0;
    for(i = 0; i < n; i++) {
        sum += a[i];
    }
    return (float)sum / n;
}
```

```
How many numbers: 5
Enter 5 numbers:
1 3 5 7 9
Array elements: 1 3 5 7 9

The average is: 5.00

Process returned 1 (0x1)   execution time : 4.370 s
Press any key to continue.
```

Set A.

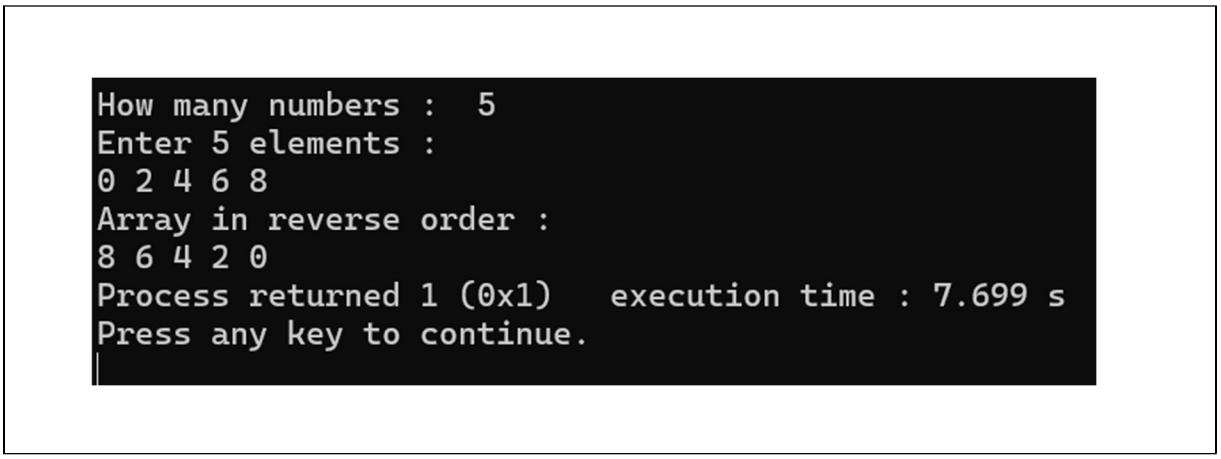
Write programs to solve the following problems

1. Write a program to accept n numbers and display the array in the reverse order. Write separate functions to accept and display.

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
void accept(int a[], int n) {
    int i;
    printf("Enter %d elements : \n", n);
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);
}
void displayReverse(int a[], int n) {
    int i;
    printf("Array in reverse order : \n");
    for(i=n-1; i>=0; i--)
        printf("%d ", a[i]);
}
int main() {
    int arr[50], n;
    printf("How many numbers :  ");
    scanf("%d", &n);
    accept(arr, n);
    displayReverse(arr, n);
    return 1;
}
```

Output:

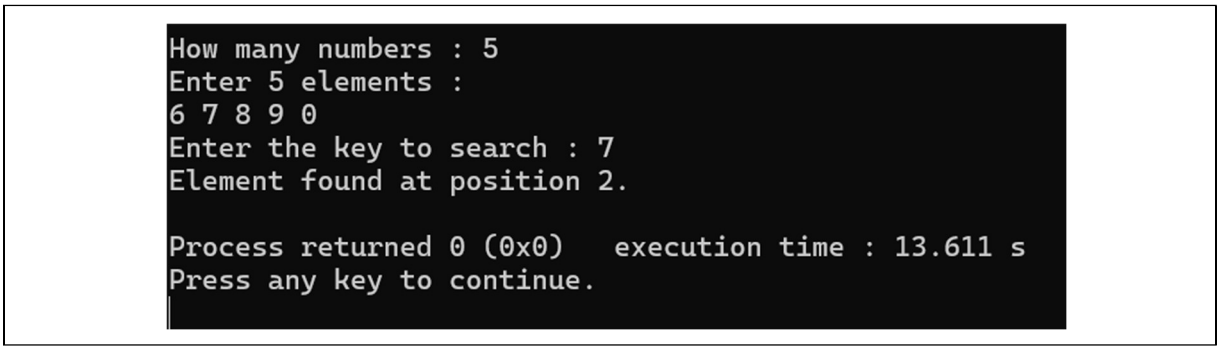

How many numbers : 5
Enter 5 elements :
0 2 4 6 8
Array in reverse order :
8 6 4 2 0
Process returned 1 (0x1)   execution time : 7.699 s
Press any key to continue.

2. Write a function for Linear Search, which accepts an array of n elements and a key as parameters and returns the position of key in the array and -1 if the key is not found. Accept n numbers from the user, store them in an array. Accept the key to be searched and search it using this function. Display appropriate messages.

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
int linearSearch(int a[], int n, int key) {
    int i;
    for(i=0; i<n; i++) {
        if(a[i]==key)
            return i;
    }
    return -1;
}
void accept(int a[], int n) {
    int i;
    printf("Enter %d elements : \n", n);
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);
}
int main() {
    int arr[50], n, key, pos;
    printf("How many numbers : ");
    scanf("%d", &n);
    accept(arr, n);
    printf("Enter the key to search : ");
    scanf("%d", &key);
    pos = linearSearch(arr, n, key);
    if(pos==-1)
        printf("Element not found.\n");
    else
        printf("Element found at position %d.\n", pos+1);
    return 0;
}
```

Output:


How many numbers : 5
Enter 5 elements :
6 7 8 9 0
Enter the key to search : 7
Element found at position 2.
Process returned 0 (0x0) execution time : 13.611 s
Press any key to continue.

3. Write a function, which accepts an integer array and an integer as parameters and counts the occurrences of the number in the array.

Example: Input 1 5 2 1 6 3 8 2 9 15 1 30

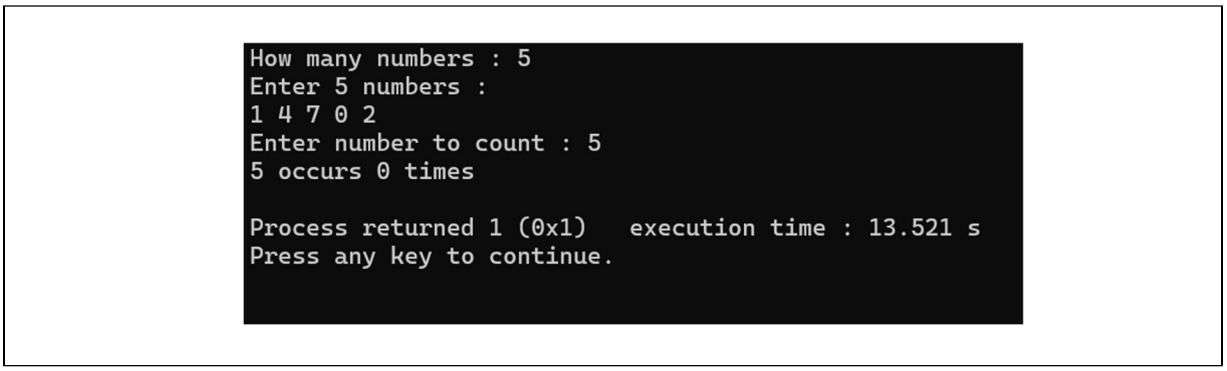
Number : 1

Output: 1 occurs 3 times

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
int countOccurrences(int a[], int n, int num) {
    int i, count=0;
    for(i=0; i<n; i++) {
        if(a[i]==num)
            count++;
    }
    return count;
}
int main() {
    int arr[50], n, num, i, count;
    printf("How many numbers : ");
    scanf("%d", &n);
    printf("Enter %d numbers : \n", n);
    for(i=0; i<n; i++)
        scanf("%d", &arr[i]);
    printf("Enter number to count : ");
    scanf("%d", &num);
    count = countOccurrences(arr, n, num);
    printf("%d occurs %d times\n", num, count);
    return 1;
}
```

Output:

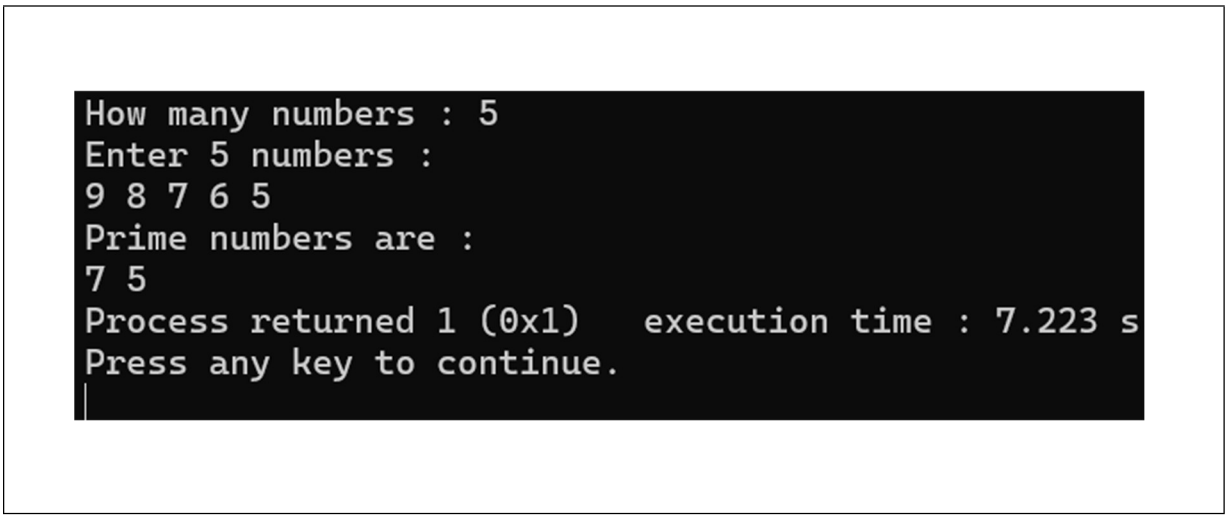

How many numbers : 5
Enter 5 numbers :
1 4 7 0 2
Enter number to count : 5
5 occurs 0 times
Process returned 1 (0x1) execution time : 13.521 s
Press any key to continue.

4. Write a program to accept n numbers and store all prime numbers in an array called prime. Display this array.

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
int isPrime(int num) {
    int i;
    if(num < 2)
        return 0;
    for(i=2; i <= num/2; i++) {
        if(num % i == 0)
            return 0;
    }
    return 1;
}
int main() {
    int arr[50], prime[50], n, i, p=0;
    printf("How many numbers : ");
    scanf("%d", &n);
    printf("Enter %d numbers : \n", n);
    for(i=0; i<n; i++)
        scanf("%d", &arr[i]);
    for(i=0; i<n; i++) {
        if(isPrime(arr[i]))
            prime[p++] = arr[i];
    }
    printf("Prime numbers are : \n");
    for(i=0; i<p; i++)
        printf("%d ", prime[i]);
    return 1;
}
```

Output:


How many numbers : 5
Enter 5 numbers :
9 8 7 6 5
Prime numbers are :
7 5
Process returned 1 (0x1)   execution time : 7.223 s
Press any key to continue.

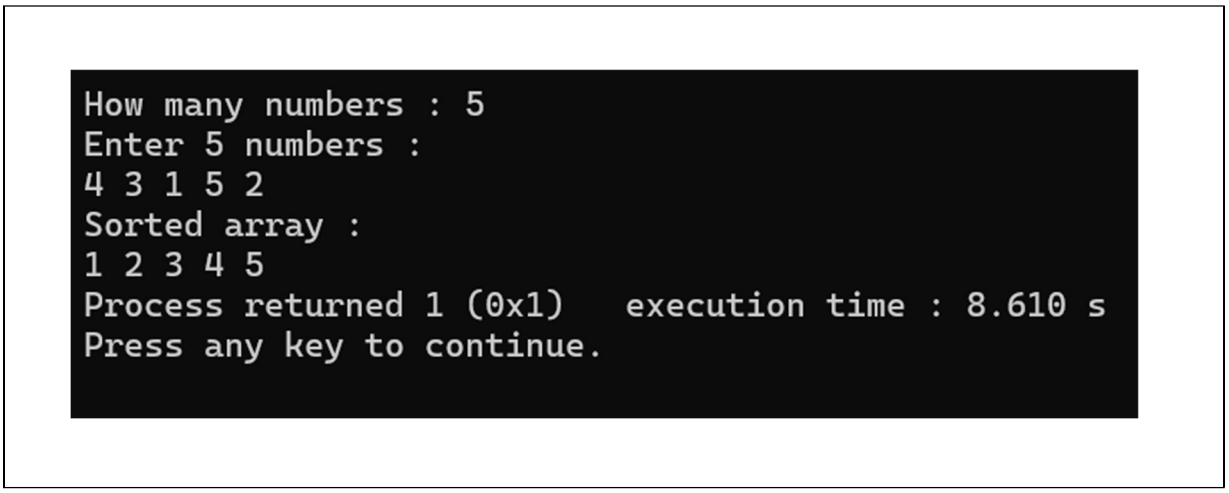
Set B. Write programs to solve the following problems

1. Write a function to sort an array of n integers using Bubble sort method. Accept n numbers from the user, store them in an array and sort them using this function. Display the sorted array.

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
void bubbleSort(int a[], int n) {
    int i, j, temp;
    for(i=0; i<n-1; i++) {
        for(j=0; j<n-i-1; j++) {
            if(a[j] > a[j+1]) {
                temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
            }
        }
    }
}
int main() {
    int arr[50], n, i;
    printf("How many numbers : ");
    scanf("%d", &n);
    printf("Enter %d numbers : \n", n);
    for(i=0; i<n; i++)
        scanf("%d", &arr[i]);
    bubbleSort(arr, n);
    printf("Sorted array : \n");
    for(i=0; i<n; i++)
        printf("%d ", arr[i]);
    return 1;
}
```

Output:

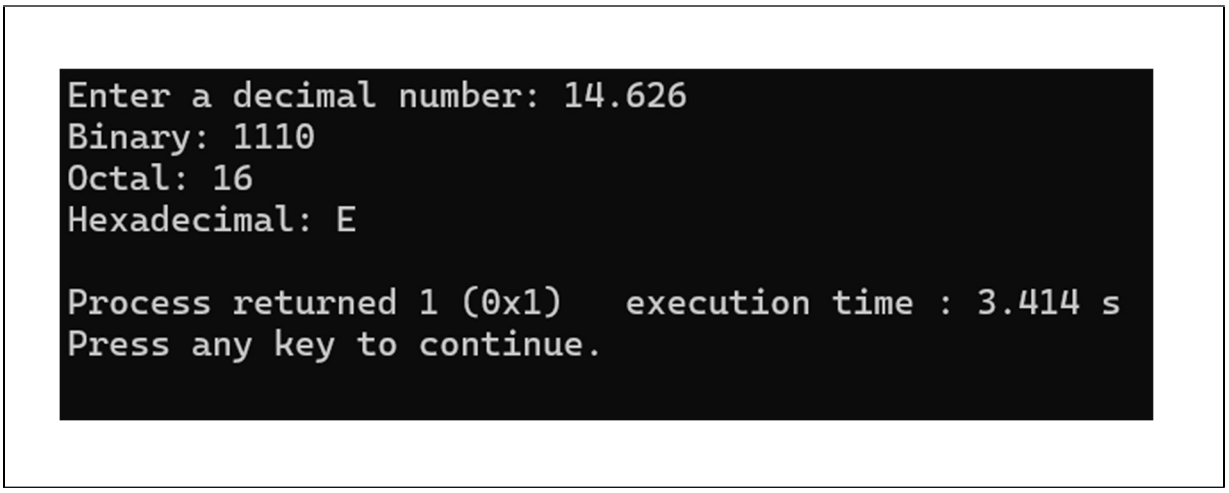

How many numbers : 5
Enter 5 numbers :
4 3 1 5 2
Sorted array :
1 2 3 4 5
Process returned 1 (0x1)   execution time : 8.610 s
Press any key to continue.

2. Write a program to accept a decimal number and convert it to binary, octal and hexadecimal. Write separate functions.

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
void toBinary(int n) {
    int bin[32], i=0;
    if(n==0) {
        printf("0");
        return;
    }
    while(n>0) {
        bin[i++] = n%2;
        n = n/2;
    }
    for(i=i-1; i>=0; i--)
        printf("%d", bin[i]);
}
int main() {
    int num;
    printf("Enter a decimal number: ");
    scanf("%d", &num);
    printf("Binary: ");
    toBinary(num);
    printf("\nOctal: %o\n", num);
    printf("Hexadecimal: %X\n", num);
    return 1;
}
```

Output:


Enter a decimal number: 14.626
Binary: 1110
Octal: 16
Hexadecimal: E
Process returned 1 (0x1) execution time : 3.414 s
Press any key to continue.

3. Write a program to find the intersection of the two sets of integers. Store the intersection in another array.

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
int main() {
    int a1[50], a2[50], a3[50];
    int n1, n2, i, j, k=0;
    printf("Enter number of elements in first set : ");
    scanf("%d", &n1);
    printf("Enter elements of first set :\n");
    for(i=0; i<n1; i++)
        scanf("%d", &a1[i]);
    printf("Enter number of elements in second set : ");
    scanf("%d", &n2);
    printf("Enter elements of second set :\n");
    for(i=0; i<n2; i++)
        scanf("%d", &a2[i]);
    for(i=0; i<n1; i++) {
        for(j=0; j<n2; j++) {
            if(a1[i]==a2[j]) {
                a3[k++] = a1[i];
                break;
            }
        }
    }
    printf("Intersection :\n");
    for(i=0; i<k; i++)
        printf("%d ", a3[i]);
    return 1;
}
```

Output:

```
Enter number of elements in first set : 5
Enter elements of first set :
1 8 5 9 5
Enter number of elements in second set : 5
Enter elements of second set :
7 5 0 7 3
Intersection :
5 5
Process returned 1 (0x1)   execution time : 14.055 s
Press any key to continue.
```

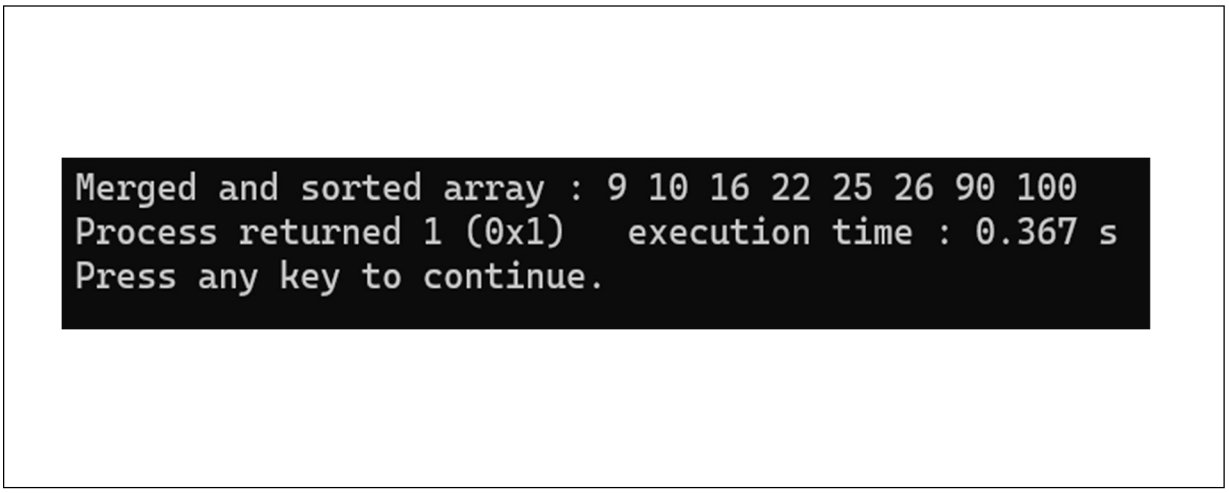
4. Write a program to merge two sorted arrays into a third array such that the third array is also in the sorted order.

a1	10	25	90					
a2	9	16	22	26	100			
a3	9	10	16	22	25	26	90	100

Program:

```c
// CH.SC.U4CSE24015
#include <stdio.h>
int main() {
    int a1[100] = {10, 25, 90};
    int a2[100] = {9, 16, 22, 26, 100};
    int a3[200];
    int n1 = 3, n2 = 5;
    int i = 0, j = 0, k = 0;
    while (i < n1 && j < n2) {
        if (a1[i] < a2[j]) {
            a3[k++] = a1[i++];
        } else {
            a3[k++] = a2[j++];
        }
    }
    while (i < n1) {
        a3[k++] = a1[i++];
    }
    while (j < n2) {
        a3[k++] = a2[j++];
    }
    printf("Merged and sorted array : ");
    for (i = 0; i < k; i++) {
        printf("%d ", a3[i]);
    }
    return 1;
}
```

Output:



Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Exercise 2

Objective

To demonstrate use of 2-D arrays and functions.

You should read the following topics before starting this exercise

1. How to declare and initialize two-dimensional array
2. Accessing elements
3. Usage of two-dimensional arrays

Actions involving 2-D arrays	syntax	Example
Declaration of 2-D array	data-type array_name[size][size];	int mat[10][10]; float sales[4][10];
Initialization of 2-D array	data-type array_name[rows][cols]= { {elements of row 0},{ elements of row 1},…..}; data-type array_name[][cols]={element1, element2, ……, element size};	int num[][2] = {12, 34, 23, 45, 56,45}; int num[3][2] = { {1,2}, {3,4}, {5,6}}; int num[3][2] = { 1,2,3,4, 5,6};
Accessing elements of 2-D array	Accessing elements of 2-dimensional array - in general, the array element is referred as: array_name[index1][index2] where index1 is the row location of and index2 is the column location of an element in the array.	int m[3][2]; m is declared as a two dimensional array (matrix) having 3 rows (numbered 0 to 2) and 2 columns (numbered 0 to 1). The first element is m[0] [0] and the last is m [2][1]. value = m[1][1];

Entering data into a 2-D array.		int mat[4][3];
		for (i=0; i<4; i++)
		/* outer loop for rows */
		for (j=0;j<3; j++)
		/* inner loop for columns */
		scanf("%d", &mat[i][j]);

Printing the data from a 2-D array		for (i=0; i<4; i++)
		/* outer loop for rows */
		{
		for (j=0;j<3; j++)
		/* inner loop for columns */
		printf("%d\t" , mat[i][j]);
		printf("\n");
		}

Sample program to accept, display and print the sum of elements of each row of a matrix.

```
#include<stdio.h>
int main()
{
  int mat[10][10], m, n;
  void display(int a[10][10], int m, int n);
  void accept(int a[10][10], int m, int n);
  void sumofrows(int a[10][10], int m, int n);
  printf("How many rows and columns? ");
  scanf("%d%d",&m, &n);

  printf("Enter the matrix elements :");
  accept(mat, m, n);
  printf("\n The matrix is :\n");
  display(mat, m, n);
  sumofrows(mat,m,n);
 }
void accept(int a[10][10], int m, int n)
{
      int i,j;
     for (i=0; i<m; i++)  /* outer loop for rows */  for
     (j=0;j<n; j++)  /* inner loop for columns */
     scanf("%d", &a[i][j]);
}
void display(int a[10][10], int m, int n)
{
     int i,j;
     printf("\nThe elements of %d by %d matrix are\n", m, n); for
     (i=0; i<m; i++)  /* outer loop for rows */
     {
       for (j=0;j<n; j++)  /* inner loop for columns */
       printf("%d\t" , a[i][j]);
       printf("\n");
     }
}
void somofrows(int a[10][10], int m, int n)
{
     int i,j, sum;
     for (i=0; i<m; i++)   /* outer loop for rows */
     {
       sum=0'
       for (j=0;j<n; j++)  /* inner loop for columns */
       sum= sum+a[i][j];
       printf("Sum of elements of row %d = %d", i, sum);
     }
}
```

1. Write a program to accept, display and print the sum of elements of each row and sum of elements of each column of a matrix. Refer to sample code given above.

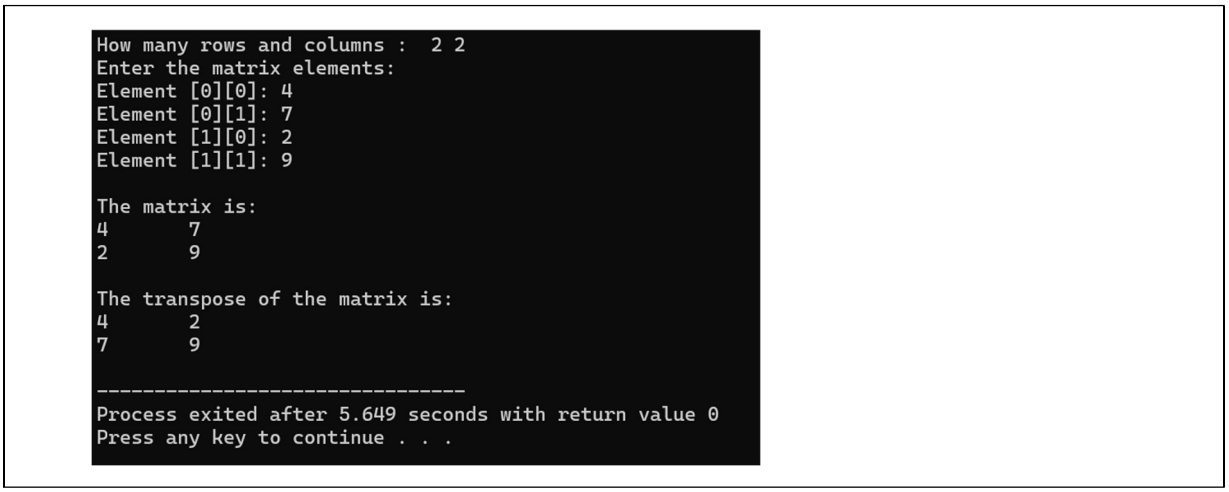
Set A . Write C programs for the following problems.

1. Write a program to accept a matrix A of size m X n and store its transpose in matrix B. Display matrix B. Write separate functions.

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
void accept(int a[10][10], int m, int n);
void display(int a[10][10], int m, int n);
void transpose(int a[10][10], int b[10][10], int m, int n);
void accept(int a[10][10], int m, int n) {
    int i, j;
    for (i = 0; i < m; i++) {
        for (j = 0; j < n; j++) {
            printf("Element [%d][%d]: ", i, j);
            scanf("%d", &a[i][j]);
        }
    }
}
void display(int a[10][10], int m, int n) {
    int i, j;
    for (i = 0; i < m; i++) {
        for (j = 0; j < n; j++)
            printf("%d\t", a[i][j]);
        printf("\n");
    }
}
void transpose(int a[10][10], int b[10][10], int m, int n) {
    int i, j;
    for (i = 0; i < m; i++) {
        for (j = 0; j < n; j++) {
            b[j][i] = a[i][j];
        }
    }
}
int main() {
    int mat[10][10], trans[10][10];
    int m, n;
    printf("How many rows and columns :  ");
    scanf("%d%d", &m, &n);
    printf("Enter the matrix elements:\n");
    accept(mat, m, n);
    printf("\nThe matrix is:\n");
    display(mat, m, n);
    transpose(mat, trans, m, n);
    printf("\nThe transpose of the matrix is:\n");
    display(trans, n, m);
    return 0;
}
```

Output:


How many rows and columns : 2 2
Enter the matrix elements:
Element [0][0]: 4
Element [0][1]: 7
Element [1][0]: 2
Element [1][1]: 9
The matrix is:
4 7
2 9
The transpose of the matrix is:
4 2
7 9
Process exited after 5.649 seconds with return value 0
Press any key to continue . . .

2. Write a program to add and multiply two matrices. Write separate functions to accept, display, add and multiply the matrices. Perform necessary checks before adding and multiplying the matrices.

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
void accept(int a[10][10], int m, int n);
void display(int a[10][10], int m, int n);
int add(int a[10][10], int b[10][10], int res[10][10], int m1, int n1, int m2, int n2);
int multiply(int a[10][10], int b[10][10], int res[10][10], int m1, int n1, int m2, int n2);

int main() {
    int a[10][10], b[10][10], sum[10][10], product[10][10];
    int m1, n1, m2, n2;
    printf("Enter rows and columns for Matrix A: ");
    scanf("%d%d", &m1, &n1);
    printf("Enter the elements for Matrix A:\n");
    accept(a, m1, n1);
    printf("Enter rows and columns for Matrix B: ");
    scanf("%d%d", &m2, &n2);
    printf("Enter the elements for Matrix B:\n");
    accept(b, m2, n2);
    printf("\nMatrix A:\n");
    display(a, m1, n1);
    printf("\nMatrix B:\n");
    display(b, m2, n2);
    if (add(a, b, sum, m1, n1, m2, n2)) {
        printf("\nSum of A and B:\n");
        display(sum, m1, n1);
    } else {
        printf("\nAddition not possible (matrices are of different sizes).\n");
    }
    if (multiply(a, b, product, m1, n1, m2, n2)) {
        printf("\nProduct of A and B:\n");
        display(product, m1, n2);
    } else {
        printf("\nMultiplication not possible (columns of A != rows of B).\n");
    }
    return 0;
}
void accept(int a[10][10], int m, int n) {
    int i, j;
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
            scanf("%d", &a[i][j]);
}
void display(int a[10][10], int m, int n) {
    int i, j;
    for (i = 0; i < m; i++) {
        for (j = 0; j < n; j++)
            printf("%d\t", a[i][j]);
        printf("\n");
    }
}
```

```c
int add(int a[10][10], int b[10][10], int res[10][10], int m1, int n1, int m2, int n2) {
    if (m1 != m2 || n1 != n2)
        return 0;
    int i, j;
    for (i = 0; i < m1; i++)
        for (j = 0; j < n1; j++)
            res[i][j] = a[i][j] + b[i][j];
    return 1;
}
int multiply(int a[10][10], int b[10][10], int res[10][10], int m1, int n1, int m2, int n2) {
    if (n1 != m2)
        return 0;
    int i, j, k;
    for (i = 0; i < m1; i++)
        for (j = 0; j < n2; j++) {
            res[i][j] = 0;
            for (k = 0; k < n1; k++)
                res[i][j] += a[i][k] * b[k][j];
        }
    return 1;
}
```

Output:

```
Enter rows and columns for Matrix A: 2 2
Enter the elements for Matrix A:
1 2 3 4
Enter rows and columns for Matrix B: 2 2
Enter the elements for Matrix B:
5 6 7 8

Matrix A:
1       2
3       4

Matrix B:
5       6
7       8

Sum of A and B:
6       8
10      12

Product of A and B:
19      22
43      50

--------------------------------
Process exited after 10.35 seconds with return value 0
Press any key to continue . . .
```

Set B . Write C programs for the following problems.

1. Write a menu driven program to perform the following operations on a square matrix. Write separate functions for each option.
   - i) Check if the matrix is symmetric.
   - ii) Display the trace of the matrix (sum of diagonal elements).
   - iii) Check if the matrix is an upper triangular matrix.
   - iv) Check if the matrix is a lower triangular matrix.
   - v) Check if it is an identity matrix.

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
void accept(int a[10][10], int n);
void display(int a[10][10], int n);
int isSymmetric(int a[10][10], int n);
int trace(int a[10][10], int n);
int isUpperTriangular(int a[10][10], int n);
int isLowerTriangular(int a[10][10], int n);
int isIdentity(int a[10][10], int n);
void accept(int a[10][10], int n) {
    int i, j;
    for(i = 0; i < n; i++)
        for(j = 0; j < n; j++)
            scanf("%d", &a[i][j]);
}
void display(int a[10][10], int n) {
    int i, j;
    for(i = 0; i < n; i++) {
        for(j = 0; j < n; j++)
            printf("%d\t", a[i][j]);
        printf("\n");
    }
}
int isSymmetric(int a[10][10], int n) {
    int i, j;
    for(i = 0; i < n; i++)
        for(j = 0; j < n; j++)
            if(a[i][j] != a[j][i])
                return 0;
    return 1;
}
```

```c
int trace(int a[10][10], int n) {
    int i, sum = 0;
    for(i = 0; i < n; i++)
        sum += a[i][i];
    return sum;
}
int isUpperTriangular(int a[10][10], int n) {
    int i, j;
    for(i = 1; i < n; i++)
        for(j = 0; j < i; j++)
            if(a[i][j] != 0)
                return 0;
    return 1;
}
int isLowerTriangular(int a[10][10], int n) {
    int i, j;
    for(i = 0; i < n; i++)
        for(j = i+1; j < n; j++)
            if(a[i][j] != 0)
                return 0;
    return 1;
}
int isIdentity(int a[10][10], int n) {
    int i, j;
    for(i = 0; i < n; i++)
        for(j = 0; j < n; j++) {
            if(i == j && a[i][j] != 1)
                return 0;
            if(i != j && a[i][j] != 0)
                return 0;
        }
    return 1;
}
int main() {
    int mat[10][10], n, choice;
    printf("Enter order of the square matrix: ");
    scanf("%d", &n);
    printf("Enter the elements of the matrix:\n");
    accept(mat, n);
    printf("\nMatrix is:\n");
    display(mat, n);
    do {
        printf("\n--- MENU ---\n");
        printf("1. Check if the matrix is symmetric\n");
        printf("2. Display the trace of the matrix\n");
        printf("3. Check if the matrix is upper triangular\n");
        printf("4. Check if the matrix is lower triangular\n");
        printf("5. Check if the matrix is identity matrix\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch(choice) {
```

```
            case 1:
                if(isSymmetric(mat, n))
                    printf("The matrix is symmetric.\n");
                else
                    printf("The matrix is NOT symmetric.\n");
                break;
            case 2:
                printf("The trace of the matrix is: %d\n", trace(mat, n));
                break;
            case 3:
                if(isUpperTriangular(mat, n))
                    printf("The matrix is upper triangular.\n");
                else
                    printf("The matrix is NOT upper triangular.\n");
                break;
            case 4:
                if(isLowerTriangular(mat, n))
                    printf("The matrix is lower triangular.\n");
                else
                    printf("The matrix is NOT lower triangular.\n");
                break;
            case 5:
                if(isIdentity(mat, n))
                    printf("The matrix is an identity matrix.\n");
                else
                    printf("The matrix is NOT an identity matrix.\n");
                break;
            case 6:
                printf("Exiting program.\n");
                break;
            default:
                printf("Invalid choice! Try again.\n");
        }
    } while(choice != 6);

    return 0;
}
```

Output:

```
Enter order of the square matrix: 2 2
Enter the elements of the matrix:
1 2 3 4

Matrix is:
2       1
2       3
```

```
--- MENU ---
1. Check if the matrix is symmetric
2. Display the trace of the matrix
3. Check if the matrix is upper triangular
4. Check if the matrix is lower triangular
5. Check if the matrix is identity matrix
6. Exit
Enter your choice: 1
The matrix is NOT symmetric.

--- MENU ---
1. Check if the matrix is symmetric
2. Display the trace of the matrix
3. Check if the matrix is upper triangular
4. Check if the matrix is lower triangular
5. Check if the matrix is identity matrix
6. Exit
Enter your choice: 2
The trace of the matrix is: 5

--- MENU ---
1. Check if the matrix is symmetric
2. Display the trace of the matrix
3. Check if the matrix is upper triangular
4. Check if the matrix is lower triangular
5. Check if the matrix is identity matrix
6. Exit
Enter your choice: 3
The matrix is NOT upper triangular.
```

```
--- MENU ---
1. Check if the matrix is symmetric
2. Display the trace of the matrix
3. Check if the matrix is upper triangular
4. Check if the matrix is lower triangular
5. Check if the matrix is identity matrix
6. Exit
Enter your choice: 4
The matrix is NOT lower triangular.

--- MENU ---
1. Check if the matrix is symmetric
2. Display the trace of the matrix
3. Check if the matrix is upper triangular
4. Check if the matrix is lower triangular
5. Check if the matrix is identity matrix
6. Exit
Enter your choice: 5
The matrix is NOT an identity matrix.

--- MENU ---
1. Check if the matrix is symmetric
2. Display the trace of the matrix
3. Check if the matrix is upper triangular
4. Check if the matrix is lower triangular
5. Check if the matrix is identity matrix
6. Exit
Enter your choice: 6
Exiting program.

--------------------------------
Process exited after 81.54 seconds with return value 0
Press any key to continue . . .
```

2. Write a program to accept an m X n matrix and display an m+1 X n+1 matrix such that the $m+1^{th}$ row contains the sum of all elements of corresponding row and the $n+1^{th}$ column contains the sum of elements of the corresponding column.

Example:

A			B			
1	2	3	1	2	3	6
4	5	6	4	5	6	15
7	8	9	7	8	9	24
			12	15	18	45

Program:

```c
//CH.SC.U4CSE24015
#include <stdio.h>
void accept(int a[10][10], int m, int n);
void display_augmented(int a[10][10], int m, int n);
void accept(int a[10][10], int m, int n) {
    int i, j;
    for(i=0; i<m; i++)
        for(j=0; j<n; j++)
            scanf("%d", &a[i][j]);
}
void display_augmented(int a[10][10], int m, int n) {
    int i, j;
    int sum_row, sum_col, total = 0;
    for(i=0; i<m; i++) {
        sum_row = 0;
        for(j=0; j<n; j++) {
            printf("%d\t", a[i][j]);
            sum_row += a[i][j];
        }
        printf("%d\n", sum_row);
        total += sum_row;
    }
    for(j=0; j<n; j++) {
        sum_col = 0;
        for(i=0; i<m; i++)
            sum_col += a[i][j];
        printf("%d\t", sum_col);
    }
    printf("%d\n", total);
}
```

```c
int main() {
    int a[10][10], m, n;
    printf("Enter the number of rows (m): ");
    scanf("%d", &m);
    printf("Enter the number of columns (n): ");
    scanf("%d", &n);
    printf("Enter the elements of the matrix:\n");
    accept(a, m, n);
    printf("\nThe augmented matrix is:\n");
    display_augmented(a, m, n);
    return 0;
}
```

Output:

```
Enter the number of rows (m): 2
Enter the number of columns (n): 2
Enter the elements of the matrix:
1 2 3 4

The augmented matrix is:
1       2       3
3       4       7
4       6       10

--------------------------------
Process exited after 5.23 seconds with return value 0
Press any key to continue . . .
```

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]