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:

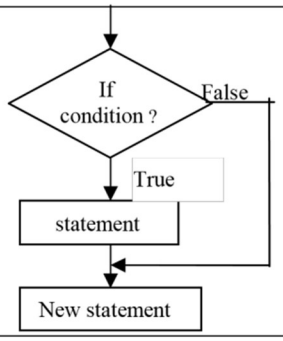
You should read following topics before starting this exercise

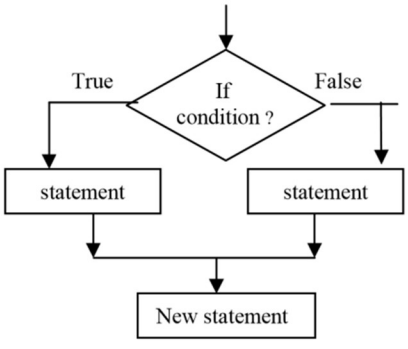
1. Different types of decision-making statements available in C.
2. Syntax for these statements.

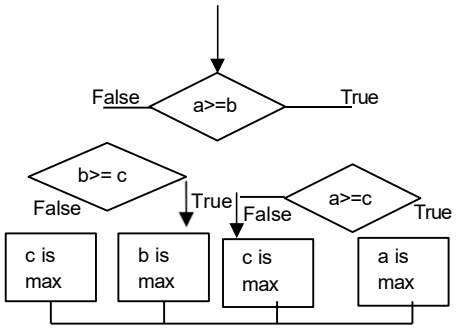
During problem solving, we come across situations when we have to choose one of the alternative paths depending upon the result of some condition. Condition is an expression evaluating to true or false. This is known as the Branching or decision-making statement. Several forms of If and else constructs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

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

Sr. No	Statement Syntax	Flowchart	Example

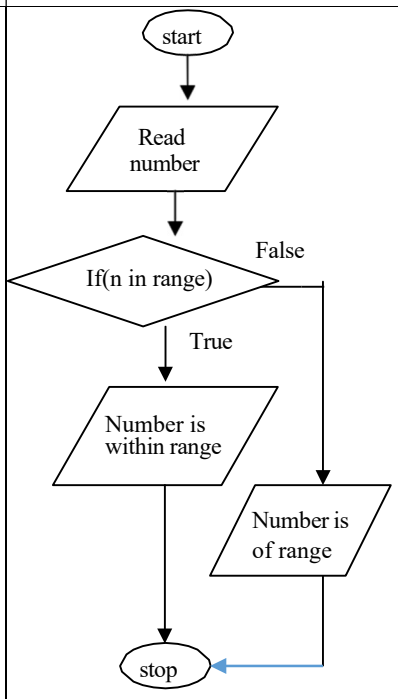
1.	**if statement**
	
	if (condition)
	{
	statement;
	}
			if(n > 0)
			printf("Number is positive");

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.
```

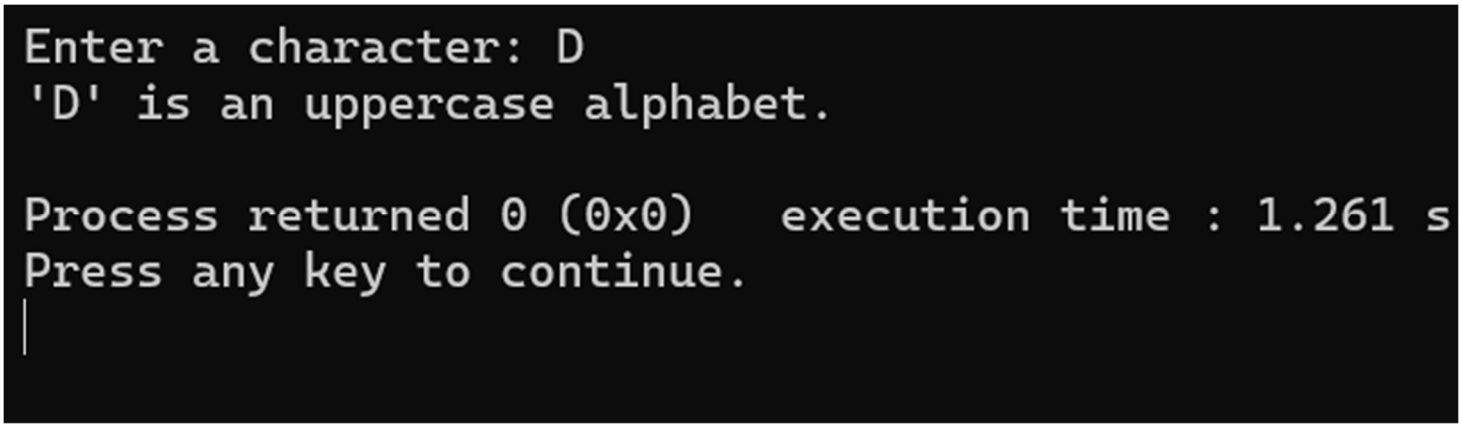
Set B: Apply all the three program development steps for the following examples.

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:

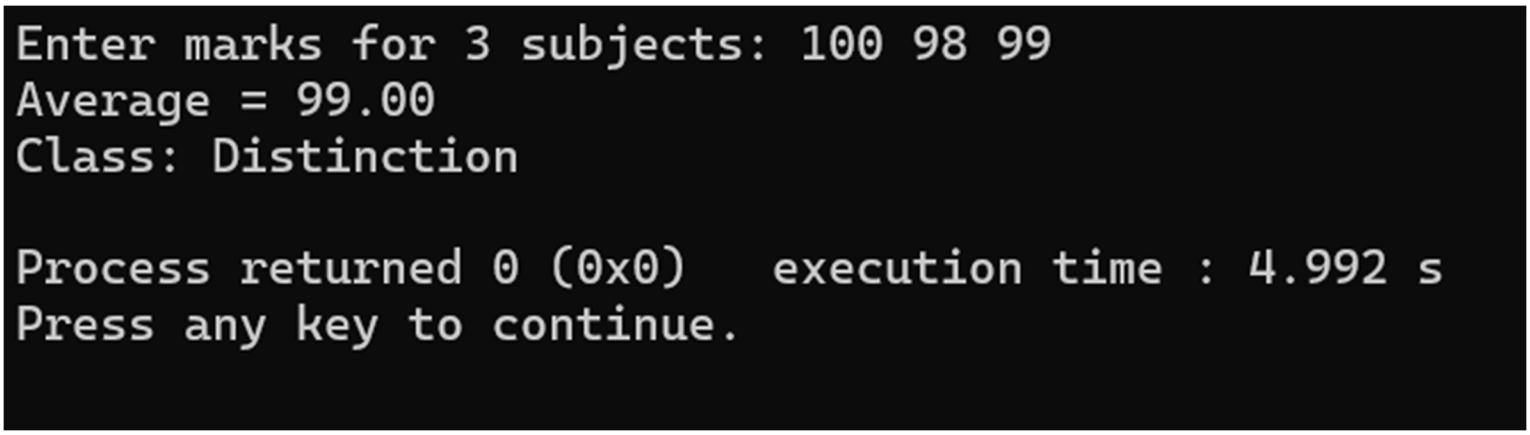
```
Enter cost price: 2000
Enter selling price: 2200
Profit: ?200.00

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;
}
```

dddddOutput:



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:

```
if (marks >= 90) {
                printf("Grade: A");
                    } else if (marks >= 75) {
                          if (marks >= 85)
                    printf("Grade: B+");
                else
                    printf("Grade: B");
} else {
                printf("Grade: C");
}
```

2. How would you handle missed cases in conditional statements.
   Ans:
   Missed cases can be handled by:
   - Using a final else block to catch all unhandled scenarios.
   - Validating inputs before processing.
   - Logging unexpected values for debugging.

3. Can you describe a situation where the use of a ternary operator would be more appropriate than a traditional if else statement.
   Ans:

```
int max = (a > b) ? a : b;
```

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

Ans:
Error handling can be done using if conditions to check for invalid inputs, null pointers, or failed operations, and responding accordingly.
Example:

```
if (denominator == 0) {
   printf("Error: Division by zero is not allowed.\n");
} else {
   result = numerator / denominator;
}
```

Date:

Exercise 2:

Objective:

To demonstrate decision making statements (switch case)

Reading:

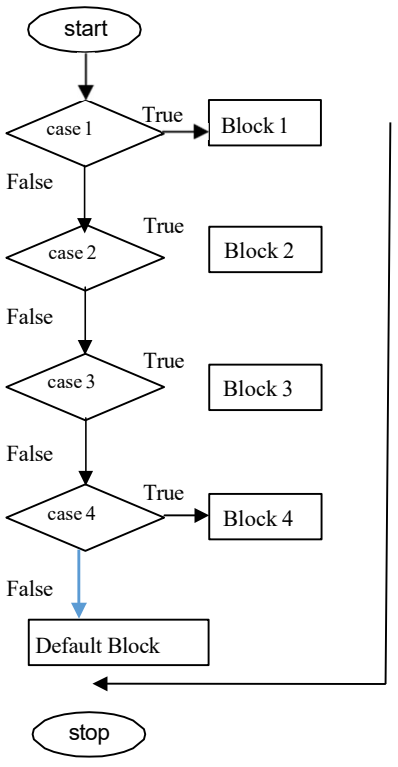
You should read following topics before starting this exercise

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.

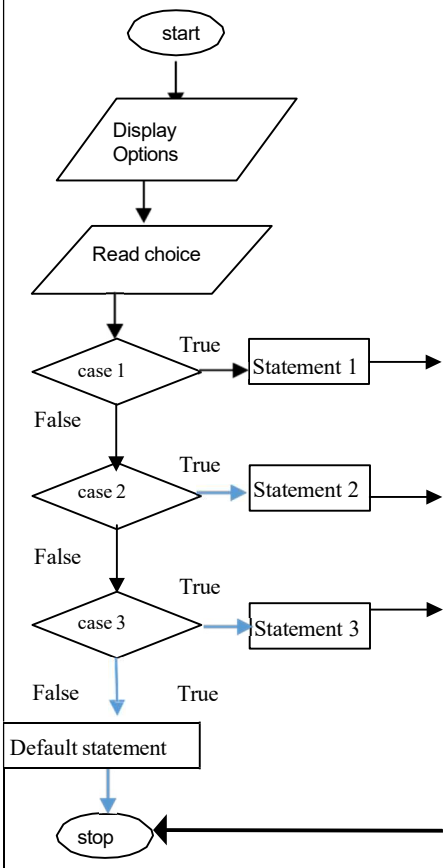
1. **Usage of switch statement**

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:

Output:

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

Program:

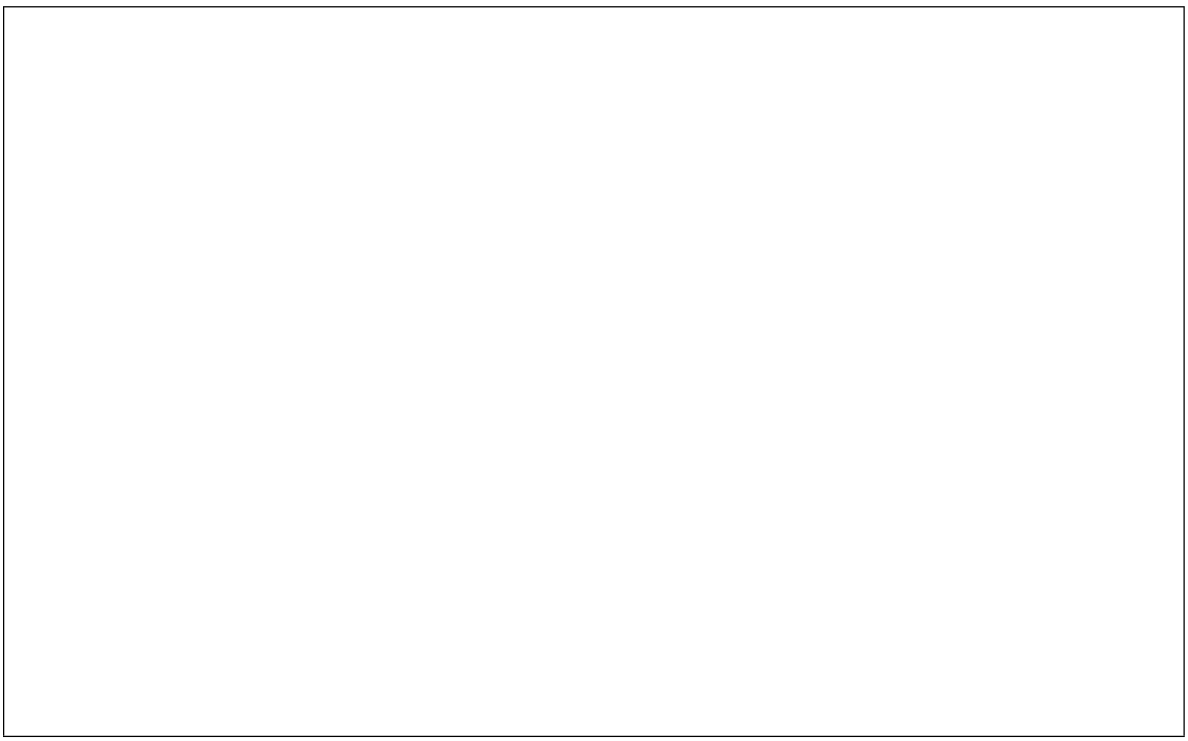
Output:

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:

Output:



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:

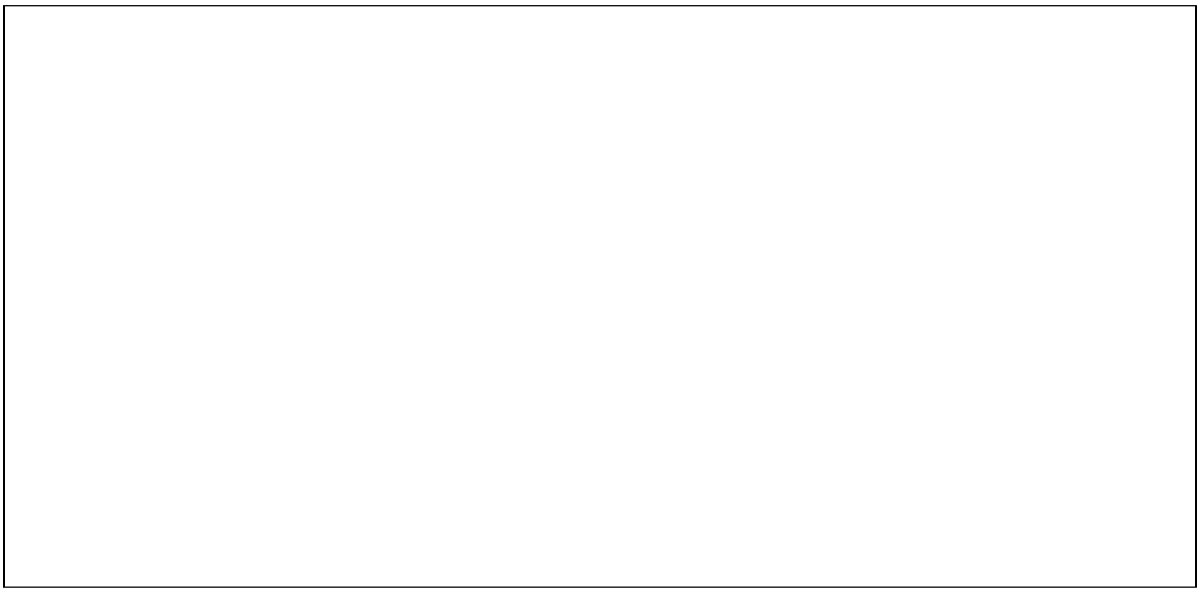
Output:

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:

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:

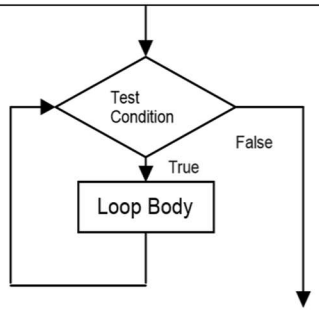
You should read following topics before starting this exercise

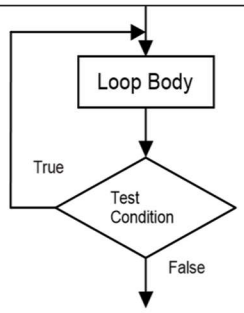
1. Different types of loop structures in C.
2. Syntax and usage of these statements.

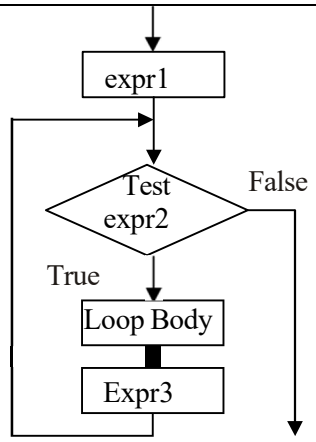
We need to perform certain actions repeatedly for a fixed number of times or till some condition holds true. These repetitive operations are done using loop control statements. The types of loop structures supported in C are

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

Sr. No	Statement Syntax	Flowchart	Example

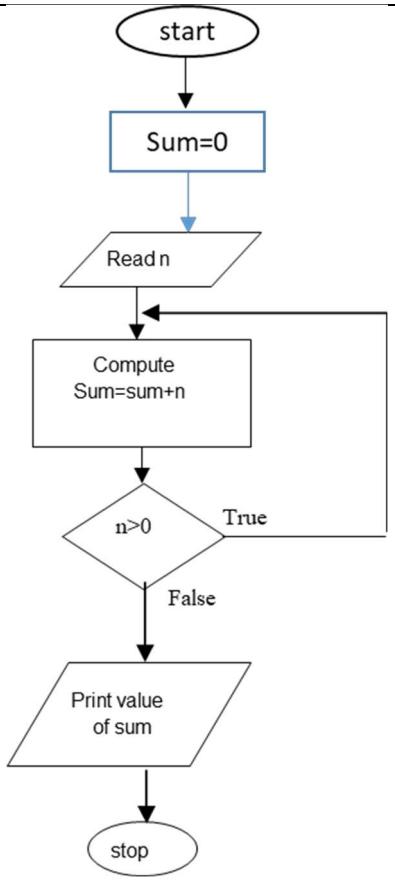
1.	**while statement**
	
	while (condition)
	{
	statement 1;
	statement 2;
	.
	.
	}
			/* accept a number*/
			scanf("%d", &n);
			/* if not a single digit */
			while (n > 9)
			{
			/* remove last digit
			n = n /10;
			}

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);
			}

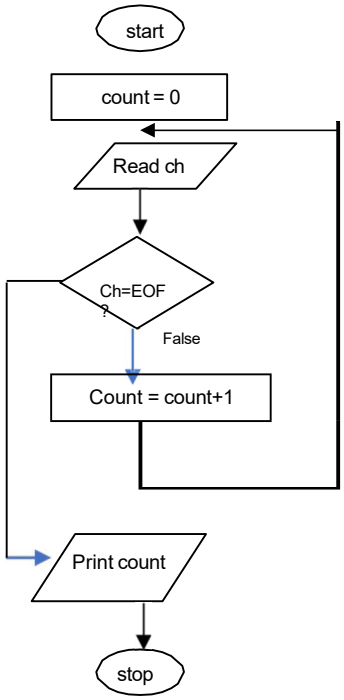
3. Sample program- to print sum of 1+2+3+….n.

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:

Output:

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

Program:

Output:

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

Program:

Output:

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

Program:

Output:

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

Program:

Output:

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:

Output:

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

546, Reverse = 645.

Program:

Output:

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:

Output:

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:

Output:

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:

Output:

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:

Output:

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:

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 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
```

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
```

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:

Output:

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
```

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:

Output:

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:

Output:

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

Program:

Output:

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:

Output:

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.

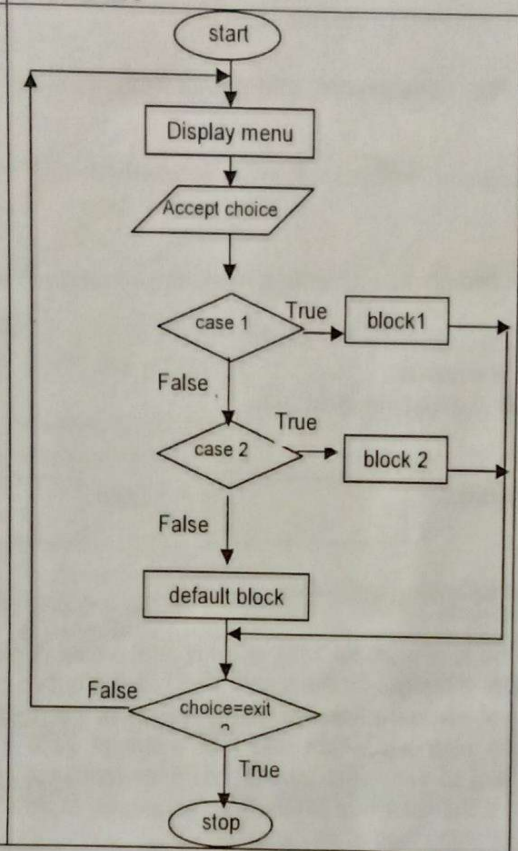
Name	Description
ceil	smallest integer not less than parameter
cos	cosine
cosh	hyperbolic cosine
exp(double x)	exponential function, computes e^x
fabs	absolute value
floor	largest integer not greater than parameter
fmod	floating point remainder
log	natural logarithm
log10	base-10 logarithm
pow(x,y)	compute 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.

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.

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:

Output:

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:

Output:

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:

Output:

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

Program:

Output:

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:

Output:

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

Program:

Output:

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

Program:

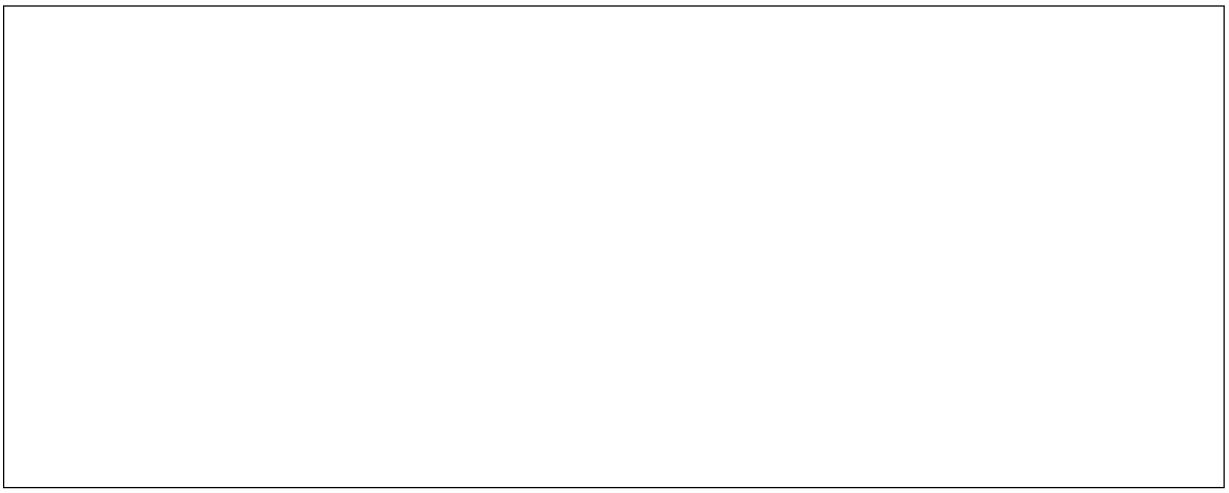
Output:

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:

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:

Output:

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:

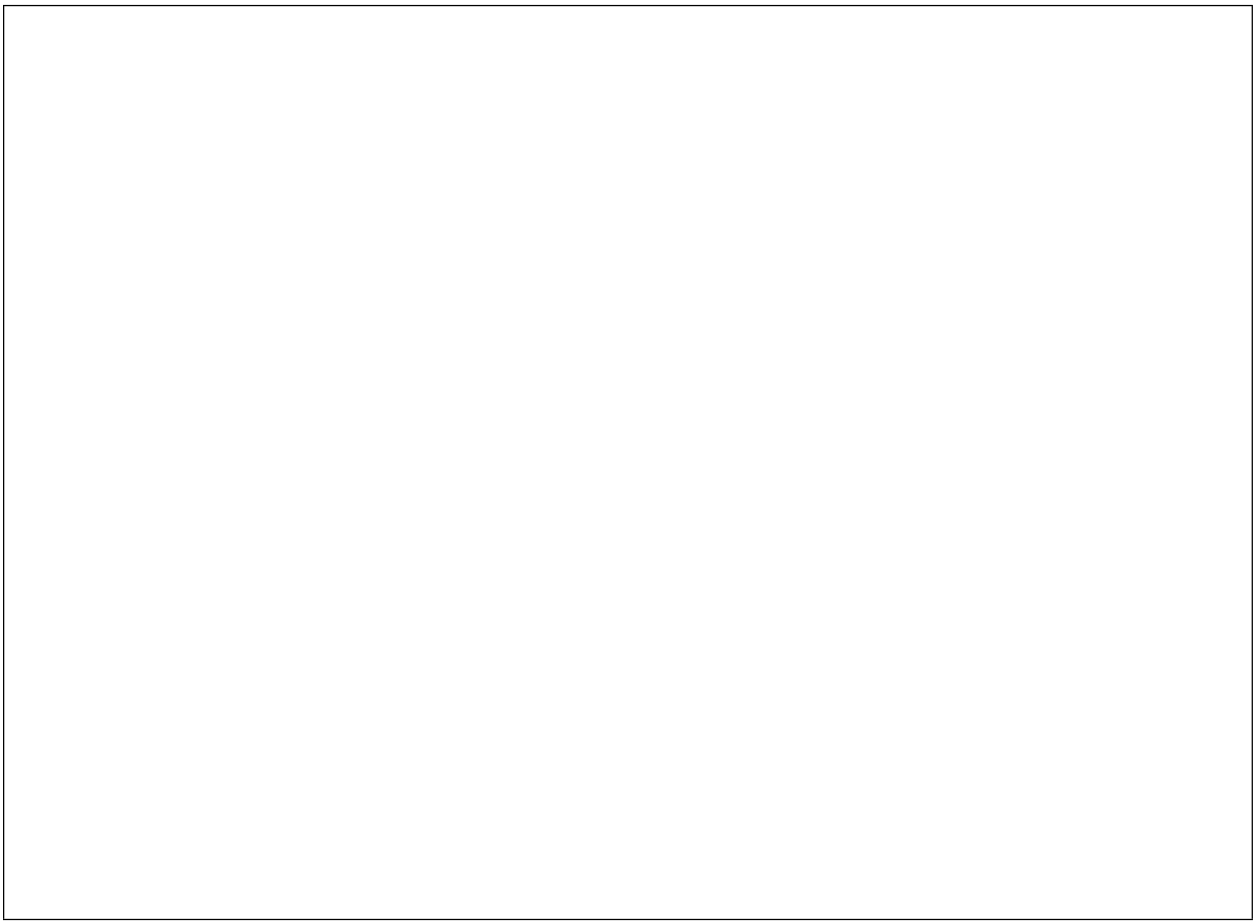
Output:

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:

Output:



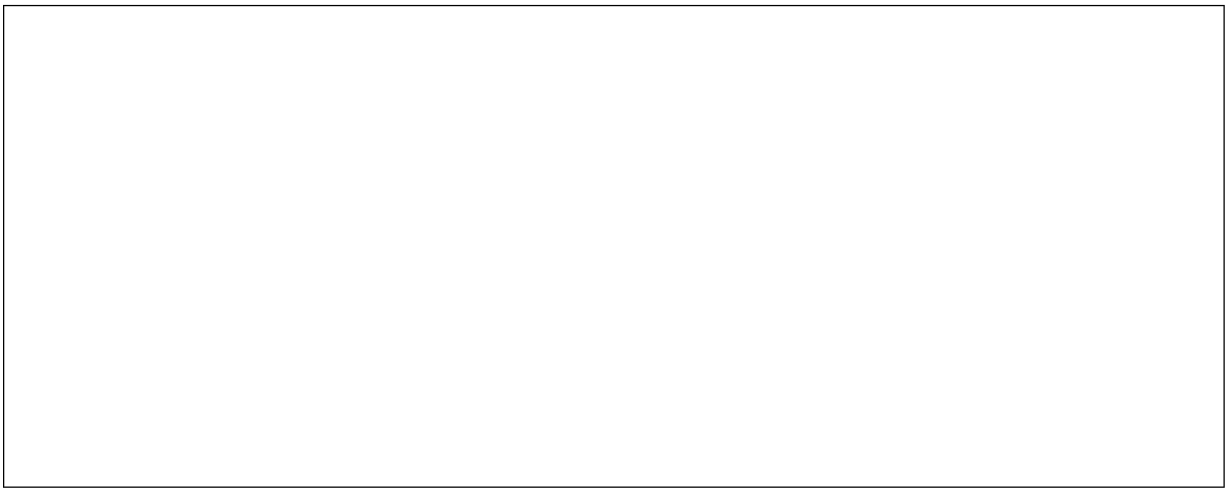
2. Write a program to accept an m X n matrix and display an m+1 X n+1 matrix such that the $m+1^{th}$ row contains the sum of all elements of corresponding row and the $n+1^{th}$ column contains the sum of elements of the corresponding column.

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:

Output:



Assignment Evaluation

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