EXPERIMENT:1-8

NAME-SIDDHANT RAI

BATCH-77

SAPID-590029215

Experiment 1: Installation, Environment Setup and Starting with C Language

1. Write a C program to print "Hello World"

- ALGORITHM:

1: Start the program

2: Display the message **"Hello World"** on the screen

3: Return 0 to indicate successful execution

4: Stop the program

PSUEDOCODE:

```
START
    PRINT "Hello World"
    RETURN 0
END
```

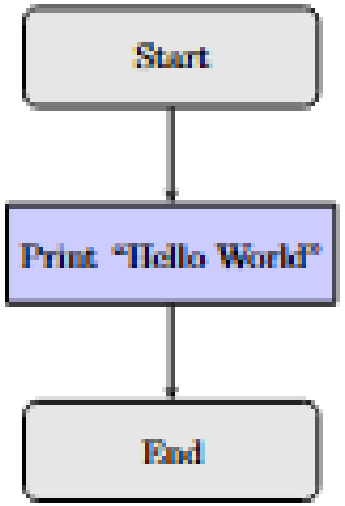
INPUT:

```c
#include <stdio.h>
int main() {
    printf("Hello World\n");
    return 0;
}
```

OUTPUT:

```
Hello World


=== Code Execution Successful ===
```

FLOWCHART:



2. Write a C program to print the address in multiple lines (new line)

- **Algorithm**

1. Start the program.
2. Print the first line of the address: *"123 Example Street"*.
3. Print the second line: *"City Name"*.
4. Print the third line: *"State Name"*.
5. Print the fourth line: *"Country"*.
6. End the program.

PSUEDOCODE:

```
BEGIN
    PRINT "123 Example Street"
    PRINT "City Name"
    PRINT "State Name"
    PRINT "Country"
END
```

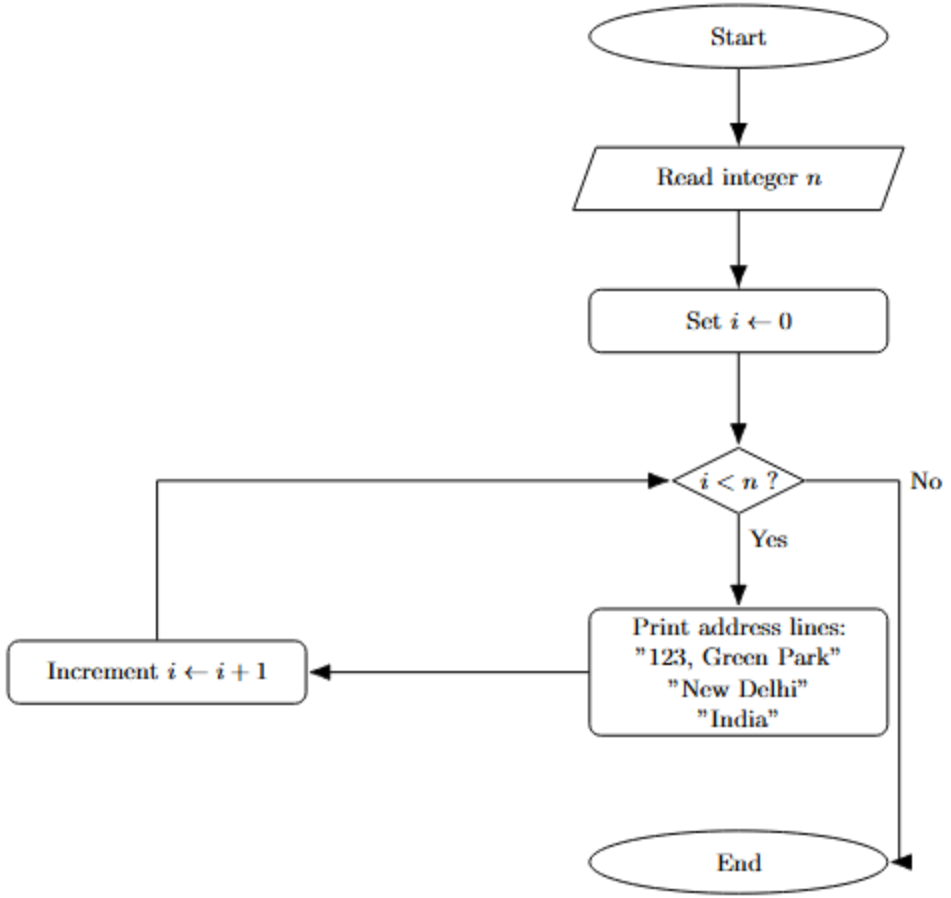
INPUT:

```c
#include <stdio.h>
int main() {
    int n;
    printf("Enter how many times to print the address: ");
    scanf("%d", &n);
    for(int i = 0; i < n; i++) {
        printf("123, Green Park\n");
        printf("New Delhi\n");
        printf("India\n\n");
    }
    return 0;
}
```

OUTPUT:

```
Enter how many times to print the address: 5
123, Green Park
New Delhi
India

123, Green Park
New Delhi
India

123, Green Park
New Delhi
India

123, Green Park
New Delhi
India

123, Green Park
New Delhi
India


=== Code Execution Successful ===
```

FLOWCHART:


Start
Read integer n
Set $i \leftarrow 0$
$i < n$?
No
Yes
Print address lines:
"123, Green Park"
"New Delhi"
"India"
Increment $i \leftarrow i + 1$
End

3. Write a program that prompts the user to enter their name and age

Algorithm

1. Start
2. Declare a character array `name` and an integer variable `age`
3. Display: *"Enter your name:"*
4. Read the value into `name`
5. Display: *"Enter your age:"*
6. Read the value into `age`
7. Display the entered name and age
8. Stop

PSUEDOCODE:

```
START

DECLARE name as string (size 50)
DECLARE age as integer

PRINT "Enter your name: "
READ name

PRINT "Enter your age: "
READ age

PRINT "Name: " + name
PRINT "Age: " + age

END
```

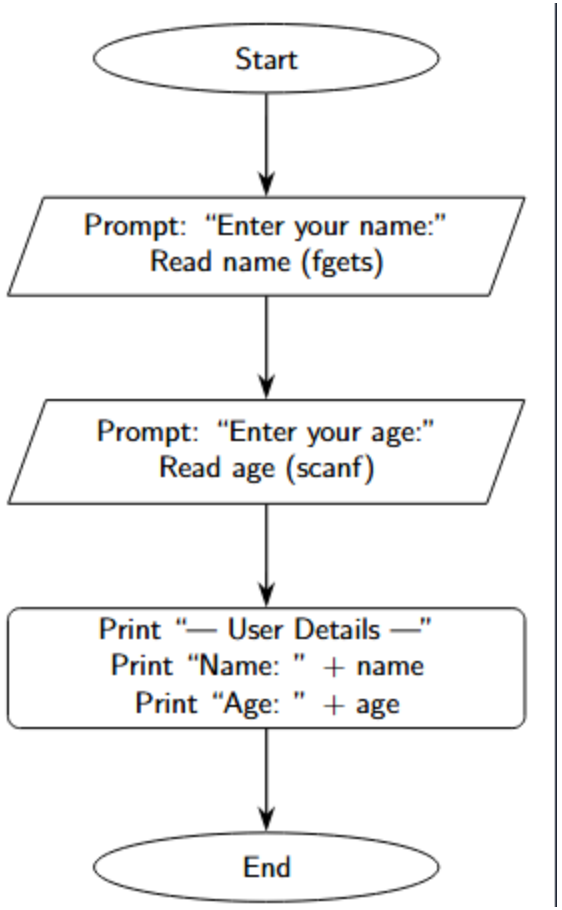
INPUT:

```c
#include <stdio.h>
int main() {
    char name[50];
    int age;
    printf("Enter your name: ");
    fgets(name, sizeof(name), stdin);
    printf("Enter your age: ");
    scanf("%d", &age);
    printf("\n--- User Details ---\n");
    printf("Name: %s", name);
    printf("Age: %d\n", age);
    return 0;
}
```

OUTPUT:

```
Enter your name: Siddhant
Enter your age: 18

--- User Details ---
Name: Siddhant
Age: 18


--- Code Execution Successful ---
```

FLOWCHART:


Start
Prompt: "Enter your name:"
Read name (fgets)
Prompt: "Enter your age:"
Read age (scanf)
Print "— User Details —"
Print "Name: " + name
Print "Age: " + age
End

4. Write a C program to add two numbers, take numbers from the user

Algorithm

1. Start
2. Declare three integer variables: `a`, `b`, and `sum`
3. Display “Enter first number:”
4. Read the value of `a`
5. Display “Enter second number:”
6. Read the value of `b`
7. Compute `sum = a + b`
8. Display the sum
9. End

PSUEDOCODE:

```
START
DECLARE a, b, sum as integers

DISPLAY "Enter first number: "
READ a

DISPLAY "Enter second number: "
READ b

SET sum = a + b

DISPLAY "Sum = " + sum

END
```

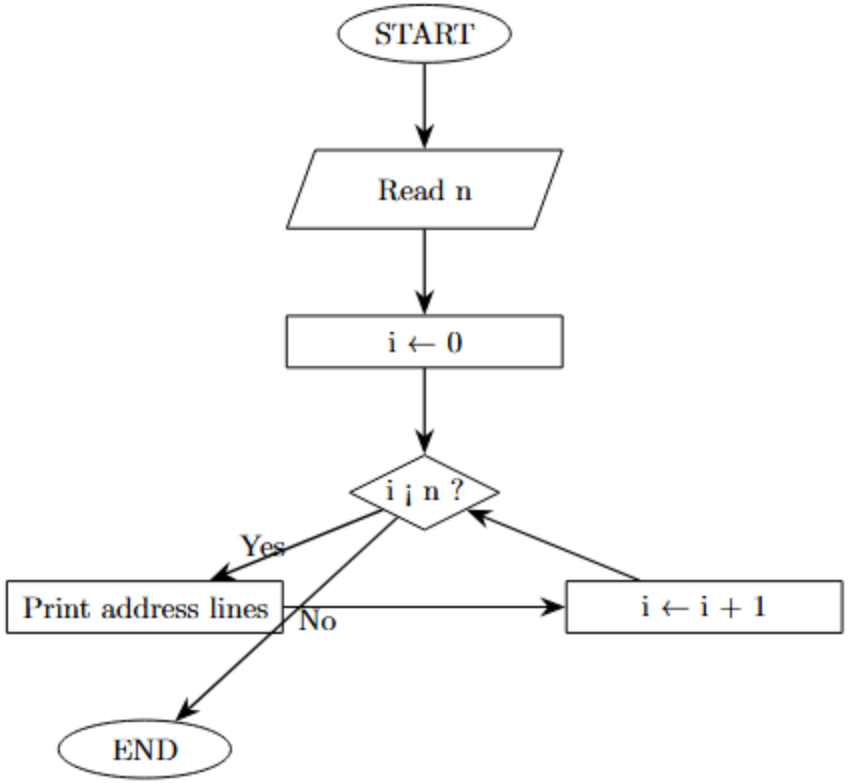
INPUT:

```c
#include <stdio.h>
int main() {
    int a, b, sum;
    printf("Enter first number: ");
    scanf("%d", &a);
    printf("Enter second number: ");
    scanf("%d", &b);
    sum = a + b;
    printf("Sum = %d\n", sum);
    return 0;
}
```

OUTPUT:

```
Enter first number: 32
Enter second number: 43
Sum = 75


=== Code Execution Successful ===
```

FLOWCHART:



EXPERIMENT 2:Operator

1. WAP a C program to calculate the area and perimeter of a rectangle based on its length and width.

ALGORITHM:

1. Start
2. Declare variables: length, width, area, perimeter
3. Read length
4. Read width
5. Compute area = length × width
6. Compute perimeter = 2 × (length + width)
7. Display area
8. Display perimeter
9. End

PSUEDOCODE:

```
START

DECLARE length, width, area, perimeter

PRINT "Enter length:"
READ length

PRINT "Enter width:"
READ width

area ← length * width
perimeter ← 2 * (length + width)

PRINT "Area =", area
PRINT "Perimeter =", perimeter

END
```

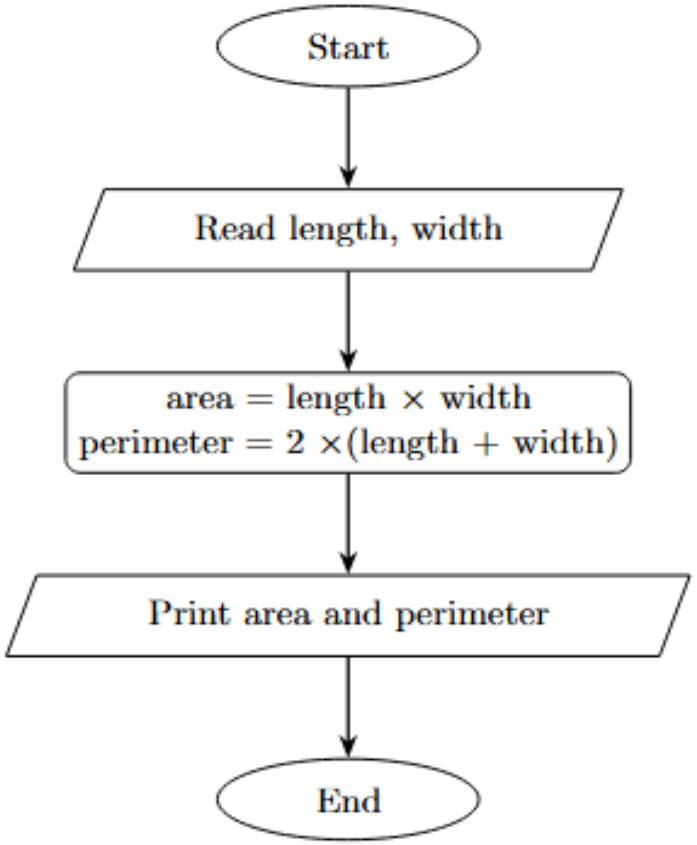
INPUT:

```c
#include <stdio.h>
int main() {
    float length, width, area, perimeter;
    printf("Enter length of the rectangle: ");
    scanf("%f", &length);
    printf("Enter width of the rectangle: ");
    scanf("%f", &width);
    area = length * width;
    perimeter = 2 * (length + width);
    printf("Area of the rectangle = %.2f\n", area);
    printf("Perimeter of the rectangle = %.2f\n", perimeter);
    return 0;
}
```

OUTPUT:

```
Enter length of the rectangle: 12
Enter width of the rectangle: 14
Area of the rectangle = 168.00
Perimeter of the rectangle = 52.00


=== Code Execution Successful ===
```

FLOWCHART:


Start
Read length, width
area = length × width
perimeter = 2 ×(length + width)
Print area and perimeter
End

2. WAP a C program to convert temperature from Celsius to Fahrenheit using the formula: F = (C * 9/5) + 32.

ALGORITHM:

1. Start
2. Declare variables: `celsius`, `fahrenheit`
3. Display: “Enter temperature in Celsius:”
4. Read value into `celsius`
5. Compute Fahrenheit using formula:
6. fahrenheit=(celsius×9/5)+32\text{fahrenheit} = (celsius \times 9/5) + 32fahrenheit=(celsius×9/5)+32
7. Display the Fahrenheit value
8. Stop

PSUEDOCODE:

```
START

DECLARE celsius, fahrenheit AS FLOAT

PRINT "Enter temperature in Celsius:"
INPUT celsius

fahrenheit ← (celsius * 9 / 5) + 32

PRINT "Temperature in Fahrenheit =", fahrenheit

END
```

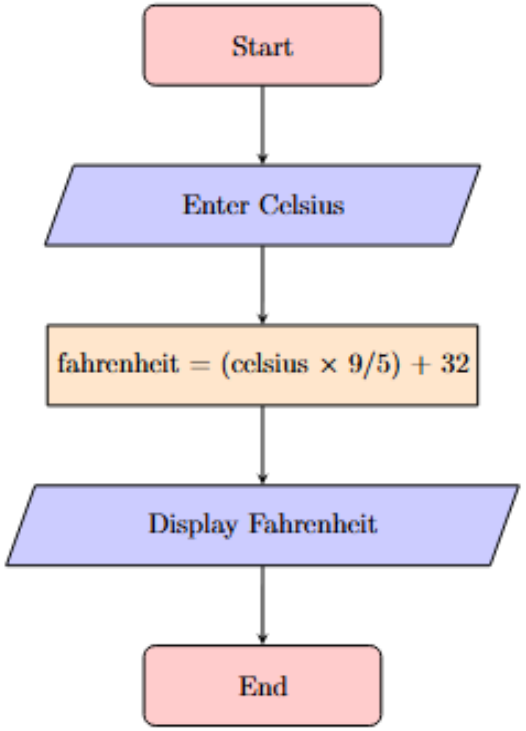
INPUT:

```c
#include <stdio.h>
int main() {
    float celsius, fahrenheit;
    printf("Enter temperature in Celsius: ");
    scanf("%f", &celsius);
    fahrenheit = (celsius * 9.0 / 5.0) + 32;
    printf("Temperature in Fahrenheit = %.2f\n", fahrenheit);
    return 0;
}
```

OUTPUT:

```
Enter temperature in Celsius: 32
Temperature in Fahrenheit = 89.60


=== Code Execution Successful ===
```

FLOWCHART:


Start
Enter Celsius
fahrenheit = (celsius × 9/5) + 32
Display Fahrenheit
End

Experiment 4: Variable and Scope of Variable

1.Declare a global variable outside all functions and use it inside various functions to understand its accessibility.

Algorithm

1. Start
2. Declare a global variable `globalVar` and set it to 10
3. Define function `display()`

   Print the value of `globalVar`
4. Define function `modify()`

   Change `globalVar` to 20

   Print the updated value
5. In `main()`

   Print initial value of `globalVar`

   Call `display()`

   Call `modify()`

   Call `display()` again
6. End

PSUEDOCODE:

```
GLOBAL globalVar = 10

FUNCTION display()
    PRINT "Inside display(): globalVar =", globalVar
END FUNCTION

FUNCTION modify()
    globalVar = 20
    PRINT "Inside modify(): globalVar changed to =", globalVar
END FUNCTION

MAIN
    PRINT "Inside main(): initial globalVar =", globalVar
    CALL display()
    CALL modify()
    CALL display()
END MAIN
```

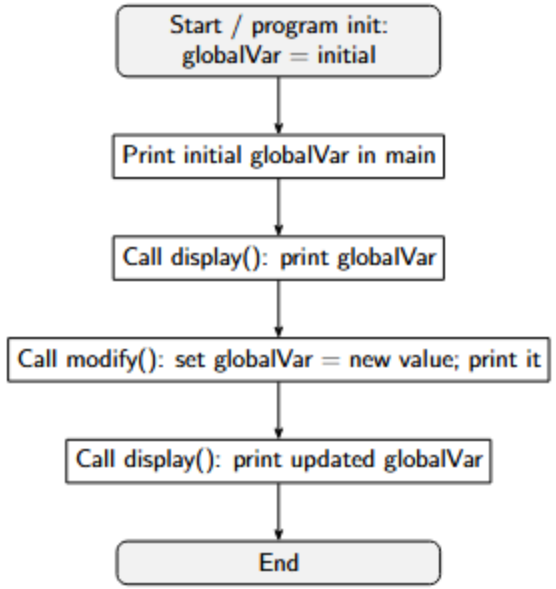
INPUT:

```c
#include <stdio.h>
int globalVar = 10;
void display() {
    printf("Inside display(): globalVar = %d\n", globalVar);
}
void modify() {
    globalVar = 20;
    printf("Inside modify(): globalVar changed to = %d\n", globalVar);
}
int main() {
    printf("Inside main(): initial globalVar = %d\n", globalVar);
    display();
    modify();
    display();
    return 0;
}
```

OUTPUT:

```
Inside main(): initial globalVar = 10
Inside display(): globalVar = 10
Inside modify(): globalVar changed to = 20
Inside display(): globalVar = 20


=== Code Execution Successful ===
```

FLOWCHART:


Start / program init:
globalVar = initial
Print initial globalVar in main
Call display(): print globalVar
Call modify(): set globalVar = new value; print it
Call display(): print updated globalVar
End

2.Declare a local variable inside a function and try to access it outside the function. Compare this with accessing the global variable from within the function.

Algorithm

1. Start
2. Declare a global variable `globalVar = 100`
3. Define function `testLocal()`
   a. `Declare a local variable localVar = 50`
   b. `Print value of localVar`
   c. `Print value of globalVar`
4. Define function `testAccess()`
   a. `Print value of globalVar`
5. In `main()`
   a. `Call testLocal()`
   b. `Call testAccess()`
6. End

PSUEDOCODE:

```
GLOBAL globalVar = 100

FUNCTION testLocal
    localVar = 50
    PRINT "Inside testLocal(): localVar =", localVar
    PRINT "Inside testLocal(): globalVar =", globalVar
END FUNCTION

FUNCTION testAccess
    PRINT "Inside testAccess(): globalVar =", globalVar
END FUNCTION

MAIN
    CALL testLocal
    CALL testAccess
END MAIN
```

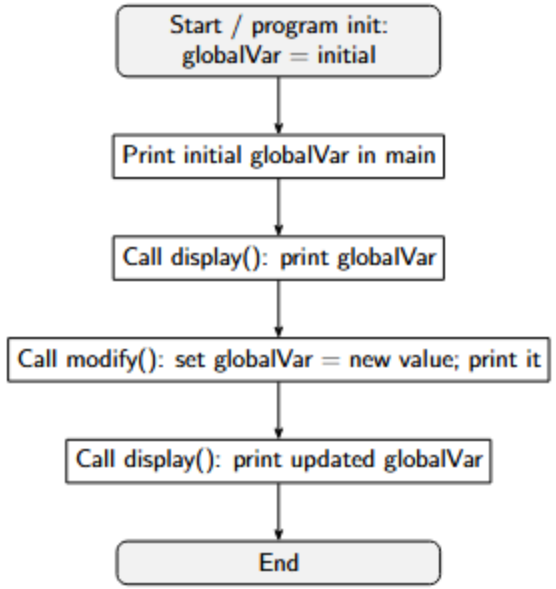
INPUT:

```c
#include <stdio.h>
int globalVar = 100;
void testLocal() {
    int localVar = 50;
    printf("Inside testLocal(): localVar = %d\n", localVar);
    printf("Inside testLocal(): globalVar = %d\n", globalVar);
}
void testAccess() {
    printf("Inside testAccess(): globalVar = %d\n", globalVar);
}
int main() {
    testLocal();
    testAccess();
    return 0;
}

```

OUTPUT:

```
Inside testLocal(): localVar = 50
Inside testLocal(): globalVar = 100
Inside testAccess(): globalVar = 100


=== Code Execution Successful ===
```

FLOWCHART:


Start / program init:
globalVar = initial
Print initial globalVar in main
Call display(): print globalVar
Call modify(): set globalVar = new value; print it
Call display(): print updated globalVar
End

3.Declare variables within different code blocks (enclosed by curly braces) and test their accessibility within and outside those blocks.

Algorithm

1. Start
2. Declare an integer variable `a` and assign value `10`
3. Print the value of `a`
4. Begin **Block 1**
    1. Declare integer variable `b` and assign value `20`
    2. Print `a`
    3. Print `b`
5. End **Block 1**
6. Begin **Block 2**
    1. Declare integer variable `c` and assign value `30`
    2. Print `a`
    3. Print `c`
7. End **Block 2**
8. End program

PSUEDOCODE:

```
BEGIN
    a ← 10
    PRINT "In main block: a =", a

    BEGIN BLOCK 1
        b ← 20
        PRINT "Inside Block 1: a =", a
        PRINT "Inside Block 1: b =", b
    END BLOCK 1

    BEGIN BLOCK 2
        c ← 30
        PRINT "Inside Block 2: a =", a
        PRINT "Inside Block 2: c =", c
    END BLOCK 2

END
```

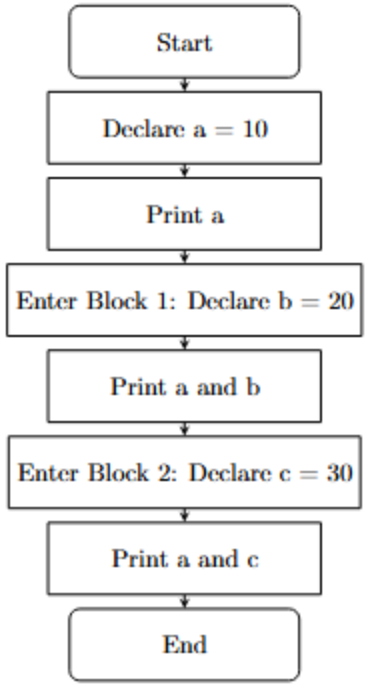
INPUT:

```c
#include <stdio.h>
int main() {
    int a = 10;  // variable in main block
    printf("In main block: a = %d\n", a);
    {
        int b = 20;
        printf("Inside Block 1: a = %d\n", a);
        printf("Inside Block 1: b = %d\n", b);
    }
    {
        int c = 30;
        printf("Inside Block 2: a = %d\n", a);
        printf("Inside Block 2: c = %d\n", c);
    }
    return 0;
}
```

OUTPUT:

```
In main block: a = 10
Inside Block 1: a = 10
Inside Block 1: b = 20
Inside Block 2: a = 10
Inside Block 2: c = 30


=== Code Execution Successful ===
```

FLOWCHART:


Start
Declare a = 10
Print a
Enter Block 1: Declare b = 20
Print a and b
Enter Block 2: Declare c = 30
Print a and c
End

4.Declare a static local variable inside a function. Observe how its value persists across function calls.

Algorithm

1. Start the program.
2. Define a function named `counter`.
3. Inside the function, declare a static variable `count` initialized to 0.
4. Increment `count` by 1.
5. Display the value of `count`.
6. In the main program, call the function `counter` three times.
7. Stop the program.

PSUEDOCODE:

```
FUNCTION counter
    DECLARE static count = 0
    INCREMENT count
    PRINT count
END FUNCTION

START
    CALL counter
    CALL counter
    CALL counter
STOP
```

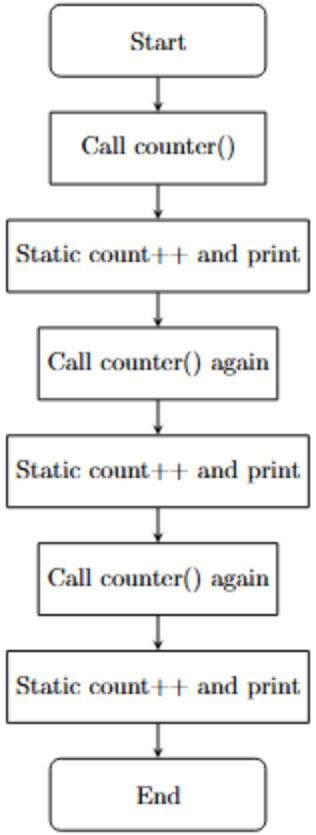
INPUT:

```c
#include <stdio.h>
void counter() {
    static int count = 0;
    count++;
    printf("Counter value: %d\n", count);
}
int main() {
    counter();
    counter();
    counter();
    return 0;
}
```

OUTPUT:

```
Counter value: 1
Counter value: 2
Counter value: 3


=== Code Execution Successful ===
```

FLOWCHART:


Start
Call counter()
Static count++ and print
Call counter() again
Static count++ and print
Call counter() again
Static count++ and print
End

Experiment 5: Array

1.WAP to read a list of integers and store it in a single dimensional array. Write a C program to print the second largest integer in a list of integers.

ALGORITHM

1. **To find the second largest number in an array**
2. Start
3. Declare n
4. Read the value of n
5. Declare array arr of size n
6. Read all n integers into the array
7. Set `largest = arr[0]`
8. Set `second_largest = arr[0]`
9. For each element from index 1 to n-1:
    1. If `arr[i] > largest`
        1. `second_largest = largest`
        2. `largest = arr[i]`
    2. Else if `arr[i] > second_largest` and `arr[i] != largest`
        1. `second_largest = arr[i]`
10. If `second_largest == largest`
    `Print "Second largest does not exist"`
    `Else`
    `Print second largest value`
11. Stop

PSUEDOCODE:

```
BEGIN
    INPUT n
    DECLARE array arr of size n

    FOR i FROM 0 TO n-1
        INPUT arr[i]
    END FOR

    SET largest = arr[0]
    SET second_largest = arr[0]

    FOR i FROM 1 TO n-1
        IF arr[i] > largest THEN
            second_largest = largest
            largest = arr[i]
        ELSE IF arr[i] > second_largest AND arr[i] != largest THEN
            second_largest = arr[i]
        END IF
    END FOR

    IF second_largest == largest THEN
        PRINT "Second largest does not exist"
    ELSE
        PRINT "Second largest integer is: ", second_largest
    END IF
END
```

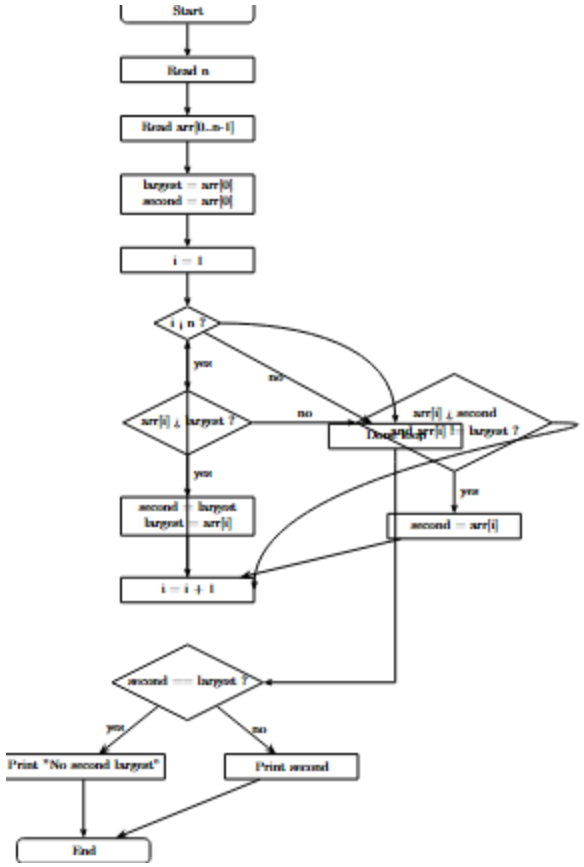
INPUT:

```c
#include <stdio.h>
int main() {
    int n;

    printf("Enter the number of elements: ");
    scanf("%d", &n);

    int arr[n];

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

    int largest = arr[0];
    int second_largest = arr[0];


    for (int i = 1; i < n; i++) {
        if (arr[i] > largest) {
            second_largest = largest;
            largest = arr[i];
        } else if (arr[i] > second_largest && arr[i] != largest) {
            second_largest = arr[i];
        }
    }


    if (second_largest == largest) {
        printf("Second largest does not exist (all numbers equal)\n");
    } else {
        printf("Second largest integer is: %d\n", second_largest);
    }

    return 0;
}
```

OUTPUT:

```
Enter the number of elements: 4
Enter 4 integers:
4 5 6 7
Second largest integer is: 6


=== Code Execution Successful ===
```

FLOWCHART:


Start
Read n
Read arr[0..n-1]
largest = arr[0]
second = arr[0]
i = 1
i < n ?
yes
no
arr[i] > largest ?
yes
second = largest
largest = arr[i]
arr[i] > second
and arr[i] != largest ?
yes
second = arr[i]
i = i + 1
second == largest ?
yes
Print "No second largest"
no
Print second
End

2.WAP to read a list of integers and store it in a single dimensional array. Write a C program to count and display positive, negative, odd, and even numbers in an array.

ALGORITHM:

1. Start
2. Declare an integer `n`
3. Ask the user to enter the number of elements
4. Read `n`
5. Declare an integer array of size `n`
6. Read all `n` integers into the array
7. Initialize counters:
8. `positive = 0`
9. `negative = 0`
10. `odd = 0`
11. `even = 0`
12. Loop through each element of the array
13. If element ≥ 0 → increment `positive`
14. Else → increment `negative`
15. If element % 2 == 0 → increment `even`
16. Else → increment `odd`
17. Display the values of all four counters
18. Stop

PSUEDOCODE:

```
BEGIN
    DECLARE n, i
    PRINT "Enter the number of elements"
    READ n

    DECLARE array of size n

    PRINT "Enter n integers"
    FOR i = 0 TO n-1
        READ arr[i]
    END FOR

    SET positive = 0
    SET negative = 0
    SET odd = 0
    SET even = 0

    FOR i = 0 TO n-1
        IF arr[i] >= 0 THEN
            positive = positive + 1
        ELSE
            negative = negative + 1
        ENDIF
            IF arr[i] MOD 2 == 0 THEN
                even = even + 1
            ELSE
                odd = odd + 1
            ENDIF
      END FOR

      PRINT "Positive numbers:", positive
      PRINT "Negative numbers:", negative
      PRINT "Odd numbers:", odd
      PRINT "Even numbers:", even
END
```

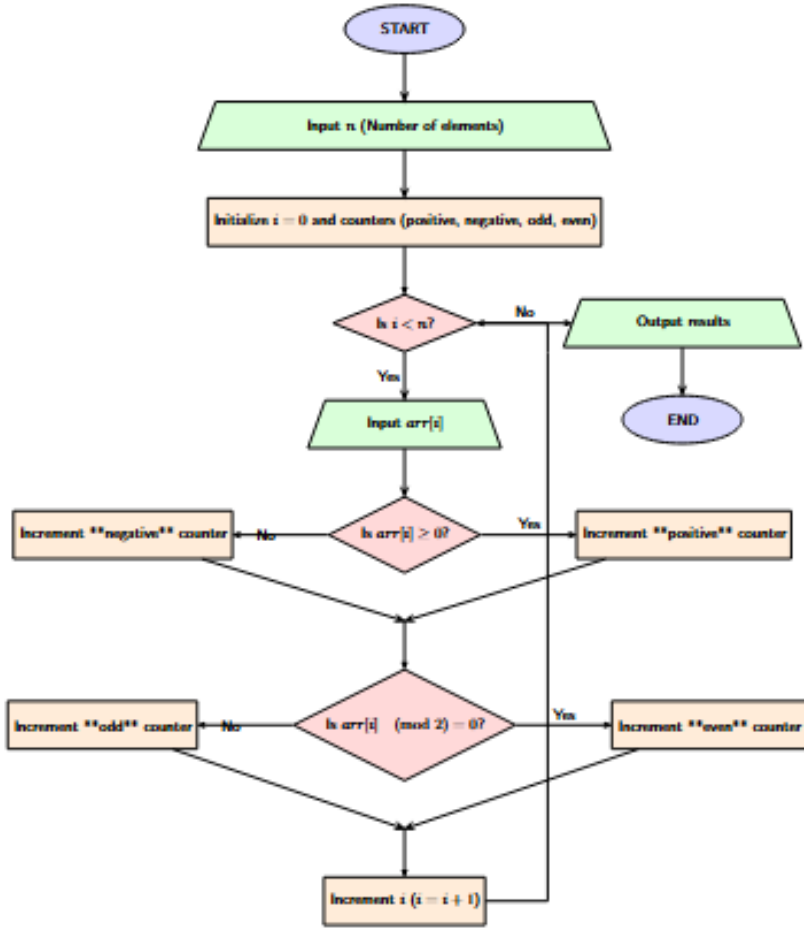
INPUT:

```c
#include <stdio.h>
int main() {
    int n;
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    int arr[n];

    printf("Enter %d integers:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    int positive = 0, negative = 0, odd = 0, even = 0;
    for (int i = 0; i < n; i++) {
        if (arr[i] >= 0)
            positive++;
        else
            negative++;

        if (arr[i] % 2 == 0)
            even++;
        else
            odd++;
    }
    printf("\n--- Result ---\n");
    printf("Positive numbers: %d\n", positive);
    printf("Negative numbers: %d\n", negative);
    printf("Odd numbers: %d\n", odd);
    printf("Even numbers: %d\n", even);
    return 0;
```

OUTPUT:

```
Enter the number of elements: 4
Enter 4 integers:
3 4 5 6

--- Result ---
Positive numbers: 4
Negative numbers: 0
Odd numbers: 2
Even numbers: 2


=== Code Execution Successful ===
```

FLOWCHART:


START
Input n (Number of elements)
Initialize i = 0 and counters (positive, negative, odd, even)
Is i < n?
No
Output results
END
Yes
Input arr[i]
Is arr[i] ≥ 0?
No
Increment "negative" counter
Yes
Increment "positive" counter
Is arr[i] (mod 2) = 0?
No
Increment "odd" counter
Yes
Increment "even" counter
Increment i (i = i + 1)

3.WAP to read a list of integers and store it in a single dimensional array. Write a C program to find the frequency of a particular number in a list of integers.

- **ALGORITHM**

- **Step 1:** Start

  Step 2: Declare integers `n`, `key`, `count` and an array `arr[]`

  Step 3: Set `count = 0`

  Step 4: Read the value of `n` (number of elements)

  Step 5: Read `n` integers into the array `arr[]`

  Step 6: Read the number `key` whose frequency is to be found

  Step 7: Repeat for `i = 0` to `n - 1`:

  - If `arr[i] == key`, increment `count`

  Step 8: Display the frequency

  Step 9: End

PSUEDOCODE:

```
BEGIN

    SET count ← 0

    OUTPUT "Enter the number of elements"
    INPUT n

    DECLARE array arr of size n

    OUTPUT "Enter n integers"
    FOR i ← 0 TO n-1 DO
        INPUT arr[i]
    END FOR

    OUTPUT "Enter the number whose frequency you want to find"
    INPUT key

    FOR i ← 0 TO n-1 DO
        IF arr[i] = key THEN
            count ← count + 1
        END IF
    END FOR

    OUTPUT "Frequency of", key, "is", count

END
```

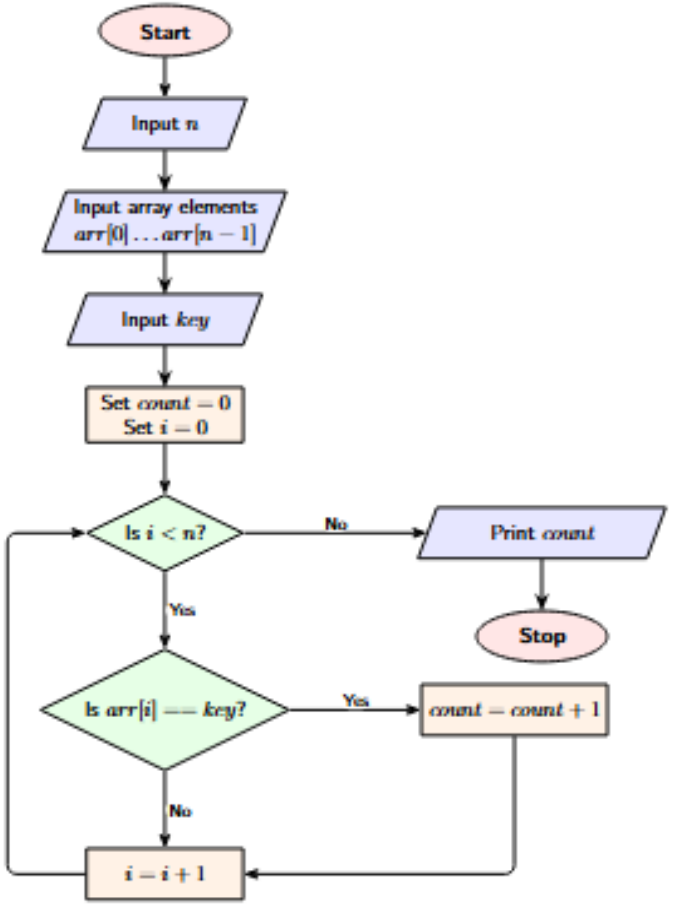
INPUT:

```c
#include <stdio.h>
int main() {
    int n, key, count = 0;
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter %d integers:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Enter the number whose frequency you want to find: ");
    scanf("%d", &key);
    for (int i = 0; i < n; i++) {
        if (arr[i] == key)
            count++;
    }
    printf("\nFrequency of %d is: %d\n", key, count);

    return 0;
}
```

OUTPUT:

```
Enter the number of elements: 4
Enter 4 integers:
1 2 3 4
Enter the number whose frequency you want to find: 3

Frequency of 3 is: 1


=== Code Execution Successful ===
```

FLOWCHART:


Start
Input n
Input array elements $arr[0] \dots arr[n-1]$
Input key
Set $count = 0$
Set $i = 0$
Is $i < n$?
No
Print $count$
Stop
Yes
Is $arr[i] == key$?
Yes
$count = count + 1$
No
$i = i + 1$

4.WAP that reads two matrices A (m x n) and B (p x q) and computes the product A and B. Read matrix A and matrix B in row major order respectively. Print both the input matrices and resultant matrix with suitable headings and output should be in matrix format only. Program must check the compatibility of orders of the matrices for multiplication. Report appropriate message in case of incompatibility.

ALGORITHM:

1. Read `m` and `n`.
2. Read `p` and `q`.
3. If `n != p` then print incompatibility message and stop.
4. Allocate matrices A (`m×n`), B (`p×q`) and C (`m×q`).
5. Read elements of A in row-major order.
6. Read elements of B in row-major order.
7. For each `i` from `0` to `m-1`:
8. For each `j` from `0` to `q-1`:

   Set `C[i][j] = 0`.

   For each `k` from `0` to `n-1`:

   - `C[i][j] += A[i][k] * B[k][j]`.
   - Print Matrix A with heading.
   - Print Matrix B with heading.
   - Print Resultant Matrix C with heading

PSUEDOCODE:

```
BEGIN
  READ m, n
  READ p, q

  IF n != p THEN
    PRINT "Matrix multiplication NOT possible!"
    PRINT "Reason: Columns of A (n) != Rows of B (p)"
    STOP
  ENDIF

  DECLARE A[m][n], B[p][q], C[m][q]

  PRINT "Enter elements of Matrix A (m × n) in row-major order:"
  FOR i = 0 TO m-1 DO
    FOR j = 0 TO n-1 DO
      READ A[i][j]
    ENDFOR
  ENDFOR

  [illegible]
   FOR j = 0 TO q-1 DO
     READ B[i][j]
   ENDFOR
  ENDFOR

FOR i = 0 TO m-1 DO
  FOR j = 0 TO q-1 DO
    C[i][j] = 0
    FOR k = 0 TO n-1 DO
      C[i][j] = C[i][j] + A[i][k] * B[k][j]
    ENDFOR
  ENDFOR
ENDFOR

PRINT "Matrix A (m × n):"
FOR i = 0 TO m-1 DO
  FOR j = 0 TO n-1 DO
    PRINT A[i][j] WITH SPACING
  ENDFOR
  PRINT NEWLINE
ENDFOR

PRINT "Matrix B (p × q):"
FOR i = 0 TO p-1 DO
  FOR j = 0 TO q-1 DO
    PRINT B[i][j] WITH SPACING
  ENDFOR
  PRINT NEWLINE
```

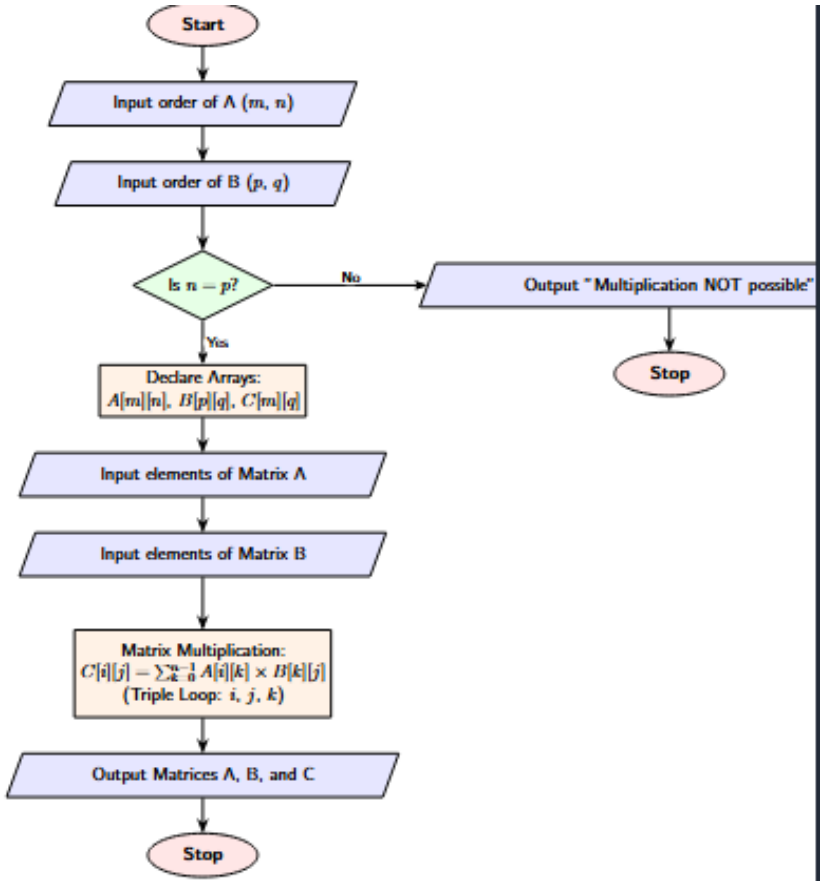
INPUT:

```c
#include <stdio.h>
int main() {
    int m, n, p, q;
    printf("Enter the order of Matrix A (m n): ");
    scanf("%d %d", &m, &n);
    printf("Enter the order of Matrix B (p q): ");
    scanf("%d %d", &p, &q);
    if (n != p) {
        printf("\nMatrix multiplication NOT possible!\n");
        printf("Reason: Columns of A (%d) != Rows of B (%d)\n", n, p);
        return 0;
    }
    int A[m][n], B[p][q], C[m][q];
    printf("\nEnter elements of Matrix A (%d x %d) in row-major order:\n", m, n);
    for (int i = 0; i < m; i++)
        for (int j = 0; j < n; j++)
            scanf("%d", &A[i][j]);

    printf("\nEnter elements of Matrix B (%d x %d) in row-major order:\n", p, q);
    for (int i = 0; i < p; i++)
        for (int j = 0; j < q; j++)
            scanf("%d", &B[i][j]);

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < q; j++) {
            C[i][j] = 0;
            for (int k = 0; k < n; k++)
                C[i][j] += A[i][k] * B[k][j];
        }
    }

    printf("\nMatrix A (%d x %d):\n", m, n);
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++)
            printf("%5d", A[i][j]);
        printf("\n");
    }

    printf("\nMatrix B (%d x %d):\n", p, q);
    for (int i = 0; i < p; i++) {
        for (int j = 0; j < q; j++)
            printf("%5d", B[i][j]);
        printf("\n");
    }
```

OUTPUT:

```
Enter the order of Matrix A (m n): 2 2
Enter the order of Matrix B (p q): 2 2

Enter elements of Matrix A (2 × 2) in row-major order:
2 2
2 2

Enter elements of Matrix B (2 × 2) in row-major order:
2
2 2
2

Matrix A (2 × 2):
    2    2
    2    2

Matrix B (2 × 2):
    2    2
    2    2

Resultant Matrix (A × B) (2 × 2):
    8    8
    8    8


=== Code Execution Successful ===
```

FLOWCHART:


Start
Input order of A (m, n)
Input order of B (p, q)
Is $n = p$?
No
Output "Multiplication NOT possible"
Stop
Yes
Declare Arrays:
$A[m][n]$, $B[p][q]$, $C[m][q]$
Input elements of Matrix A
Input elements of Matrix B
Matrix Multiplication:
$C[i][j] = \sum_{k=0}^{n-1} A[i][k] \times B[k][j]$
(Triple Loop: i, j, k)
Output Matrices A, B, and C
Stop

Experiment 6

1.Develop a recursive and non-recursive function FACT(num) to find the factorial of a number, n!, defined by FACT(n) = 1, if n = 0. Otherwise, FACT(n) = n x FACT(n−1). Using this function, write a C program to compute the binomial coefficient. Tabulate the results for different values of n and r with suitable messages.

ALGORITHM:

1. Start
2. Read the values of **n** and **r**
3. If **r > n**, print "Invalid input" and stop
4. Compute **n!** using:
5. recursive FACT(n)
6. non-recursive FACT(n)
7. Compute **r!** and **(n−r)!** Similarly
8. Display **nCr**
9. Create a table by running loops:
    1. For i = 0 to n
    1. For j = 0 to i
        1. Compute iCj using factorial function
        2. Print the value
10. Stop

PSUEDOCODE:

```
FUNCTION FACT_RECURSIVE(n)
    IF n == 0 THEN
        RETURN 1
    ELSE
        RETURN n * FACT_RECURSIVE(n - 1)
    ENDIF
END FUNCTION
```

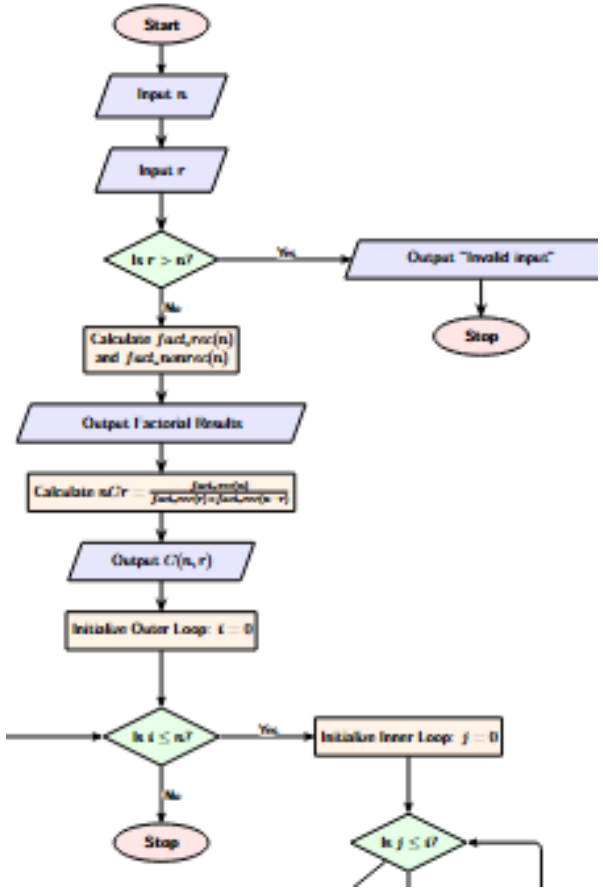
INPUT:

```c
#include <stdio.h>
long fact_rec(long n) {
    if (n == 0)
        return 1;
    return n * fact_rec(n - 1);
long fact_nonrec(long n) {
    long f = 1;
    for (long i = 1; i <= n; i++)
        f *= i;
    return f;
long binomial(long n, long r) {
    return fact_rec(n) / (fact_rec(r) * fact_rec(n - r));
int main() {
    int n, r;
    printf("Enter n value: ");
    scanf("%d", &n);
    printf("Enter r value: ");
    scanf("%d", &r);
    if (r > n) {
        printf("Invalid input! r cannot be greater than n.\n");
        return 0;
    printf("\n--- FACTORIAL RESULTS ---\n");
    printf("Recursive FACT(%d)     = %ld\n", n, fact_rec(n));
    printf("Non-recursive FACT(%d) = %ld\n", n, fact_nonrec(n));
    long ncr = binomial(n, r);
    printf("\n--- BINOMIAL COEFFICIENT ---\n");
    printf("C(%d, %d) = %ld\n", n, r, ncr);
    printf("\n--- TABLE OF nCr FOR n UP TO %d ---\n", n);
    printf("  n    r     nCr\n");
    printf("----------------------\n");
```

OUTPUT:

```
Enter n value: 4
Enter r value: 4

--- FACTORIAL RESULTS ---
Recursive FACT(4)     = 24
Non-recursive FACT(4) = 24

--- BINOMIAL COEFFICIENT ---
C(4, 4) = 1

--- TABLE OF nCr FOR n UP TO 4 ---
  n    r     nCr
----------------------
  0    0       1
  1    0       1
  1    1       1
  2    0       1
  2    1       2
  2    2       1
  3    0       1
  3    1       3
  3    2       3
  3    3       1
  4    0       1
  4    1       4
  4    2       6
  4    3       4
  4    4       1
```

FLOWCHART:


Start
Input n
Input r
Is r > n?
Yes
Output "Invalid input"
Stop
No
Calculate fact_rec(n) and fact_nonrec(n)
Output Factorial Results
Output C(n, r)
Initialize Outer Loop: i = 0
Is i ≤ n?
Yes
Initialize Inner Loop: j = 0
Stop
No
Is j ≤ i?

2.Develop a recursive function GCD (num1, num2) that accepts two integer arguments. Write a C program that invokes this function to find the greatest common divisor of two given integers.

- ALGORITHM:
- **Step 1:** Start

  Step 2: Read two integers `num1` and `num2`

  Step 3: Call the recursive function `gcd(num1, num2)`

  Step 4: Inside the function:
- If `b == 0`, return `a`
- Otherwise return `gcd(b, a % b)`

  Step 5: Display the returned GCD value

  Step 6: Stop

PSUEDOCODE:

```
FUNCTION gcd(a, b)
    IF b = 0 THEN
        RETURN a
    ELSE
        RETURN gcd(b, a MOD b)
    ENDIF
END FUNCTION

START
    PRINT "Enter two integers: "
    READ num1, num2

    result ← gcd(num1, num2)

    PRINT "Greatest Common Divisor (GCD) of ", num1, " and ", num2, " = ", result
END
```

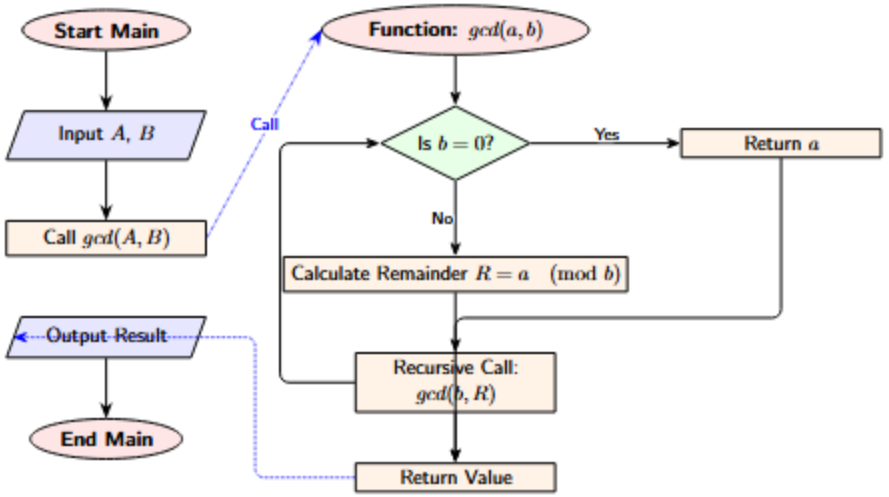
INPUT:

```c
#include <stdio.h>

int gcd(int a, int b) {
    if (b == 0)
        return a;
    return gcd(b, a % b);
}

int main() {
    int num1, num2;

    printf("Enter two integers: ");
    scanf("%d %d", &num1, &num2);

    printf("Greatest Common Divisor (GCD) of %d and %d = %d\n",
           num1, num2, gcd(num1, num2));

    return 0;
}
```

OUTPUT:

```
Enter two integers: 2 3
Greatest Common Divisor (GCD) of 2 and 3 = 1


=== Code Execution Successful ===
```

FLOWCHART:


Start Main
Input A, B
Call $gcd(A, B)$
Call
Function: $gcd(a, b)$
Is $b = 0$?
Yes
Return a
No
Calculate Remainder $R = a \pmod b$
Recursive Call: $gcd(b, R)$
Return Value
Output Result
End Main

3.Develop a recursive function FIBO(num) that accepts an integer argument. Write a C program that invokes this function to generate the Fibonacci sequence up to num.

- **ALGORITHM**
- **Step 1:** Start

  Step 2: Define a recursive function `FIBO(num)`

  `If num = 0 → return 0`

  `If num = 1 → return 1`

  `Else return FIBO(num - 1) + FIBO(num - 2)`

  Step 3: In `main()`, declare integer `n`

  Step 4: Ask the user to enter the number of terms

  Step 5: Read `n`

  Step 6: Print message “Fibonacci sequence up to n terms”

  Step 7: Loop `i` from 0 to n-1

  Call `FIBO(i)` and print result

  Step 8: En

PSUEDOCODE:

```
FUNCTION FIBO(num)
    IF num == 0 THEN
        RETURN 0
    ELSE IF num == 1 THEN
        RETURN 1
    ELSE
        RETURN FIBO(num - 1) + FIBO(num - 2)
END FUNCTION

BEGIN
    DECLARE n
    PRINT "Enter the number of terms:"
    READ n

    PRINT "Fibonacci sequence up to n terms:"

    FOR i ← 0 TO n-1 DO
        PRINT FIBO(i)
    END FOR
END
```

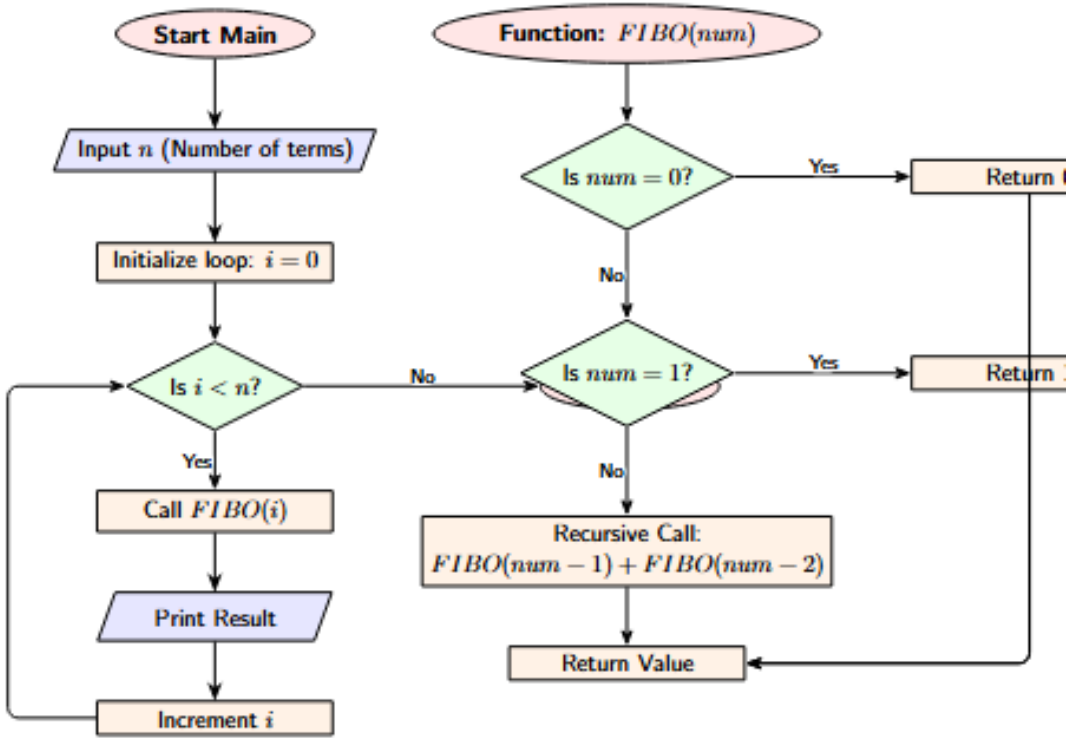
INPUT:

```c
#include <stdio.h>

int FIBO(int num) {
    if (num == 0)
        return 0;
    if (num == 1)
        return 1;
    return FIBO(num - 1) + FIBO(num - 2);
}

int main() {
    int n;

    printf("Enter the number of terms: ");
    scanf("%d", &n);

    printf("Fibonacci sequence up to %d terms:\n", n);
    for (int i = 0; i < n; i++) {
        printf("%d ", FIBO(i));
    }

    return 0;
}
```

OUTPUT:

```
Enter the number of terms: 5
Fibonacci sequence up to 5 terms:
0 1 1 2 3

=== Code Execution Successful ===
```

FLOWCHART:


Start Main
Input n (Number of terms)
Initialize loop: $i = 0$
Is $i < n$?
Yes
Call $FIBO(i)$
Print Result
Increment i
No
Function: $FIBO(num)$
Is $num = 0$?
Yes
Return
No
Is $num = 1$?
Yes
Return
No
Recursive Call:
$FIBO(num - 1) + FIBO(num - 2)$
Return Value

4.Develop a C function ISPRIME(num) that accepts an integer argument and returns 1 if the argument is prime, and 0 otherwise. Write a C program that invokes this function to generate prime numbers between the given ranges

- ALGORITHM:
- **Step 1:**Start.
- **Step 2:**Declare two integer variables: `low` and `high`.
- **Step 3:**Read the values of `low` and `high` from the user.
- **Step 4:**Display a message indicating that prime numbers within the given range will be printed.
- **Step 5:**Repeat for each number `i` from `low` to `high`:
- Call the function **ISPRIME(i)**.
- If ISPRIME(i) returns **1**, print the number `i`.
- **Step 6:**
- End.

PSUEDOCODE:

```
FUNCTION ISPRIME(num)
    IF num < 2 THEN
        RETURN 0
    END IF
    FOR i FROM 2 TO floor(sqrt(num)) DO
        IF num % i == 0 THEN
            RETURN 0
        END IF
    END FOR
    RETURN 1
END FUNCTION

MAIN
    READ low, high
    PRINT "Prime numbers between low and high are:"
    FOR n FROM low TO high DO
        IF ISPRIME(n) == 1 THEN
            PRINT n
        END IF
    END FOR
END MAIN
```

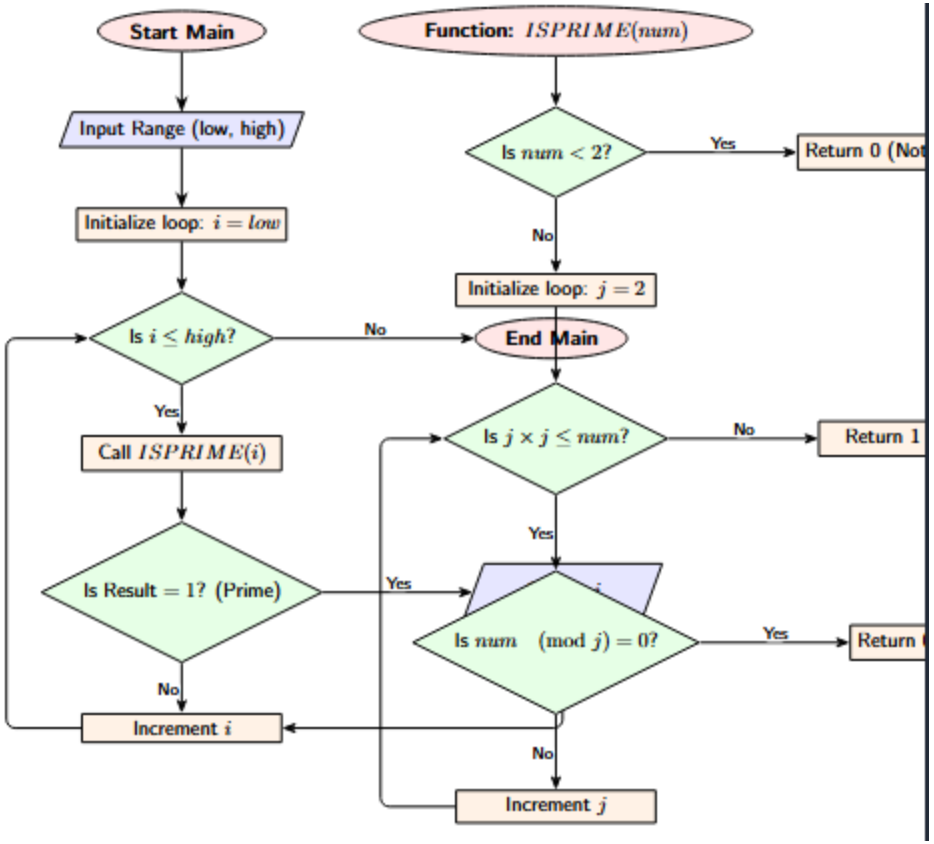
INPUT:

```c
#include <stdio.h>
int ISPRIME(int num) {
    if (num < 2)
        return 0;
    for (int i = 2; i * i <= num; i++) {
        if (num % i == 0)
            return 0;
    }
    return 1;
}
int main() {
    int low, high;
    printf("Enter the range (low high): ");
    scanf("%d %d", &low, &high);
    printf("Prime numbers between %d and %d are:\n", low, high);
    for (int i = low; i <= high; i++) {
        if (ISPRIME(i))
            printf("%d ", i);
    }
    return 0;
}
```

OUTPUT:

```
Enter the range (low high): 7 8
Prime numbers between 7 and 8 are:
7

=== Code Execution Successful ===
```

FLOWCHART:


Start Main
Input Range (low, high)
Initialize loop: $i = low$
Is $i \leq high$?
No
Yes
Call $ISPRIME(i)$
Is Result $= 1$? (Prime)
Yes
No
Increment i
End Main
Function: $ISPRIME(num)$
Is $num < 2$?
Yes
Return 0 (Not
No
Initialize loop: $j = 2$
Is $j \times j \leq num$?
No
Return 1
Yes
Is $num \pmod j = 0$?
Yes
Return
No
Increment j

5.DEVELOP A FUNCTION REVERSE STR THAT ACCEPTS A STRING ARGUMENT. WRITE A C PROGRAM THAT INVOKES THIS FUNCTION TO FIND THE REVERSE OF A GIVEN STRING.

ALGORITHM:

1. Start
2. Declare a character array `str`
3. Prompt the user to enter a string
4. Read the input string
5. Initialize two index variables:
   1. `i = 0`
   2. `j = length of string - 1`
6. Repeat while `i < j`
   a. `Swap characters at positions i` and `j`
   b. `Increment i`
   c. `Decrement j`
7. After the loop, the string is reversed
8. Display the reversed string
9. End

PSUEDOCODE:

```
BEGIN

    DECLARE string str
    PRINT "Enter a string:"
    READ str

    SET i = 0
    SET j = length(str) - 1

    WHILE i < j DO
        temp = str[i]
        str[i] = str[j]
        str[j] = temp
        i = i + 1
        j = j - 1
    END WHILE

    PRINT "Reversed string: ", str

END
```

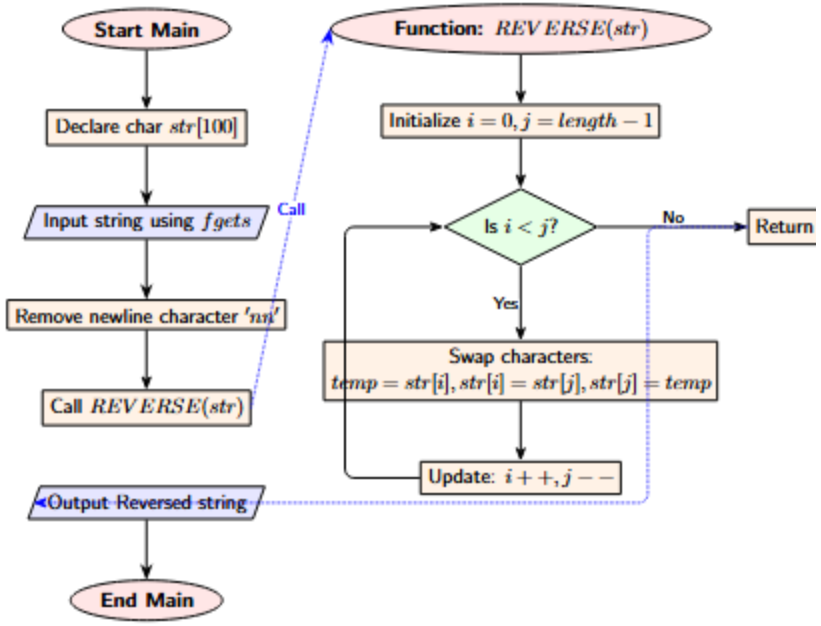
INPUT:

```c
#include <stdio.h>
#include <string.h>
void REVERSE(char str[]) {
    int i, j;
    char temp;
    for (i = 0, j = strlen(str) - 1; i < j; i++, j--) {
        temp = str[i];
        str[i] = str[j];
        str[j] = temp;
    }
}
int main() {
    char str[100];
    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);
    str[strcspn(str, "\n")] = '\0';
    REVERSE(str);
    printf("Reversed string: %s\n", str);
    return 0;
}
```

OUTPUT:

```
Enter a string: SIDDHANT
Reversed string: TNAHDDIS


=== Code Execution Successful ===
```

FLOWCHART:


Start Main
Declare char $str[100]$
Input string using $fgets$
Remove newline character '$\backslash n$'
Call $REVERSE(str)$
Output Reversed string
End Main
Call
Function: $REVERSE(str)$
Initialize $i = 0, j = length - 1$
Is $i < j$?
No
Return
Yes
Swap characters:
$temp = str[i], str[i] = str[j], str[j] = temp$
Update: $i++, j--$

Experiment 7: Structures and Union

- Write a C program that uses functions to perform the following operations:
  a. Reading a complex number.

ALGORITHM:

1. Start
2. Declare a structure `Complex` with two float members: `real` and `imag`.
3. Define a function `readComplex()`
    1. Inside the function:
        1. Declare a variable `c` of type `Complex`.
        2. Ask the user to enter the real part.
        3. Read and store it in `c.real`.
        4. Ask the user to enter the imaginary part.
        5. Read and store it in `c.imag`.
        6. Return structure `c`.
4. In `main()` function:
    1. Declare a variable `c` of type `Complex`.
    2. Display a message “Reading a complex number”.
    3. Call `readComplex()` and store the returned structure in `c`.
    4. Print the complex number in the form **a + bi**.
5. End.

PSUEDOCODE:

```
START

DECLARE structure Complex
    real : float
    imag : float
END structure

FUNCTION readComplex()
    DECLARE c as Complex
    PRINT "Enter real part: "
    INPUT c.real
    PRINT "Enter imaginary part: "
    INPUT c.imag
    RETURN c
END FUNCTION

DECLARE c as Complex

PRINT "Reading a complex number:"
c ← readComplex()

PRINT "You entered: ", c.real, " + ", c.imag, "i"

END
```

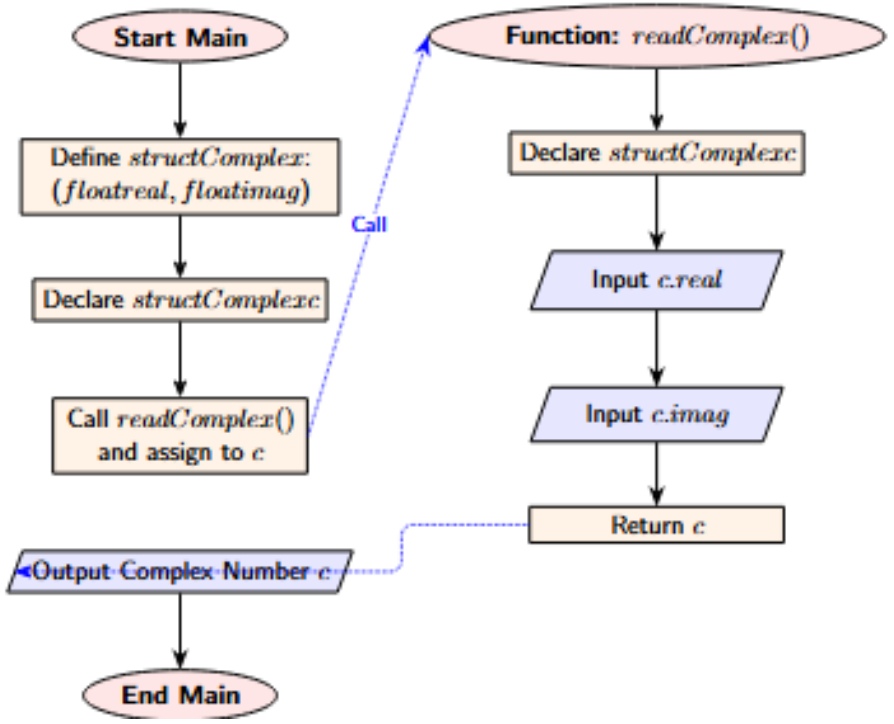
INPUT:

```c
#include <stdio.h>
struct Complex {
    float real;
    float imag;
};
struct Complex readComplex() {
    struct Complex c;
    printf("Enter real part: ");
    scanf("%f", &c.real);
    printf("Enter imaginary part: ");
    scanf("%f", &c.imag);
    return c;
}
int main() {
    struct Complex c;
    printf("Reading a complex number:\n");
    c = readComplex();
    printf("\nYou entered: %.2f + %.2fi\n", c.real, c.imag);
    return 0;
}
```

OUTPUT:

```
Reading a complex number:
Enter real part: 45
Enter imaginary part: : b

You entered: 45.00 + 0.00i


=== Code Execution Successful ===
```

FLOWCHART:


Start Main
Define $structComplex$: ($floatreal$, $floatimag$)
Declare $structComplex c$
Call $readComplex()$ and assign to c
Call
Function: $readComplex()$
Declare $structComplex c$
Input $c.real$
Input $c.imag$
Return c
Output Complex Number c
End Main

b. writing the complex part

Algorithm

1. Start
2. Declare a structure `Complex` with two members: `real` and `imag`
3. In `main()`, declare a variable `c` of type `struct Complex`
4. Ask the user to enter the real part of the complex number
5. Read and store the real part in `c.real`
6. Ask the user to enter the imaginary part of the complex number
7. Read and store the imaginary part in `c.imag`
8. Print the message: "The complex number is:"
9. Call the function `writeComplex(c)`
10. Inside `writeComplex()`
    1. If imaginary part is **positive or zero**, print `real + imag i`
    2. Else print `real - imag i` (imaginary part printed as positive value)
11. End

PSUEDOCODE:

```
STRUCT Complex
    real : float
    imag : float
END STRUCT

FUNCTION writeComplex(c : Complex)
    IF c.imag >= 0 THEN
        PRINT c.real, "+", c.imag, "i"
    ELSE
        PRINT c.real, "-", ABS(c.imag), "i"
    ENDIF
END FUNCTION

BEGIN
    DECLARE c AS Complex

    PRINT "Enter the real part: "
    INPUT c.real

    PRINT "Enter the imaginary part: "
    INPUT c.imag

    PRINT "The complex number is: "
    CALL writeComplex(c)

END
```

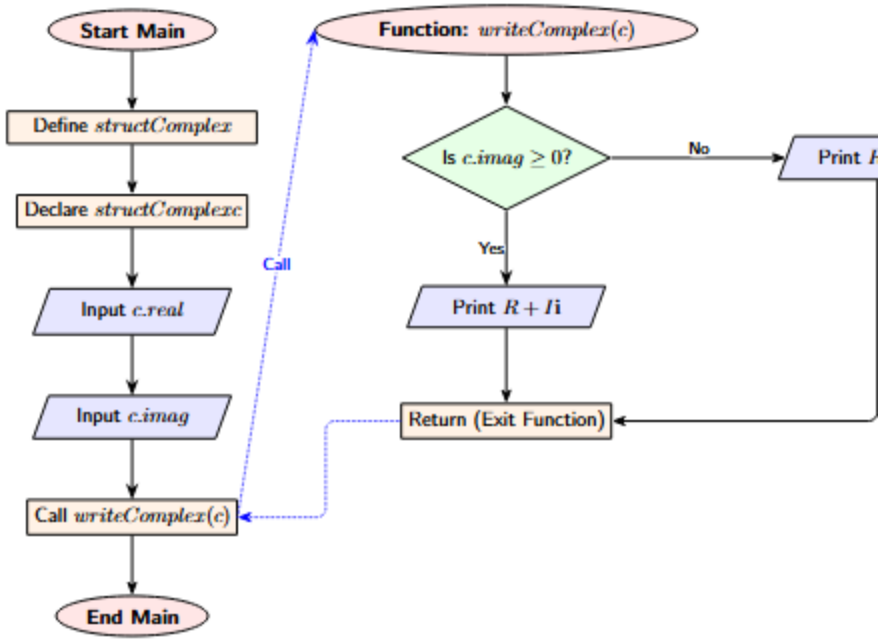
INPUT:

```c
#include <stdio.h>
struct Complex {
    float real;
    float imag;
};
void writeComplex(struct Complex c) {
    if (c.imag >= 0)
        printf("%.2f + %.2fi\n", c.real, c.imag);
    else
        printf("%.2f - %.2fi\n", c.real, -c.imag);
}
int main() {
    struct Complex c;
    printf("Enter the real part: ");
    scanf("%f", &c.real);
    printf("Enter the imaginary part: ");
    scanf("%f", &c.imag);
    printf("\nThe complex number is: ");
    writeComplex(c);
    return 0;
}
```

OUTPUT:

```
Enter the real part: 45
Enter the imaginary part: 6

The complex number is: 45.00 + 6.00i


=== Code Execution Successful ===
```

FLOWCHART:


Start Main
Define structComplex
Declare structComplex c
Input c.real
Input c.imag
Call writeComplex(c)
End Main
Call
Function: writeComplex(c)
Is c.imag ≥ 0?
No
Print
Yes
Print R + Ii
Return (Exit Function)

c. Addition and Subtraction of two complex number

ALGORITHM:

1. **Start**
2. Declare a structure **Complex** with two members:
3. `real`
4. `imag`
5. Define function **add(a, b)** that:
6. Creates a Complex variable `r`
7. Computes `r.real = a.real + b.real`
8. Computes `r.imag = a.imag + b.imag`
9. Creates a Complex variable `r`
10. Computes `r.real = a.real - b.real`
11. Computes `r.imag = a.imag - b.imag`
12. Returns `r`
13. In **main**:
14. y:
15. Sum in form `a + bi`
16. Difference in form `a + bi`
17. **Stop**

PSUEDOCODE:

```
START

DECLARE c1, c2, sum, diff : Complex

PRINT "Enter real and imaginary part of first complex number:"
READ c1.real, c1.imag

PRINT "Enter real and imaginary part of second complex number:"
READ c2.real, c2.imag

sum ← add(c1, c2)
diff ← subtract(c1, c2)

PRINT "Sum = ", sum.real, " + ", sum.imag, "i"
PRINT "Difference = ", diff.real, " + ", diff.imag, "i"

END
```

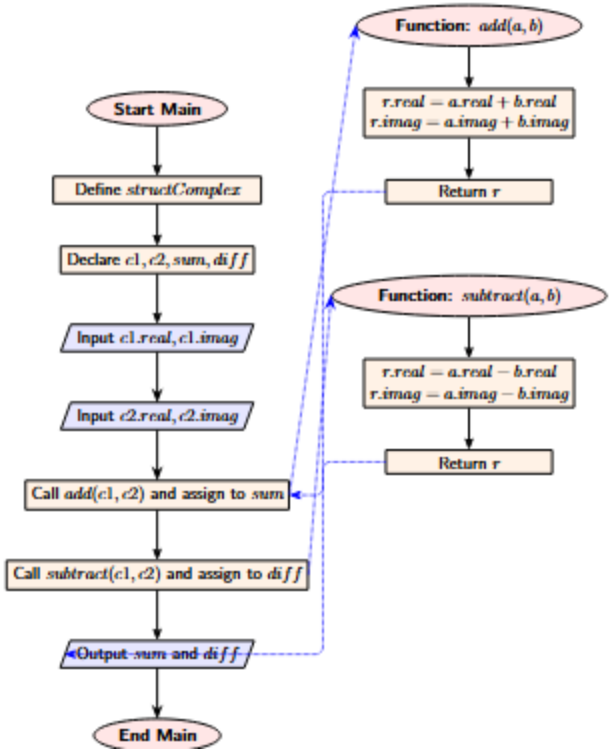
INPUT:

```c
#include <stdio.h>
struct Complex {
    float real;
    float imag;
};
struct Complex add(struct Complex a, struct Complex b) {
    struct Complex r;
    r.real = a.real + b.real;
    r.imag = a.imag + b.imag;
    return r;
}

struct Complex subtract(struct Complex a, struct Complex b) {
    struct Complex r;
    r.real = a.real - b.real;
    r.imag = a.imag - b.imag;
    return r;
}
int main() {
    struct Complex c1, c2, sum, diff;
    printf("Enter real and imaginary part of first complex number: ");
    scanf("%f %f", &c1.real, &c1.imag);
    printf("Enter real and imaginary part of second complex number: ");
    scanf("%f %f", &c2.real, &c2.imag);
    sum = add(c1, c2);
    diff = subtract(c1, c2);
    printf("\nSum = %.2f + %.2fi\n", sum.real, sum.imag);
    printf("Difference = %.2f + %.2fi\n", diff.real, diff.imag);
    return 0;
}
```

OUTPUT:

```
Enter real and imaginary part of first complex number: 5 6
Enter real and imaginary part of second complex number: 6 7

Sum = 11.00 + 13.00i
Difference = -1.00 + -1.00i


=== Code Execution Successful ===
```

FLOWCHART:


Start Main
Define structComplex
Declare c1, c2, sum, diff
Input c1.real, c1.imag
Input c2.real, c2.imag
Call add(c1, c2) and assign to sum
Call subtract(c1, c2) and assign to diff
Output sum and diff
End Main
Function: add(a, b)
r.real = a.real + b.real
r.imag = a.imag + b.imag
Return r
Function: subtract(a, b)
r.real = a.real − b.real
r.imag = a.imag − b.imag
Return r

2. Write a C program to compute the monthly pay of 100 employees using each employee's name, basic pay. The DA is computed as 52% of the basic pay. Gross-salary (basic pay + DA). Print the employees name and gross salary.

- **Algorithm**

- **Step 1:** Start

  Step 2: Create a structure `Employee` with fields: `name`, `basic`, and `gross`.

  Step 3: Declare an array `emp[100]` of type `Employee`.

  Step 4: Display a message to enter employee details.

  Step 5: Repeat steps for `i = 0` to `99`

  a. `Read employee name`

  b. `Read basic salary`

  c. `Calculate DA = 52% of basic pay`

  d. `Compute gross salary = basic + DA`

  e. `Store values in structure array`

  Step 6: Print table heading "Employee Name" and "Gross Salary".

  Step 7: Loop again from `i = 0` to `99`

  `Display each employee's name and gross salary`

  Step 8: Stop

PSUEDOCODE:

```
BEGIN

    DEFINE STRUCT Employee
        name : string
        basic : float
        gross : float
    END STRUCT

    DECLARE emp[100] AS ARRAY OF Employee

    PRINT "Enter details of 100 employees:"

    FOR i ← 0 TO 99 DO
        PRINT "Employee", i+1
        PRINT "Enter name: "
        READ emp[i].name

        PRINT "Enter basic pay: "
        READ emp[i].basic

        DA ← 0.52 * emp[i].basic
        emp[i].gross ← emp[i].basic + DA
    END FOR

    PRINT "----------------------------------------"
    PRINT "Employee Name        Gross Salary"
    PRINT "----------------------------------------"

    FOR i ← 0 TO 99 DO
        PRINT emp[i].name, emp[i].gross
    END FOR

END
```

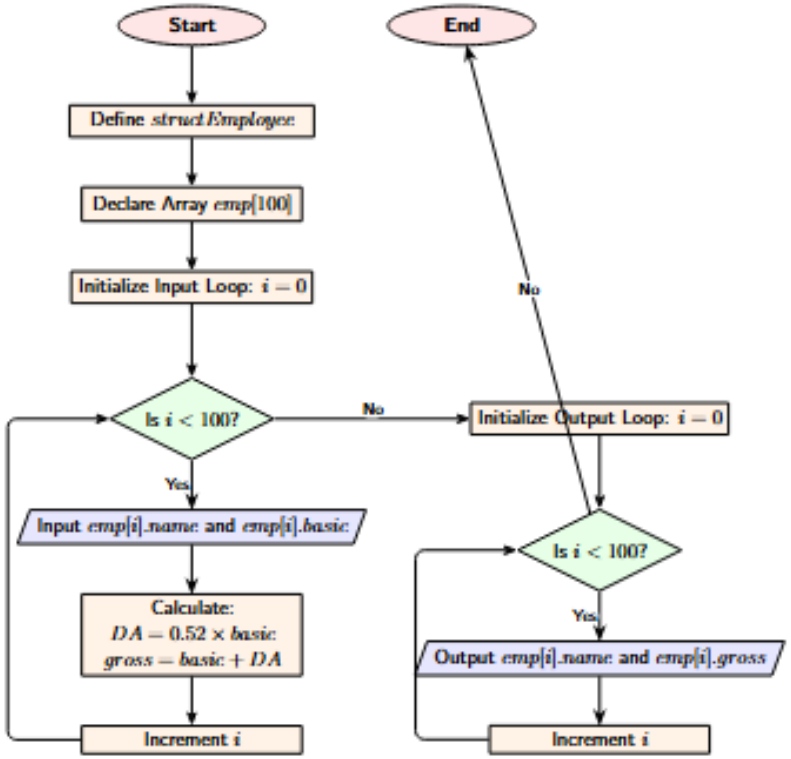
INPUT:

```c
#include <stdio.h>
struct Employee {
    char name[50];
    float basic;
    float gross;
};
int main() {
    struct Employee emp[100];
    printf("Enter details of 100 employees:\n");
    for (int i = 0; i < 100; i++) {
        printf("\nEmployee %d\n", i + 1);
        printf("Enter name: ");
        scanf("%s", emp[i].name);
        printf("Enter basic pay: ");
        scanf("%f", &emp[i].basic);
        float DA = 0.52 * emp[i].basic;
        emp[i].gross = emp[i].basic + DA;
    }
    printf("\n-----------------------------------------\n");
    printf("Employee Name\t\tGross Salary\n");
    printf("-----------------------------------------\n");
    for (int i = 0; i < 100; i++) {
        printf("%-20s %.2f\n", emp[i].name, emp[i].gross);
    }
    return 0;
}
```

OUTPUT:

```
Enter details of 100 employees:

Employee 1
Enter name: A
Enter basic pay: 1

Employee 2
Enter name: B
Enter basic pay: 2

Employee 3
Enter name:
```

FLOWCHART:


Start
Define struct Employee
Declare Array emp[100]
Initialize Input Loop: i = 0
Is i < 100?
No
Yes
Input emp[i].name and emp[i].basic
Calculate:
DA = 0.52 × basic
gross = basic + DA
Increment i
Initialize Output Loop: i = 0
Is i < 100?
No
Yes
Output emp[i].name and emp[i].gross
Increment i
End

3.Create a Book structure containing book_id, title, author name and price. Write a C program to pass a structure as a function argument and print the book details.

Algorithm:

1. Start
2. Declare a structure **Book** with members:
    1. book_id
    2. title
    3. author
    4. price
3. Declare a variable **b** of type structure Book
4. Ask the user to enter book ID
5. Read book ID
6. Ask the user to enter book title
7. Read book title
8. Ask the user to enter author name
9. Read author name
10. Ask the user to enter price
11. Read price
12. Call function `printBook(b)`
13. Inside `printBook(b)`
    1. Print book ID
    2. Print title
    3. Print author
    4. Print price
14. End

PSUEDOCODE:

```
BEGIN

DEFINE STRUCT Book
    INTEGER book_id
    STRING title
    STRING author
    FLOAT price
END STRUCT

FUNCTION printBook(b)
    PRINT "--- Book Details ---"
    PRINT "Book ID :", b.book_id
    PRINT "Title :", b.title
    PRINT "Author :", b.author
    PRINT "Price :", b.price
END FUNCTION

DECLARE b AS Book

PRINT "Enter Book ID: "
READ b.book_id

PRINT "Enter Book Title: "
READ b.title

PRINT "Enter Author Name: "
READ b.author

PRINT "Enter Price: "
READ b.price

CALL printBook(b)

END
```

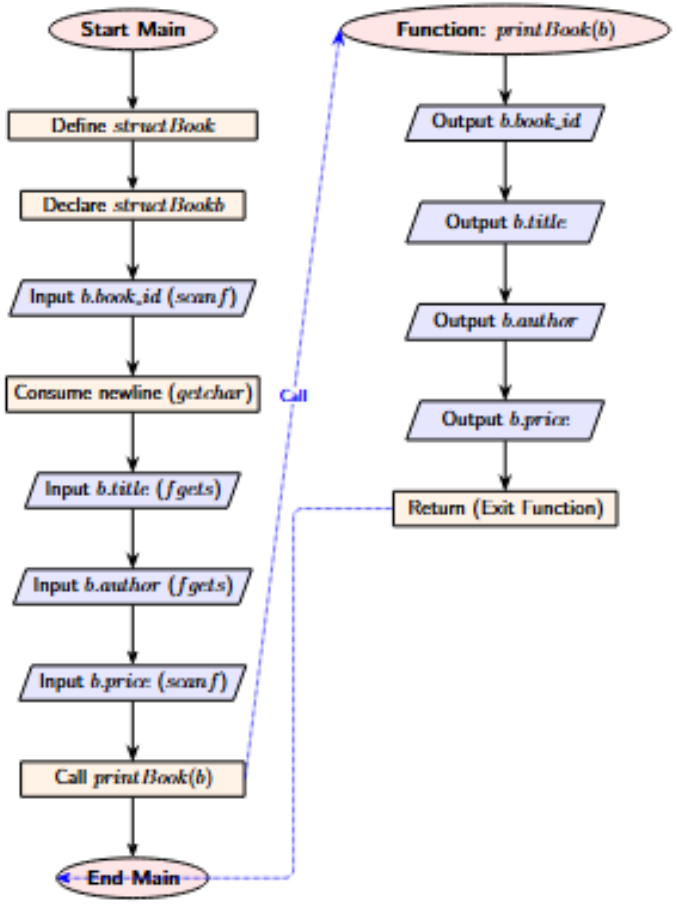
INPUT:

```c
#include <stdio.h>
struct Book {
    int book_id;
    char title[50];
    char author[50];
    float price;
};
void printBook(struct Book b) {
    printf("\n--- Book Details ---\n");
    printf("Book ID   : %d\n", b.book_id);
    printf("Title     : %s\n", b.title);
    printf("Author    : %s\n", b.author);
    printf("Price     : %.2f\n", b.price);
}
int main() {
    struct Book b;
    printf("Enter Book ID: ");
    scanf("%d", &b.book_id);
    getchar();
    printf("Enter Book Title: ");
    fgets(b.title, sizeof(b.title), stdin);
    printf("Enter Author Name: ");
    fgets(b.author, sizeof(b.author), stdin);
    printf("Enter Price: ");
    scanf("%f", &b.price);
    printBook(b);
    return 0;
}
```

OUTPUT:

```
Enter Book ID: 12
Enter Book Title: Jungle book
Enter Author Name: Siddhant
Enter Price: 500

--- Book Details ---
Book ID   : 12
Title     : Jungle book

Author    : Siddhant

Price     : 500.00


=== Code Execution Successful ===
```

FLOWCHART:


Start Main
Define structBook
Declare structBookb
Input b.book_id (scanf)
Consume newline (getchar)
Input b.title (fgets)
Input b.author (fgets)
Input b.price (scanf)
Call printBook(b)
End Main
Call
Function: printBook(b)
Output b.book_id
Output b.title
Output b.author
Output b.price
Return (Exit Function)

4. Create a union containing 6 strings: name, home_address, hostel_address, city, state and zip. Write a C program to display your present address.

Algorithm

1. Start
2. Declare a union named **Address** containing:
    1. name
    2. home_address
    3. hostel_address
    4. city
    5. state
    6. zip
3. Declare a union variable **A**
4. Store the hostel address into **A.hostel_address** using string copy
5. Print **"Present Address:"**
6. Display the value stored in **A.hostel_address**
7. End

PSUEDOCODE:

```
BEGIN

DECLARE UNION Address
    STRING name
    STRING home_address
    STRING hostel_address
    STRING city
    STRING state
    STRING zip
END UNION

DECLARE variable A of type Address

COPY "ABC Hostel, Room 204, Campus Road" INTO A.hostel_address

PRINT "Present Address:"
PRINT A.hostel_address

END
```

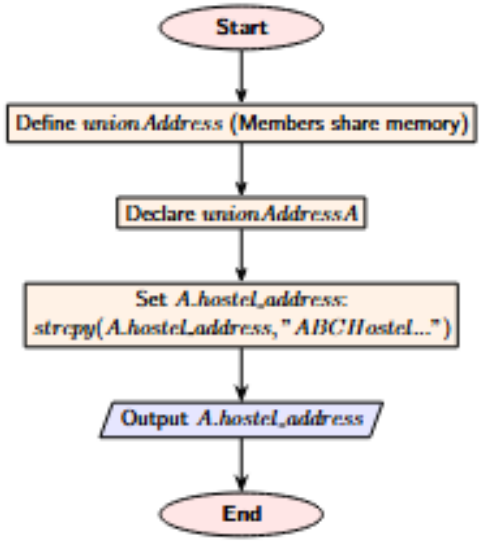
INPUT:

```c
#include <stdio.h>
#include <string.h>
union Address {
    char name[50];
    char home_address[100];
    char hostel_address[100];
    char city[50];
    char state[50];
    char zip[10];
};
int main() {
    union Address A;
    strcpy(A.hostel_address, "ABC Hostel, Room 204, Campus Road");
    printf("Present Address:\n");
    printf("%s\n", A.hostel_address);
    return 0;
}
```

OUTPUT:

```
Present Address:
ABC Hostel, Room 204, Campus Road


=== Code Execution Successful ===
```

FLOWCHART:


Start
Define unionAddress (Members share memory)
Declare unionAddressA
Set A.hostel_address:
strcpy(A.hostel_address,"ABC Hostel...")
Output A.hostel_address
End

Experiment 8: Pointers

1.Declare different types of pointers (int, float, char) and initialize them with the addresses of variables. Print the values of both the pointers and the variables they point to.

Algorithm

1. Start
2. Declare integer variable `a = 10`
3. Declare float variable `b = 12.5`
4. Declare char variable `c = 'X'`
5. Declare pointer `p1` and store the address of `a`
6. Declare pointer `p2` and store the address of `b`
7. Declare pointer `p3` and store the address of `c`
8. Print value of `a`, address in `p1`, and value pointed by `p1`
9. Print value of `b`, address in `p2`, and value pointed by `p2`
10. Print value of `c`, address in `p3`, and value pointed by `p3`
11. End

PSUEDOCODE:

```
BEGIN

    SET a = 10
    SET b = 12.5
    SET c = 'X'

    SET p1 = address of a
    SET p2 = address of b
    SET p3 = address of c

    PRINT "Value of a", a
    PRINT "Address stored in p1", p1
    PRINT "Value pointed by p1", *p1

    PRINT "Value of b", b
    PRINT "Address stored in p2", p2
    PRINT "Value pointed by p2", *p2

    PRINT "Value of c", c
    PRINT "Address stored in p3", p3
    PRINT "Value pointed by p3", *p3

END
```

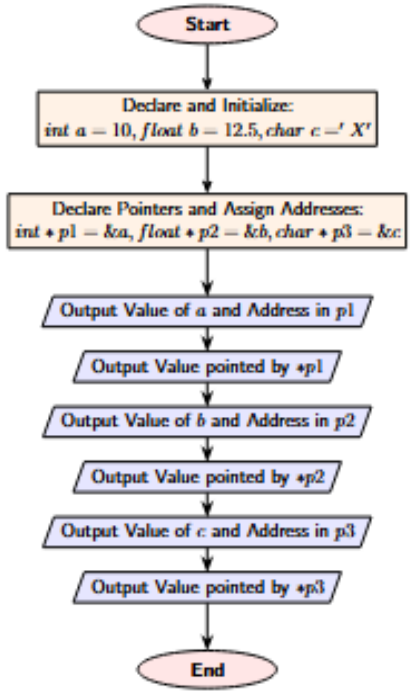
INPUT:

```c
#include <stdio.h>
int main() {
    int a = 10;
    float b = 12.5;
    char c = 'X';
    int *p1 = &a;
    float *p2 = &b;
    char *p3 = &c;
    printf("Value of a = %d\n", a);
    printf("Address stored in int pointer p1 = %p\n", (void*)p1);
    printf("Value pointed by p1 = %d\n\n", *p1);
    printf("Value of b = %.2f\n", b);
    printf("Address stored in float pointer p2 = %p\n", (void*)p2);
    printf("Value pointed by p2 = %.2f\n\n", *p2);
    printf("Value of c = %c\n", c);
    printf("Address stored in char pointer p3 = %p\n", (void*)p3);
    printf("Value pointed by p3 = %c\n", *p3);
    return 0;
}
```

OUTPUT:

```
Value of a = 10
Address stored in int pointer p1 = 0x7ffd761c1f14
Value pointed by p1 = 10

Value of b = 12.50
Address stored in float pointer p2 = 0x7ffd761c1f10
Value pointed by p2 = 12.50

Value of c = X
Address stored in char pointer p3 = 0x7ffd761c1f0f
Value pointed by p3 = X


=== Code Execution Successful ===
```

FLOWCHART:


Start
Declare and Initialize:
int a = 10, float b = 12.5, char c = 'X'
Declare Pointers and Assign Addresses:
int * p1 = &a, float * p2 = &b, char * p3 = &c
Output Value of a and Address in p1
Output Value pointed by *p1
Output Value of b and Address in p2
Output Value pointed by *p2
Output Value of c and Address in p3
Output Value pointed by *p3
End

2.Perform pointer arithmetic (increment and decrement) on pointers of different data types. Observe how the memory addresses change and the effects on data access.

Algorithm

1. Start
2. Declare integer variable `a = 10`
3. Declare float variable `b = 20.5`
4. Declare char variable `c = 'X'`
5. Declare pointers:
    1. `p1` pointing to `a`
    2. `p2` pointing to `b`
    3. `p3` pointing to `c`
6. Print initial pointer addresses (`p1`, `p2`, `p3`)
7. Increment pointers:
    1. `p1++`
    2. `p2++`
    3. `p3++`
8. Print the pointer addresses after increment
9. Decrement pointers:
    1. `p1--`
    2. `p2--`
    3. `p3--`
10. Print pointer addresses after decrement
11. End

PSUEDOCODE:

```
BEGIN

  SET a = 10
  SET b = 20.5
  SET c = 'X'

  SET p1 = address of a
  SET p2 = address of b
  SET p3 = address of c

  PRINT "Initial Addresses"
  PRINT p1, p2, p3

  INCREMENT p1
  INCREMENT p2
  INCREMENT p3

  PRINT "Addresses After Increment"
  PRINT p1, p2, p3

  DECREMENT p1
  DECREMENT p2
  DECREMENT p3

  PRINT "Addresses After Decrement"
  PRINT p1, p2, p3

END
```

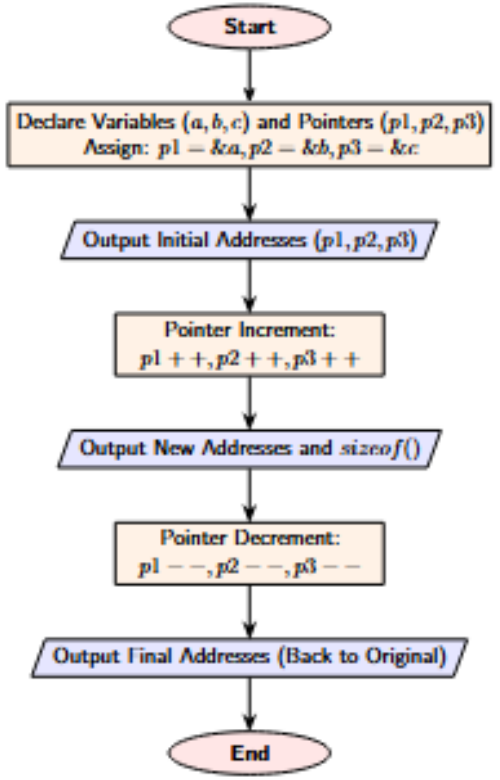
INPUT:

```c
#include <stdio.h>
int main() {
    int a = 10;
    float b = 20.5;
    char c = 'X';
    int *p1 = &a;
    float *p2 = &b;
    char *p3 = &c;
    printf("Initial Addresses:\n");
    printf("p1 (int*)  = %p\n", (void*)p1);
    printf("p2 (float*)= %p\n", (void*)p2);
    printf("p3 (char*) = %p\n\n", (void*)p3);
    p1++;
    p2++;
    p3++;
    printf("Addresses After Increment:\n");
    printf("p1 (int*)  = %p  (int pointer increases by %zu bytes)\n",
           (void*)p1, sizeof(int));
    printf("p2 (float*)= %p  (float pointer increases by %zu bytes)\n",
           (void*)p2, sizeof(float));
    printf("p3 (char*) = %p  (char pointer increases by %zu bytes)\n\n",
           (void*)p3, sizeof(char));
    p1--;
    p2--;
    p3--;
    printf("Addresses After Decrement (back to original):\n");
    printf("p1 (int*)  = %p\n", (void*)p1);
    printf("p2 (float*)= %p\n", (void*)p2);
    printf("p3 (char*) = %p\n", (void*)p3);
    return 0;
```

OUTPUT:

```
Initial Addresses:
p1 (int*)  = 0x7fffcbc79704
p2 (float*)= 0x7fffcbc79700
p3 (char*) = 0x7fffcbc796ff

Addresses After Increment:
p1 (int*)  = 0x7fffcbc79708  (int pointer increases by 4 bytes)
p2 (float*)= 0x7fffcbc79704  (float pointer increases by 4 bytes)
p3 (char*) = 0x7fffcbc79700  (char pointer increases by 1 bytes)

Addresses After Decrement (back to original):
p1 (int*)  = 0x7fffcbc79704
p2 (float*)= 0x7fffcbc79700
p3 (char*) = 0x7fffcbc796ff


=== Code Execution Successful ===
```

FLOWCHART:


Start
Declare Variables (a, b, c) and Pointers $(p1, p2, p3)$
Assign: $p1 = \&a, p2 = \&b, p3 = \&c$
Output Initial Addresses $(p1, p2, p3)$
Pointer Increment:
$p1++, p2++, p3++$
Output New Addresses and $sizeof()$
Pointer Decrement:
$p1--, p2--, p3--$
Output Final Addresses (Back to Original)
End

3.Write a function that accepts pointers as parameters. Pass variables by reference using pointers and modify their values within the function.

ALGORITHM:

1. Start
2. Declare two integer variables: `x` and `y`
3. Ask the user to enter two numbers
4. Read the values of `x` and `y`
5. Call the function `changeValues(&x, &y)`
6. Inside the function:
7. Add 10 to the value pointed by pointer `a`
8. Multiply the value pointed by pointer `b` by 2
9. Return to main
10. Print the updated values of `x` and `y`
11. Stop

PSUEDOCODE:

```
BEGIN

    DECLARE x, y AS INTEGER

    PRINT "Enter two numbers: "
    INPUT x, y

    CALL changeValues(address of x, address of y)

    PRINT "Updated values: ", x, y

END


FUNCTION changeValues(a, b as POINTER TO INTEGER)

    SET value_at(a) = value_at(a) + 10
    SET value_at(b) = value_at(b) * 2

END FUNCTION
```

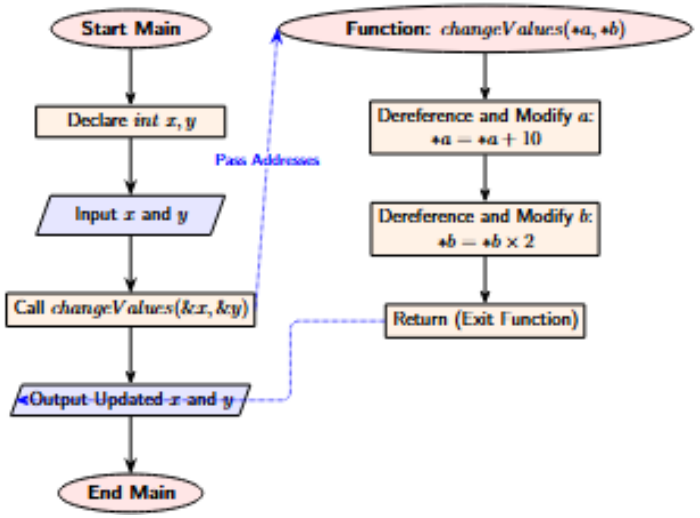
INPUT:

```c
#include <stdio.h>
void changeValues(int *a, int *b) {
    *a = *a + 10;
    *b = *b * 2;
}
int main() {
    int x, y;
    printf("Enter two numbers: ");
    scanf("%d %d", &x, &y);
    changeValues(&x, &y);
    printf("Updated values: %d %d\n", x, y);
    return 0;
}

```

OUTPUT:

```
Enter two numbers: 4 5
Updated values: 14 10


=== Code Execution Successful ===
```

FLOWCHART:


Start Main
Declare $int\ x, y$
Input x and y
Call $changeValues(\&x, \&y)$
Output Updated x and y
End Main
Pass Addresses
Function: $changeValues(*a, *b)$
Dereference and Modify a: $*a = *a + 10$
Dereference and Modify b: $*b = *b \times 2$
Return (Exit Function)

Activity-5:

C program to find the rectangle with the highest perimeter among three rectangles using ternary operator

- ALGORITHM
- **Start**
- **Input** length (`l1`) and breadth (`b1`) of rectangle 1.
- **Input** length (`l2`) and breadth (`b2`) of rectangle 2.
- **Input** length (`l3`) and breadth (`b3`) of rectangle 3.
- **Calculate** perimeter of rectangle 1:
  `p1 = 2 × (l1 + b1)`
- **Calculate** perimeter of rectangle 2:
  `p2 = 2 × (l2 + b2)`
- **Calculate** perimeter of rectangle 3:
  `p3 = 2 × (l3 + b3)`
- **Find maximum perimeter** among `p1`, `p2`, `p3`.
- If `p1` is the largest → `max = p1`
- Else if `p2` is the largest → `max = p2`
- Else → `max = p3`
- **Display** which rectangle has the highest perimeter and its value.
- **Stop**

PSUEDOCODE:

```
BEGIN

    DECLARE l1, b1, l2, b2, l3, b3, p1, p2, p3, max

    PRINT "Enter length and breadth of rectangle 1"
    READ l1, b1

    PRINT "Enter length and breadth of rectangle 2"
    READ l2, b2

    PRINT "Enter length and breadth of rectangle 3"
    READ l3, b3

    p1 ← 2 * (l1 + b1)
    p2 ← 2 * (l2 + b2)
    p3 ← 2 * (l3 + b3)

    IF p1 > p2 THEN
        IF p1 > p3 THEN
            max ← p1
        ELSE
            max ← p3
        ENDIF
    ELSE
        IF p2 > p3 THEN
            max ← p2
        ELSE
            max ← p3
        ENDIF
    ENDIF

    IF max = p1 THEN
        PRINT "Rectangle 1 has the highest perimeter:", p1
    ELSE IF max = p2 THEN
        PRINT "Rectangle 2 has the highest perimeter:", p2
    ELSE
        PRINT "Rectangle 3 has the highest perimeter:", p3
    ENDIF

END
```

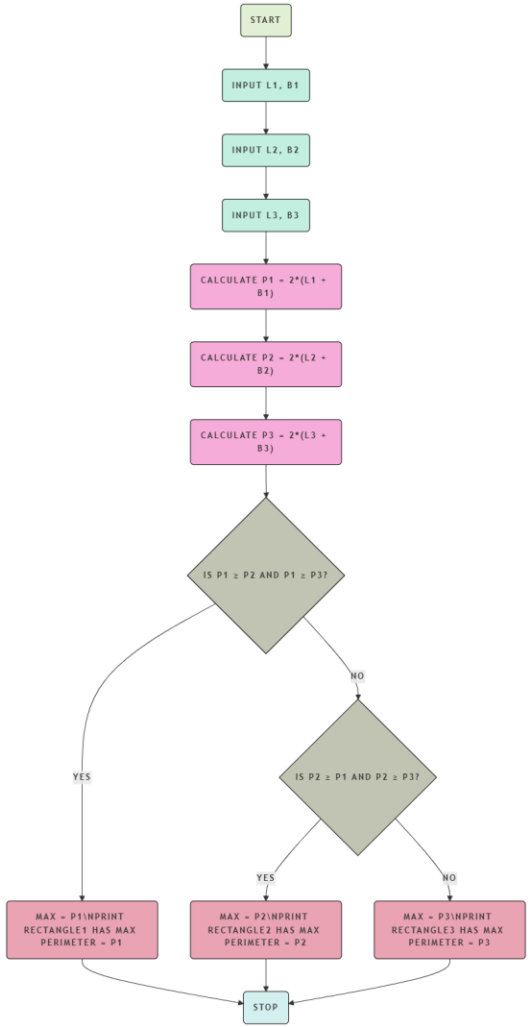
INPUT

```c
#include <stdio.h>
int main() {
    int l1, b1, l2, b2, l3, b3;
    int p1, p2, p3, max;
    printf("Enter length and breadth of rectangle 1: ");
    scanf("%d %d", &l1, &b1);
    printf("Enter length and breadth of rectangle 2: ");
    scanf("%d %d", &l2, &b2);
    printf("Enter length and breadth of rectangle 3: ");
    scanf("%d %d", &l3, &b3);
    p1 = 2 * (l1 + b1);
    p2 = 2 * (l2 + b2);
    p3 = 2 * (l3 + b3);
    max = (p1 > p2) ? ((p1 > p3) ? p1 : p3) : ((p2 > p3) ? p2 : p3);
    if (max == p1)
        printf("Rectangle
    else if (max == p2)
        printf("Rectangle 2 has the highest perimeter: %d\n", p2);
    else
        printf("Rectangle 3 has the highest perimeter: %d\n", p3);
    return 0;
}
```

OUTPUT

```
cd "/Users/nitishkumar/Documents/Code/pracricecode/" && gcc 3.1.5.c -o 3.1.5 && "/Users/nitishkumar/
Documents/Code/pracricecode/"3.1.5
NitishK@Nitishs-MacBook-Air Code % cd "/Users/nitishkumar/Documents/Code/pracricecode/" && gcc 3.1.5
.c -o 3.1.5 && "/Users/nitishkumar/
Documents/Code/pracricecode/"3.1.5
Enter length and breadth of rectangle 1: cd "/Users/nitishkumar/Documents/Code/pracricecode/" && gcc
 3.1.5.c -o 3.1.5 && "/Users/nitishkumar/Documents/Code/pracricecode/"3.1.5
Enter length and breadth of rectangle 2: Enter length and breadth of rectangle 3: Rectangle 3 has th
e highest perimeter: -95068690
NitishK@Nitishs-MacBook-Air pracricecode %
```

FLOWCHART


START
INPUT L1, B1
INPUT L2, B2
INPUT L3, B3
CALCULATE P1 = 2*(L1 + B1)
CALCULATE P2 = 2*(L2 + B2)
CALCULATE P3 = 2*(L3 + B3)
IS P1 ≥ P2 AND P1 ≥ P3?
YES
NO
IS P2 ≥ P1 AND P2 ≥ P3?
YES
NO
MAX = P1\nPRINT RECTANGLE1 HAS MAX PERIMETER = P1
MAX = P2\nPRINT RECTANGLE2 HAS MAX PERIMETER = P2
MAX = P3\nPRINT RECTANGLE3 HAS MAX PERIMETER = P3
STOP

Experimet-3.2
Activity-1:

a program that reads a series of numbers from the user and counts how many of them are positive, negative, and zero. The program should continue to prompt the user for numbers until they choose to stop. After the user decides to stop entering numbers, the program should display the counts of positive numbers, negative numbers, and zeroes entered.

ALGORITHM

Start

Initialize three counters:

`countPositive = 0`

`countNegative = 0`

`countZero = 0`

Repeat the following steps **at least once**:

a. Input a number (`num`).

b. If `num > 0`, increment `countPositive`.

c. Else if `num < 0`, increment `countNegative`.

d. Else, increment `countZero`.

e. Ask user: *"Do you want to enter more numbers? (Y/N)"*.

Continue the loop while the user enters `Y` or `y`.

After loop ends, display the results:

Print number of positive entries.

Print number of negative entries.

Print number of zeros entered.

Stop

PSUEDOCODE:

```
BEGIN

    countPositive ← 0
    countNegative ← 0
    countZero ← 0

    REPEAT
        PRINT "Enter a number: "
        READ num

        IF num > 0 THEN
            countPositive ← countPositive + 1
        ELSE IF num < 0 THEN
            countNegative ← countNegative + 1
        ELSE
            countZero ← countZero + 1
        ENDIF

        PRINT "Do you want to enter more numbers? (Y/N): "
        READ choice

    UNTIL choice ≠ 'Y' AND choice ≠ 'y'

    PRINT "Summary of numbers entered:"
    PRINT "Positive numbers entered: ", countPositive
    PRINT "Negative numbers entered: ", countNegative
    PRINT "Zeroes entered: ", countZero

END
```

INPUT

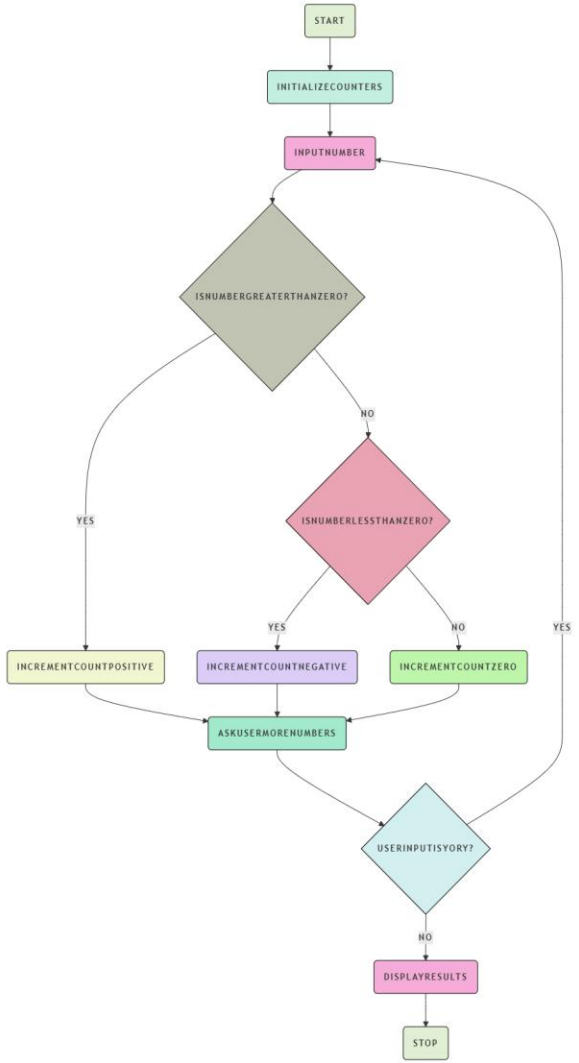
3.2.1.c > main()

```c
#include <stdio.h>
int main() {
    int num, countPositive = 0, countNegative = 0, countZero = 0;
    char choice;
    do {
        printf("Enter a number: ");
        scanf("%d", &num);
        if (num > 0)
            countPositive++;
        else if (num < 0)
            countNegative++;
        else
            countZero++;
        printf("Do you want to enter more numbers? (Y/N): ");
        scanf(" %c", &choice);  // space before %c to consume leftover newline
    } while (choice == 'Y' || choice == 'y');
    printf(" Positive numbers entered: %d", countPositive);
    printf(" Negative numbers entered: %d", countNegative);
    printf(" Zeroes entered: %d", countZero);
    return 0;
}
```

OUTPUT

```
NitishK@Nitishs-MacBook-Air pracricecode % clear
ishkumar/Documents/Code/pracricecode/"3.2.1
Enter a number: 22
Do you want to enter more numbers? (Y/N): 25
 Positive numbers entered: 1 Negative numbers entered: 0 Zeroes entered: 0%
NitishK@Nitishs-MacBook-Air pracricecode % cd "/Users/nitishkumar/Documents/Code/pracricecode/" && gcc 3.2.1.c -o 3.2.1 && "/Users/nit
ishkumar/Documents/Code/pracricecode/"3.2.1
Enter a number: 22
Do you want to enter more numbers? (Y/N): 55
 Positive numbers entered: 1 Negative numbers entered: 0 Zeroes entered: 0%
NitishK@Nitishs-MacBook-Air pracricecode %
```

FLOWCHART


START
INITIALIZECOUNTERS
INPUTNUMBER
ISNUMBERGREATERTHANZERO?
YES
NO
ISNUMBERLESSTHANZERO?
YES
NO
INCREMENTCOUNTPOSITIVE
INCREMENTCOUNTNEGATIVE
INCREMENTCOUNTZERO
ASKUSERMORENUMBERS
USERINPUTISYORY?
YES
NO
DISPLAYRESULTS
STOP

Experimet-3.2
Activity-2:

C program to print multiplication table of a given number

- **ALGORITHM**
- **Start**
- Declare variables `num` (to store the number) and `i` (loop counter).
- Print a message: *"Enter a number:"*
- Read the number from the user and store it in `num`.
- Print a header: *"Multiplication Table of num"*.
- Print a line separator (for formatting).
- Repeat the following steps using a loop from `i = 1` to `10`:

  Calculate `num * i`.

  Print the result in the format: `num * i = result`.
- End loop.
- **Stop**

PSUEDOCODE:

```
START

DECLARE num, i

PRINT "Enter a number: "
READ num

PRINT "Multiplication Table of num"
PRINT "----------------------------"

FOR i = 1 TO 10 DO
    product = num * i
    PRINT num, "*", i, "=", product
END FOR

STOP
```

INPUT

C 3.2.2.c > ...

```c
#include <stdio.h>
int main() {
    int num, i;
    printf("Enter a number: ");
    scanf("%d", &num);
    printf("\nMultiplication Table of %d\n", num);
    printf("--------------------------\n");
    for (i = 1; i <= 10; i++) {
        printf("%d * %d = %d\n", num, i, num * i);
    }
    return 0;
}
```

OUTPUT

```
NitishK@Nitishs-MacBook-Air pracricecode % clear
ishkumar/Documents/Code/pracricecode/"3.2.2
Enter a number: 22

Multiplication Table of 22
--------------------------
22 * 1 = 22
22 * 2 = 44
22 * 3 = 66
22 * 4 = 88
22 * 5 = 110
22 * 6 = 132
22 * 7 = 154
22 * 8 = 176
22 * 9 = 198
22 * 10 = 220
NitishK@Nitishs-MacBook-Air pracricecode %
```

FLOWCHART:


START
DECLARE VARIABLES NUM
PRINTENTERANUMBER
READ NUMBER FROM USER
PRINTMULTIPLICATIONTAB
PRINT LINE SEPARATOR
I FROM 1 TO 10
CALCULATE NUM*I
PRINTNUMXI=RESULT
INCREMENT I
END LOOP AFTER 10
STOP

Experiment-3.2
Activity-3(a)

C program to print multiplication table of a given number

- ALGORITHM:
- **Start**
- Declare variables:
- `num` → to store the number entered by the user.
- `i` → loop counter.
- Display message: *"Enter a number:"*.
- Read the value of `num` from the user.
- Print a header: *"Multiplication Table of num"*.
- Print a line separator (for formatting).
- Initialize loop counter `i = 1`.
- **Repeat until `i <= 10`:**
  a. Calculate `result = num × i`.
  b. Print in format → `num * i = result`.
  c. Increment `i` by 1.
- End loop.
- **Stop**

PSUEDOCODE:

```
BEGIN
    SET rows = 4

    FOR i = 0 TO rows - 1 DO
        SET val = 1

        FOR j = 0 TO i DO
            PRINT val
            val = val * (i - j) / (j + 1)
        END FOR

        PRINT newline
    END FOR
END
```

INPUT

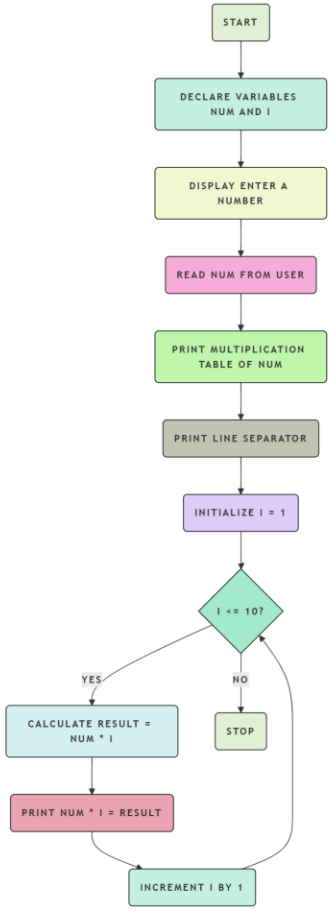
3.2.3.a.c > main()

```c
#include <stdio.h>
int main() {
    int rows = 4;
    for (int i = 0; i < rows; i++) {
        int val = 1;
        for (int j = 0; j <= i; j++) {
            printf("%d ", val);
            val = val * (i - j) / (j + 1); // Calculate next value
        }
        printf("\n");
    }
    return 0;
}
```

OUTPUT

```
NitishK@Nitishs-MacBook-Air pracricecode % cd "/Users/nitishkumar/Documents/Code/pracricecode/" && gcc 4_practice_.c -o 4_practice_ && "/Users/nitishkumar/Documents/Code/pracricecode/"4_practice_
Enter year: 2000
1st January 2000 was Saturday
NitishK@Nitishs-MacBook-Air pracricecode %
```

FLOWCHART


START
DECLARE VARIABLES NUM AND I
DISPLAY ENTER A NUMBER
READ NUM FROM USER
PRINT MULTIPLICATION TABLE OF NUM
PRINT LINE SEPARATOR
INITIALIZE I = 1
I <= 10?
YES
NO
CALCULATE RESULT = NUM * I
STOP
PRINT NUM * I = RESULT
INCREMENT I BY 1

Experiment-3.2
Activity-3(b)

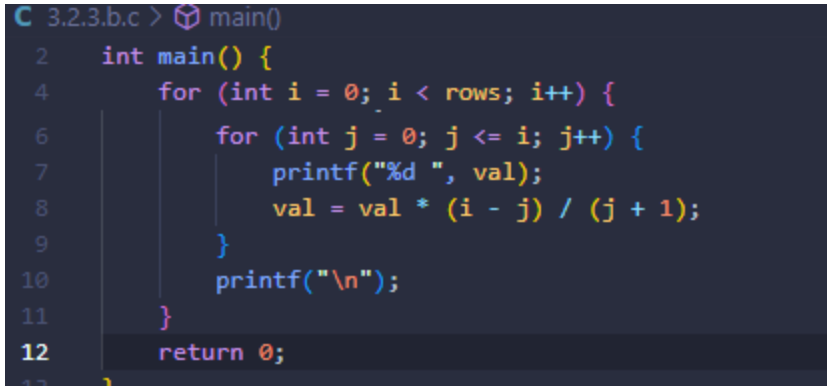
C program to print Pascal's Triangle

- ALGORITHM
- **Start**
- Initialize variable `rows = 4`.
- Set loop variable `i = 0`.
- **Repeat until `i < rows`:**
  a. Set `val = 1`.
  b. Set loop variable `j = 0`.
  c. **Repeat until `j <= i`:**
- Print `val`.
- Update value: `val = val × (i − j) / (j + 1)`.
- Increment `j` by 1.
  d. Print newline (move to next row).
  e. Increment `i` by 1.
- End loop.
- **Stop**

PSUEDOCODE:

```
BEGIN
    SET rows ← 5

    FOR i ← 0 TO rows - 1 DO
        SET val ← 1

        FOR j ← 0 TO i DO
            PRINT val
            val ← val * (i - j) / (j + 1)
        END FOR

        PRINT newline
    END FOR
END
```

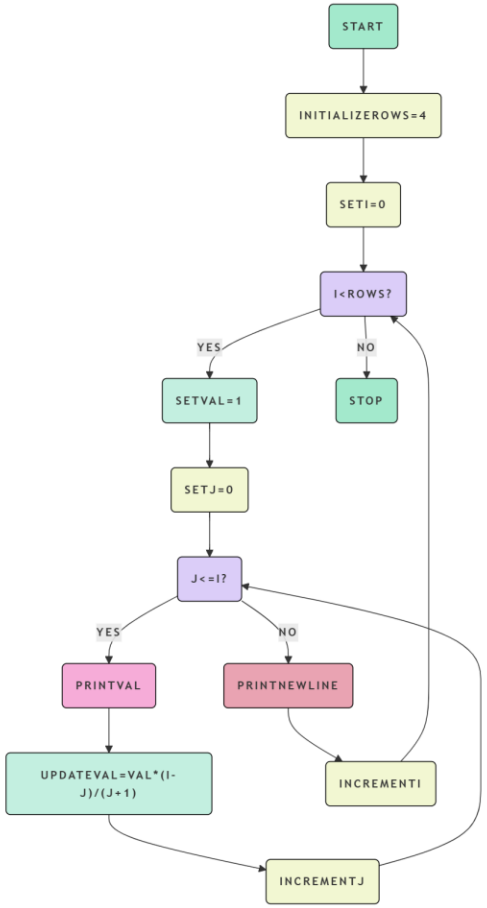
INPUT


3.2.3.b.c > main()
int main() {
for (int i = 0; i < rows; i++) {
for (int j = 0; j <= i; j++) {
printf("%d ", val);
val = val * (i - j) / (j + 1);
}
printf("\n");
}
return 0;

OUTPUT

```
NitishK@Nitishs-MacBook-Air Code % cd "/Users/nitishkumar/Documents/Code/" && gcc 3.2.3.b.c -o 3.2.3.b && "/Users/nitishkumar/Documents/Code/"3.2.3.b
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
NitishK@Nitishs-MacBook-Air Code %
```

FLOWCHART


START
INITIALIZEROWS=4
SETI=0
I<ROWS?
YES
NO
SETVAL=1
STOP
SETJ=0
J<=I?
YES
NO
PRINTVAL
PRINTNEWLINE
UPDATEVAL=VAL*(I-J)/(J+1)
INCREMENTI
INCREMENTJ

Experiment-3.2
Activity-4

Write a program that calculates the population of a town after 10 years, given an initial population of 100,000 and an annual growth rate of 10%. The program should display the population for each year.

- ALGORITHM
- **Start**
- Initialize variables:

  `population = 100000` (initial population)

  `rate = 0.10` (growth rate = 10%)

  `years = 10` (time period in years)
- Print table heading → `"Year Population"`.
- Set loop variable `i = 1`.
- **Repeat until `i <= years`:**

  a. Update population → `population = population × (1 + rate)`.

  b. Print → `Year (i) and Population`.

  c. Increment `i = i + 1`.
- End loop.
- **Stop**

PSUEDOCODE:

```
START

SET population ← 100000
SET rate ← 0.10
SET years ← 10

PRINT "Year    Population"

FOR i ← 1 TO years DO
    population ← population × (i + rate)
    PRINT i, population
END FOR

END
```

INPUT

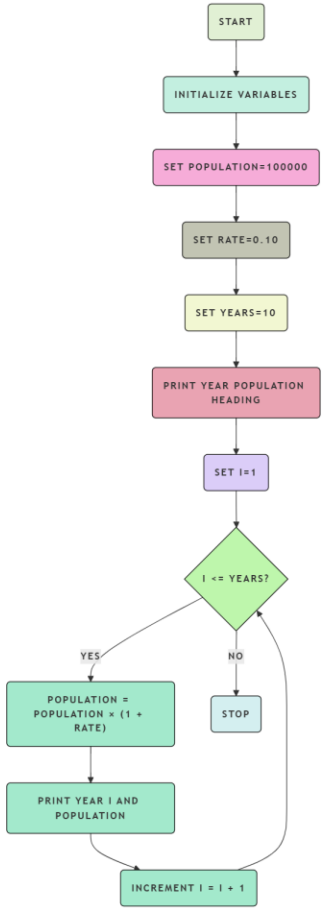
3.2.4.c > ...

```c
#include <stdio.h>
#include <math.h>
int main() {
    double population = 100000;
    double rate = 0.10;
    int years = 10;
    printf("Year\tPopulation");
    for (int i = 1; i <= years; i++) {
        population = population * (1 + rate);
        printf("%d\t%.0f", i, population);
    }
    return 0;
}
```

OUTPUT

```
 ents/Code/"3.2.4
 Year    Population1     1100002 1210003 1331004 1464105 1610516 1771567 1948728 2143599 235795
 10        259374%
NitishK@Nitishs-MacBook-Air Code %
```

FLOWCHART


START
INITIALIZE VARIABLES
SET POPULATION=100000
SET RATE=0.10
SET YEARS=10
PRINT YEAR POPULATION HEADING
SET I=1
I <= YEARS?
YES
NO
POPULATION = POPULATION × (1 + RATE)
STOP
PRINT YEAR I AND POPULATION
INCREMENT I = I + 1

Experiment-3.2
Activity-5

Program to find all numbers that can be expressed as the sum of two cubes in two different ways

- ALGORITHM
- **Start**
- Declare variable `limit`.
- Print → *"Enter the limit:"*.
- Read value of `limit` from the user.
- **For** each integer `a` from 1 to `limit^(1/3)` →
  a. **For** each integer `b` from `a` to `limit^(1/3)` →
  i. **For** each integer `c` from `a+1` to `limit^(1/3)` →
  - **For** each integer `d` from `c` to `limit^(1/3)` →
  * Compute `sum1 = a³ + b³`.
  * Compute `sum2 = c³ + d³`.
  * If `(sum1 == sum2) AND (sum1 <= limit)` →
  - Print: `sum1 = a³ + b³ = c³ + d³`.
- End all loops.
- **Stop**

PSUEDOCODE:

```
BEGIN
    DECLARE limit, a, b, c, d, sum1, sum2

    PRINT "Enter the limit: "
    READ limit

    FOR a = 1 TO a^3 <= limit
        FOR b = a TO b^3 <= limit
            sum1 = a^3 + b^3

            FOR c = a+1 TO c^3 <= limit
                FOR d = c TO d^3 <= limit
                    sum2 = c^3 + d^3

                    IF sum1 = sum2 AND sum1 <= limit THEN
                        PRINT sum1, a, b, c, d
                    ENDIF
                ENDFOR
            ENDFOR
        ENDFOR
    ENDFOR

END
```

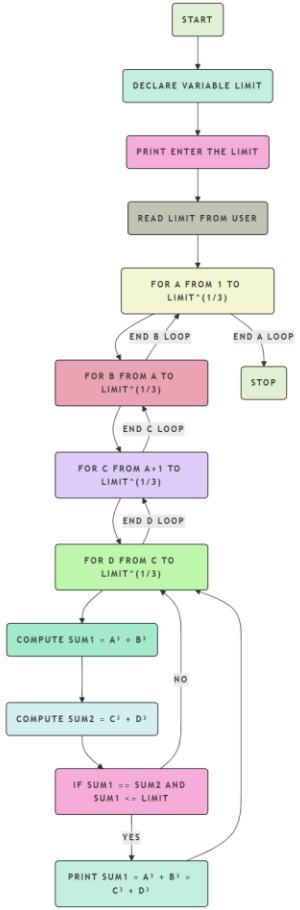
INPUT

```c
#include <stdio.h>
int main() {
    int limit;
    printf("Enter the limit: ");
    scanf("%d", &limit);
    for (int a = 1; a * a * a <= limit; a++) {
        for (int b = a; b * b * b <= limit; b++) {
            for (int c = a + 1; c * c * c <= limit; c++) {
                for (int d = c; d * d * d <= limit; d++) {
                    int sum1 = a*a*a + b*b*b;
                    int sum2 = c*c*c + d*d*d;
                    if (sum1 == sum2 && sum1 <= limit) {
                        printf("%d = %d^3 + %d^3 = %d^3 + %d^3",
                               sum1, a, b, c, d);
                    }
                }
            }
        }
    }
  return 0;
}

```

OUTPUT

```
NitishK@Nitishs-MacBook-Air Code % clear
ents/Code/"3.2.5
Enter the limit: 44
NitishK@Nitishs-MacBook-Air Code % cd "/Users/nitishkumar/Documents/Code/" && gcc 3.2.5.c -o 3.2.5 && "/Users/nitishkumar/Docum
ents/Code/"3.2.5
Enter the limit: 33cd "/Users/nitishkumar/Documents/Code/" && gcc 3.2.5.c -o 3.2.5 && "/Users/nitishkumar/Documents/Code/"3.2.5
NitishK@Nitishs-MacBook-Air Code % 22cd "/Users/nitishkumar/Documents/Code/" && gcc 3.2.5.c -o 3.2.5 && "/Users/nitishkumar/Doc
uments/Code/"3.2.5
zsh: command not found: 22cd
NitishK@Nitishs-MacBook-Air Code %
```

FLOWCHART


START
DECLARE VARIABLE LIMIT
PRINT ENTER THE LIMIT
READ LIMIT FROM USER
FOR A FROM 1 TO LIMIT^(1/3)
END B LOOP
END A LOOP
STOP
FOR B FROM A TO LIMIT^(1/3)
END C LOOP
FOR C FROM A+1 TO LIMIT^(1/3)
END D LOOP
FOR D FROM C TO LIMIT^(1/3)
COMPUTE SUM1 = A³ + B³
NO
COMPUTE SUM2 = C³ + D³
IF SUM1 == SUM2 AND SUM1 <= LIMIT
YES
PRINT SUM1 = A³ + B³ = C³ + D³