ವಿಶ್ವೇಶ್ವರಯ್ಯ ತಾಂತ್ರಿಕ ವಿಶ್ವವಿದ್ಯಾಲಯ
ವಿಟಿಯು ಅಧಿನಿಯಮ ೧೯೯೪ರ ಅಡಿಯಲ್ಲಿ ಕರ್ನಾಟಕ ಸರ್ಕಾರದಿಂದ ಸ್ಥಾಪಿತವಾದ ರಾಜ್ಯವಿಶ್ವವಿದ್ಯಾಲಯ
VISVESVARAYA TECHNOLOGICAL UNIVERSITY
State University of Government of Karnataka Established as per the VTU Act, 1994"JnanaSangama" Belagavi-590018, Karnataka, India

Prof. B. E. Rangaswamy, Ph.D
REGISTRAR

Phone: (0831) 2498100
Fax: (0831) 2405467

REF: VTU/BGM/Online courses/827/2025-26/3327 DATE: 30 SEP 2025

CIRCULAR

Subject: Reference Lab Manual for C Programming Laboratory (1BPOPL107/207)- reg."
Reference: BoS chairperson's email dated: 27.09.2025

The **C Programming Lab Manual** for courses **1BPOPL107/207** has been prepared by the Chairman and Members of the Board of Studies (CSE Stream), VTU, Belagavi.

All affiliated colleges' faculty who are handling these laboratory sessions are hereby instructed to **use this Lab Manual as a reference document** while conducting the C Programming Laboratory. This reference manual is intended to bring **uniformity in the conduct of laboratory sessions, evaluation, and learning outcomes** across all institutions. Colleges may adopt suitable enhancements as required by local laboratory setup, but the core structure and exercises outlined **in this manual should be followed**. The soft copy of the Lab Manual is made available for reference @ https://vtu.ac.in/en/b-e-scheme-syllabus/ under the heading of Programme Specific Courses Laboratory.

All the principals of engineering colleges under the ambit of the university are hereby informed to update the content of the circular to the notice of all concerned.

30/09/25
REGISTRAR

To:

The Principals of all Engineering Colleges under the ambit of the university
The Chairpersons/Programme Coordinators of University Departments at Kalaburagi, Bengaluru, Mysuru, and Belagavi.

Copy to:

- The Hon'ble Vice-Chancellor, through the Secretary to the Vice-Chancellor, for kind information.
- The Dean, Faculty of Engineering, VTU Belagavi, for information.
- The Registrar (Evaluation) for information
- The Director ITI, SMU, VTU Belagavi, for information and needful, and also make arrangements to upload the circular on the VTU web portal.
- The Special Officer, CoE, Mysuru, for information and needful
- The Special Officer QPDS, Examination Section, VTU Belagavi
- Office file

30092025

Visvesvaraya Technological University

C Programming Lab
(1BPOPL107/207)

Lab manual prepared by
Chairman and Members
Board of Studies (CSE Stream)

30092025

	display: "Keyword " found in the course description." Otherwise, display: "Keyword " not found in the course description."	
7.	Develop a C program that takes marks for three subjects as input. Use a function to check if the student has passed (minimum 40 marks in each subject). Display the average and whether the student passed or failed.	25
8.	In an ATM system, two account balances need to be swapped temporarily for validation. Develop a C program that accepts two balances and uses a function with pointers to swap them. Display the balances before and after swapping.	29

PART – B		
SL No.	**TYPICAL OPEN-ENDED EXPERIMENTS** Open-ended experiments are a type of laboratory activity where the outcome is not predetermined, and students are given the freedom to explore, design, and conduct the experiment based on the problem statements as per the concepts defined by the course coordinator. It encourages creativity, critical thinking, and inquiry-based learning.	**Page No.**
1.	A college library has a digital bookshelf system where each book is assigned a unique Book ID. The bookshelf is organized in ascending order of Book IDs. Develop a C Program to quickly find whether a book with a specific Book ID is available in the shelf.	33
2.	A sports teacher has recorded the scores of students in a 100-meter race. To prepare the result sheet, the teacher wants the scores arranged in descending order (from highest to lowest). Develop a C program to sort the scores.	39
3.	A small warehouse tracks how many units of different products are shipped from multiple branches. Another dataset shows how much revenue each product generates per unit. Develop a C program which combines these datasets to calculate the total revenue generated by each branch.	42

4.	A basic mobile contact manager stores first and last names separately. For displaying full names in the contact list, you need to join them manually. Additionally, the system must check the length of each full name to ensure it fits the screen. Perform these operations by developing a C program without using built in string functions.	43
5.	A currency exchange booth allows users to convert between two currencies. Before confirming the exchange, the system simulates a swap of the values to preview the result without actually changing the original data. In other cases, it updates the actual values. Develop a C program that implements both behaviours using Call by Value and Call by reference	47
6.	A local library needs to store and display details of its books, including title, author, and year of publication. Design a structure that can hold these details and develop a C program to display a list of all books entered.	50

PART-A

1. A robot needs to find how far it must travel between two points on a 2D plane. Develop a C program to calculate the straight-line distance between the given coordinates.

Problem Description

You have two points on a 2D plane, each defined by coordinates (x1, y1) and (x2, y2) The task is to calculate the straight-line (Euclidean) distance between these two points.

Input

The program will take four floating-point numbers as input, representing the coordinates of the two points:

- x_1, y_1: Coordinates of the first point.
- x_2, y_2: Coordinates of the second point.

Output

The program should output the distance between the two points, typically rounded or formatted to a reasonable number of decimal places.

Constraints

- Coordinates can be any valid floating-point numbers.
- The output distance should be non-negative.
- You can assume valid input (no need to validate input format).

Method

To calculate the distance between two points (x1, y1) (x2,y2), use the Euclidean distance formula:

For two points $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$, the distance d between them is:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Illustration

Example 1:

Input: Input Coordinates (x1, y1) (x2, y2)= (0, 0) (3, 4)

Output: 5.00

Illustration

Example 2: (1, 1) (4, 5)

Input: Input Coordinates (x1, y1) (x2, y2)= ((1, 1) (4, 5)

Output: 5.0

Algorithm

Step 1: Start

Step 2: Input the coordinates of the first point (x1, y1)

Step 3: Input the coordinates of the second point (x2, y2)

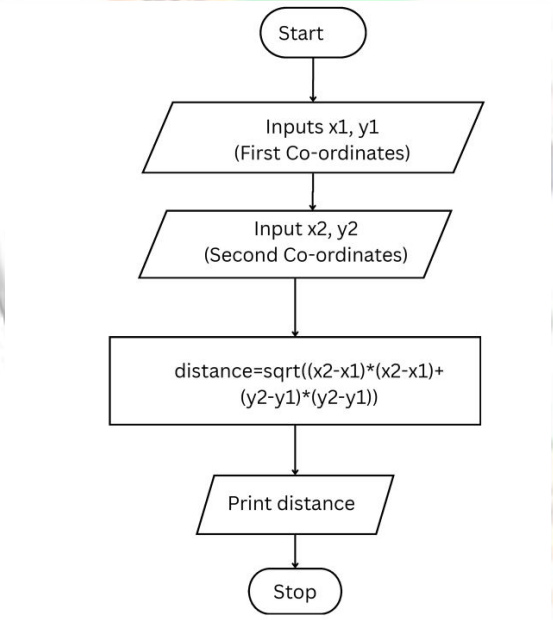
Step 4: Compute the distance

distance = sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1))

Step 8: Display the distance

Step 9: Stop

Flowchart:



Program Code

```
#include <stdio.h>
#include <math.h>
void main()
 {
  float x1, y1, x2, y2, distance;
  printf("Enter x1 and y1 (coordinates of the first point): ");
  scanf("%f %f", &x1, &y1);
  printf("Enter x2 and y2 (coordinates of the second point): ");
  scanf("%f %f", &x2, &y2);
  distance = sqrt((x2 - x1)*(x2 - x1) + (y2 - y1)*(y2 - y1));
  printf("The straight-line distance between the two points is: %.2f\n", distance);
}
```

Test Case	Input Coordinates (x1, y1) (x2, y2)	Expected Output
1	(0, 0) (3, 4)	5.00
2	(1, 1) (4, 5)	5.00
3	(-2, -3) (-4, -7)	4.47
4	(10, 10) (10, 10)	0.00

2. Develop a C program that takes a student's marks as input and displays their grade based on the following criteria: 90 and above: Grade A 75 to 89: Grade B 60 to 74: Grade C 50 to 59: Grade D Below 50: Grade F Choose a suitable control structure to implement this logic efficiently.

Problem Description

Write a C program that takes a student's marks as input and displays the corresponding grade based on the following criteria:

- 90 and above: Grade A
- 75 to 89: Grade B
- 60 to 74: Grade C
- 50 to 59: Grade D
- Below 50: Grade F

Input

- A single integer number representing the student's marks.

Output

- Display the grade corresponding to the marks.

Constraints

- Marks will be in the range 0 to 100.
- You can assume valid input (no need for error handling if marks are out of range).

Method

- Use conditional statements (if-else or switch won't be suitable here since ranges are involved).
- if-else ladder is the most straightforward and efficient approach here.

Illustration

Example 1:

Input: Marks=95

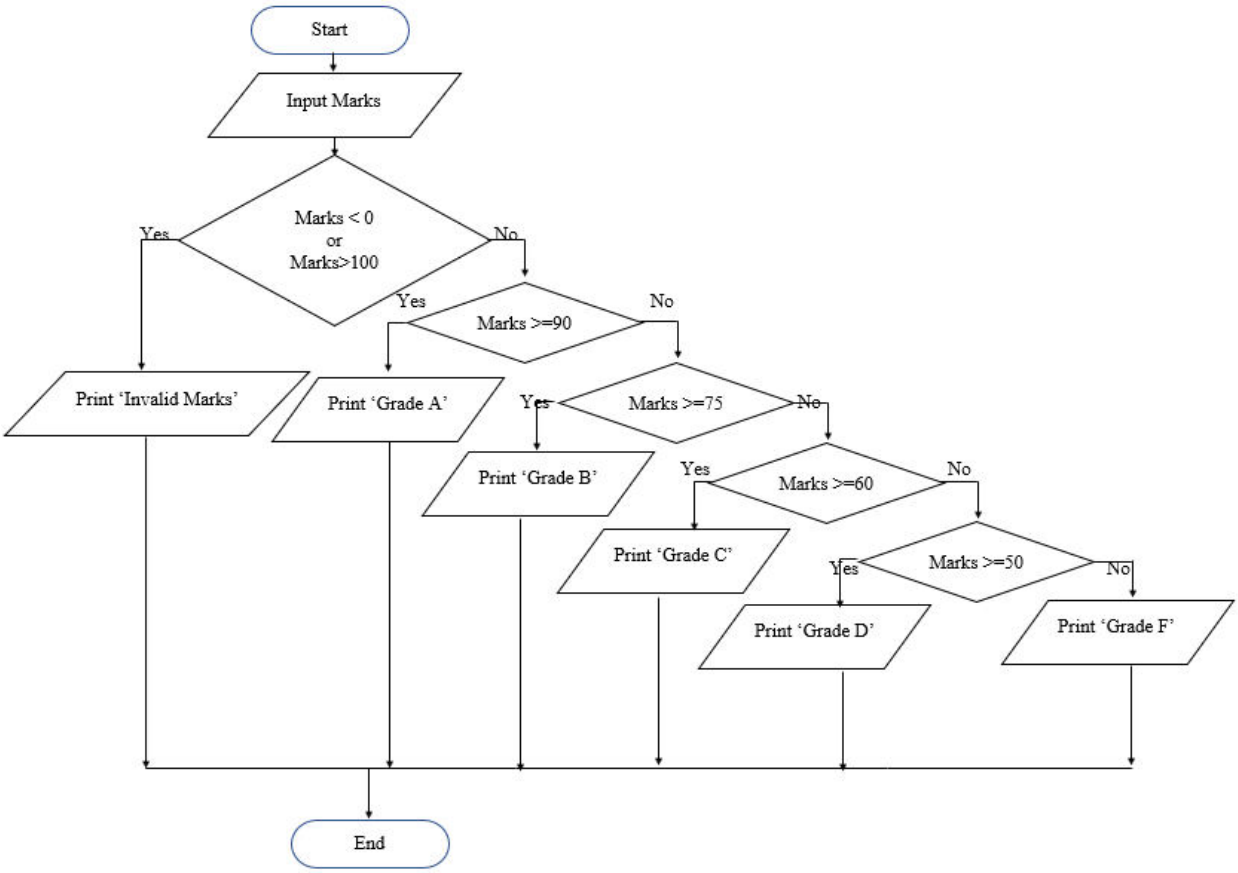
Output: Grade=A

Example 2:

Input: Marks=45

Output: Grade=F

Flowchart:



Algorithm

Step 1: Start

Step 2: Input the student's marks

Step 3: Use if-else if statements to determine the grade:

- If marks ≥ 90 → Grade A
- Else if marks ≥ 75 → Grade B
- Else if marks ≥ 60 → Grade C
- Else if marks ≥ 50 → Grade D
- Else → Grade F

Step 4: Display the grade

Step 5: Stop

Program Code

```
#include <stdio.h>
void main()
 {
   int marks;
   printf("Enter the student's marks (0 to 100): ");
   scanf("%d", &marks);
   if (marks < 0 || marks > 100)
       printf("Invalid marks! Please enter a value between 0 and 100.\n");
   else if (marks >= 90)
      printf("Grade: A\n");
  else if (marks >= 75)
      printf("Grade: B\n");
   else if (marks >= 60)
      printf("Grade: C\n");
   else if (marks >= 50)
      printf("Grade: D\n");
  else
      printf("Grade: F\n");
}
```

Test Case	Input Marks	Expected Grade
1	95	A
2	80	B
3	65	C
4	55	D
5	45	F
6	74	C
7	75	B
8	89	B
9	90	A

3. Develop a C program that takes a unique identification input like PAN Number, AADHAR_Number, APAAR_Id, Driving License, Passport and checks it against a set of stored KYC records. Based on the input, display whether the individual is verified or not. Use an appropriate control structure to handle multiple possible ID matches. Assume all Unique identification are of integer type.

Problem Description

You have a set of stored KYC records identified by different unique IDs such as PAN Number, AADHAR Number, APAAR ID, Driving License, Passport. The program should take an integer input representing one of these IDs and check if it matches any stored record. If yes, print "Verified", else "Not Verified".

Input

- An integer input representing a unique identification number (like PAN, AADHAR, etc).

Output

- Display **"Verified"** if the ID matches any stored record.
- Display **"Not Verified"** if the ID doesn't match any record.

Constraints

- Assume IDs(choices) are unique integer
- Use switch case

Method

- Use a suitable control structure to compare the input against multiple stored IDs.
- Since there are multiple IDs to check using switch statement in simplified way gives suitable output

---

Illustration

Example 1:

Input: 1

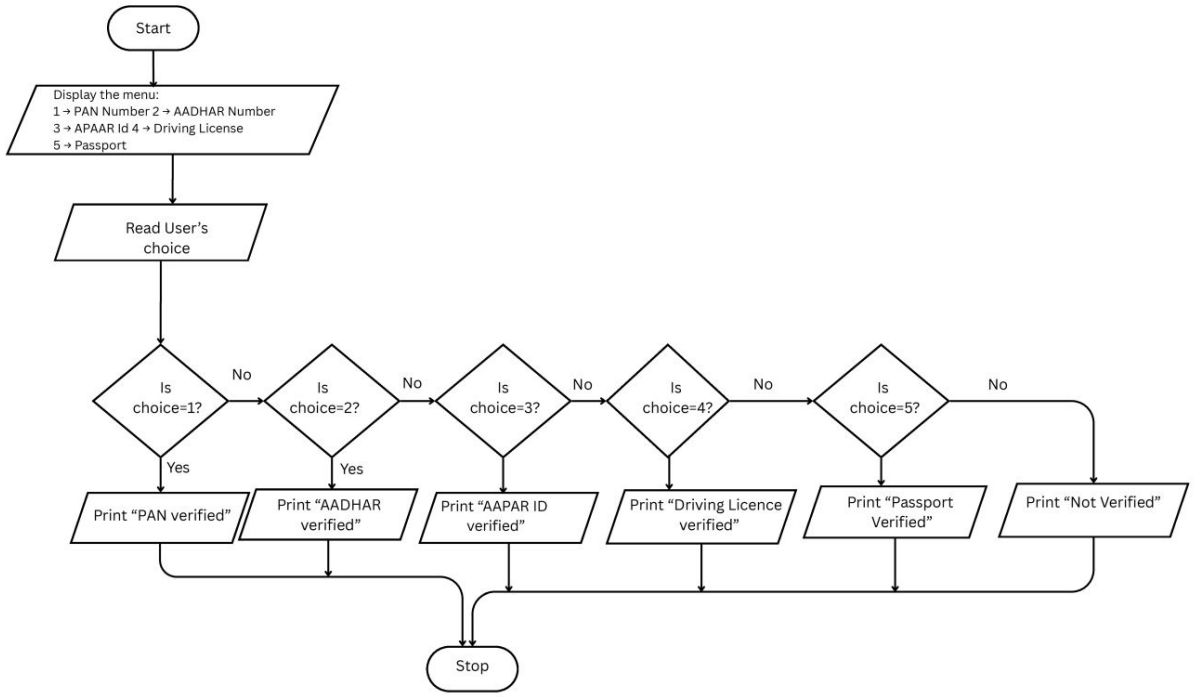
Output: PAN verified

Example 2:

Input: 7

Output: Not Verified

Flowchart:



Algorithm

Step 1: Start

Step 2: Display the menu:

1→ PAN Number

2 → AADHAR Number

3 → APAAR Id

4 → Driving License

5 → Passport

Step 3: Read the user's choice.

Step 4: Use a switch(choice) to process the selected ID type:

- Case 1 (PAN):

  Print "PAN Verified".

- Case 2 (AADHAR):

Print "AADHAR Verified".

- Case 3 (APAAR):

  Print "APAAR Verified".

- Case 4 (Driving License):

  Print "Driving License Verified".

- Case 5 (Passport):

  Print "Passport Verified".

- Default: Print "Not verified".

Step 5: Stop

Program Code

```
#include <stdio.h>
int main()
{
   int choice, id;
   printf("------ KYC Verification System ------\n");
   printf("1. PAN Number\n");
   printf("2. AADHAR Number\n");
   printf("3. APAAR Id\n");
   printf("4. Driving License\n");
   printf("5. Passport\n");
   printf("Enter your choice (1-5): ");
   scanf("%d", &choice);
   switch(choice)
   {
      case 1:   // PAN
            printf("PAN Verified!\n");
            break;
      case 2:   // AADHAAR
            printf("AADHAR Verified!\n");
            break;
      case 3:   // APAAR
            printf("APAAR Verified!\n");
           break;
```

```
        case 4:   // Driving License
                printf("Driving License Verified!\n");
                break;
        case 5:   // Passport
                printf("Passport Verified!\n");
                break;
        default:
           printf("Not Verified!\n");
    }
    return 0;
}
```

Input Choice	Expected Output	Explanation
1	PAN Verified	Choice 1 for PAN
2	AADHAR Verified	Choice 2 for AADHAR ID
3	APAAR Verified	Choice 3 for APAAR ID
4	Driving License Verified	Choice 4 for Driving License ID
5	Passport Verified	Choice 5 for Passport ID
6	Not verified.	Choice other than 1 to 5, doesn't match any ID

4. A math app needs to determine the type of roots for a quadratic equation based on user input. Develop a C program to calculate and display the roots based on the given coefficients.

Problem Description

A math learning app needs to determine the **nature of the roots** of a quadratic equation of the form:

The program takes coefficients a, b, and c of a quadratic equation:

$ax^2+bx+c=0$

It calculates the roots and determines the type of roots based on the discriminant:

- If a=0 and b=0: Invalid co-efficients
- If a=0(a alone zero): Equation is linear, roots are equal

$\Delta=b^2-4ac$

- If $\Delta>0$: Two distinct real roots.
- If $\Delta=0$: One real root (repeated).
- If $\Delta<0$: Two complex roots.

Input

- Three floating-point numbers representing the coefficients a, b, and c.

Output

- Display the roots.
- Display the type of roots (real and distinct, real and equal, or complex).

Constraints

- a=0 and b=0: Invalid co-efficients
- a=0(a alone zero): Equation is linear
- a≠0 (not a linear equation).
- Coefficients can be positive or negative real numbers and complex number

---

Method

- Calculate the discriminant.
- Use conditional statements to check the discriminant.
- Use sqrt for square roots.

- For complex roots, display real and imaginary parts separately.

Illustration

Example 1:

Input: a,b,c: 1, -3, 2

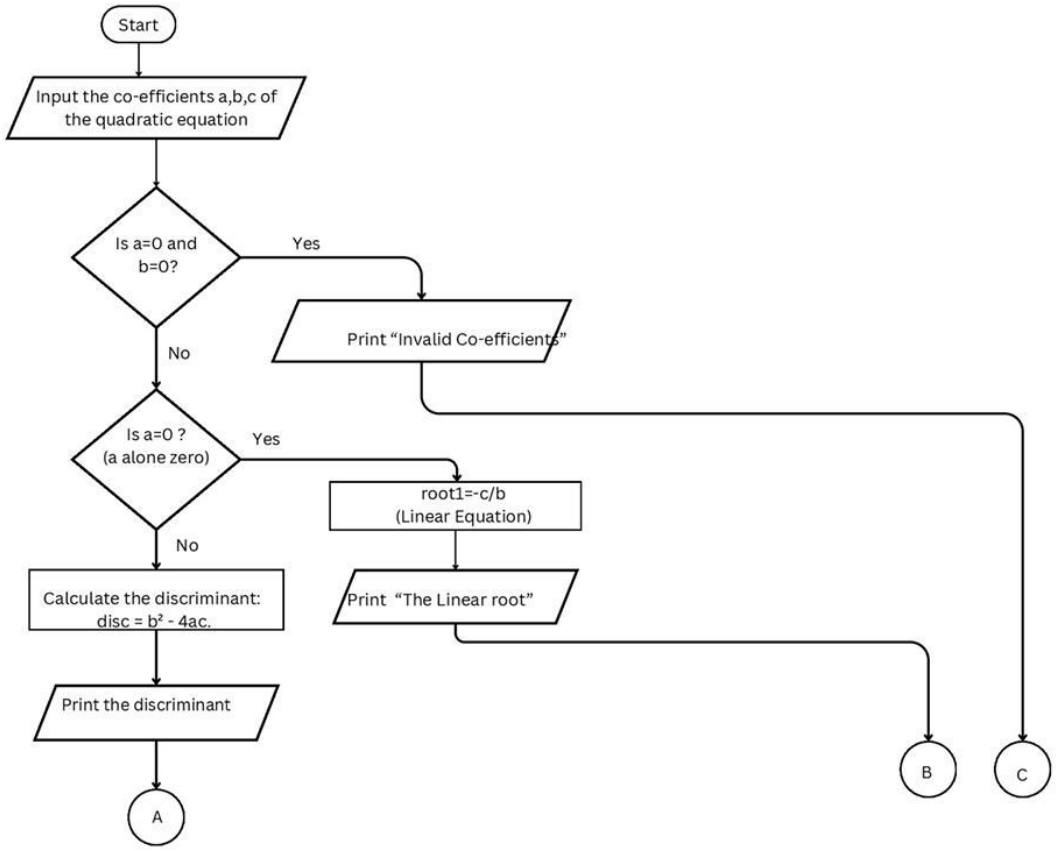
Output: Two distinct Roots are: 2.00, 1.00

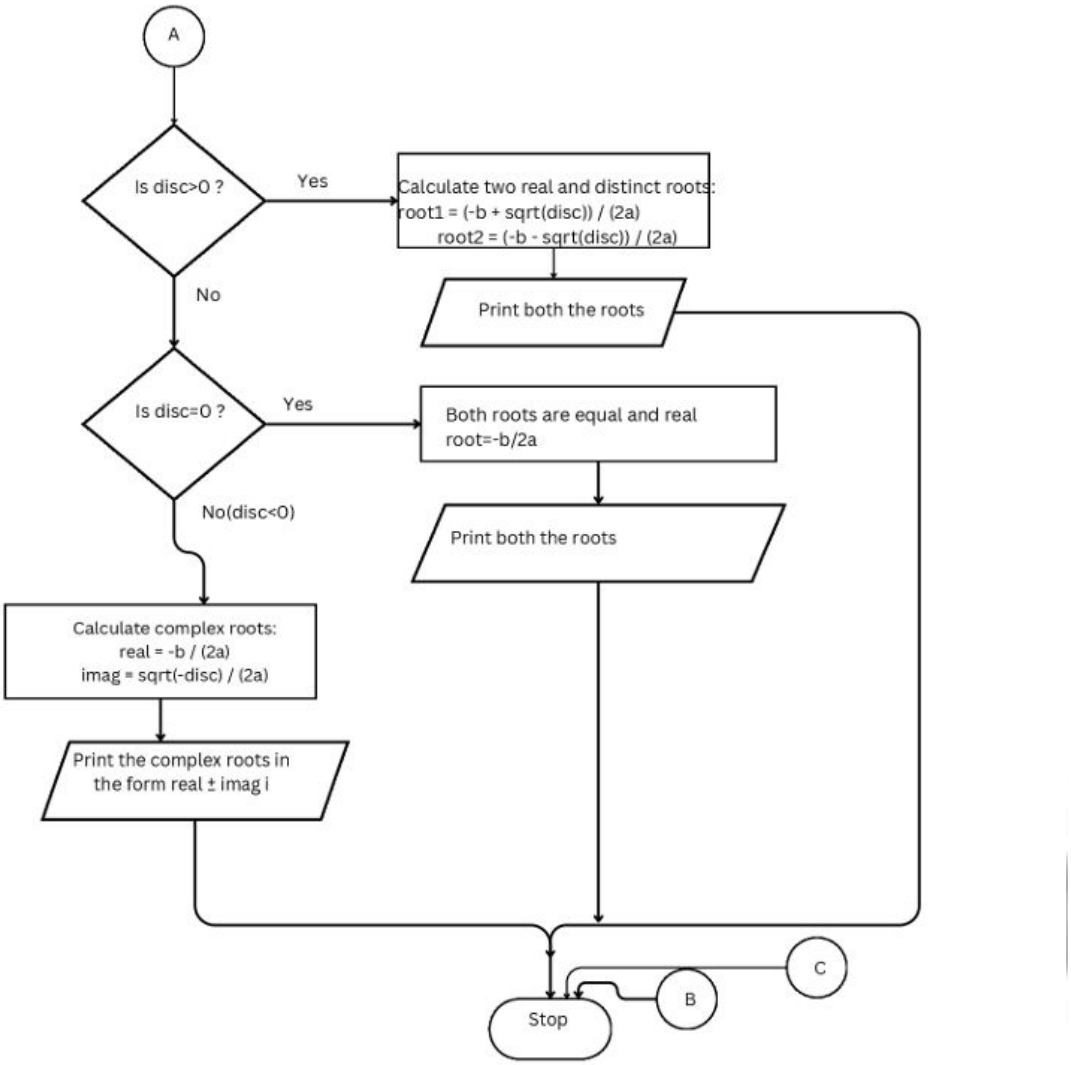
Example 2:

Input: a,b,c: 0,0,1

Output: Invalid co-efficients

Flowchart:





Algorithm:

Step1: Start

Step2: Input the coefficients a, b, and c of the quadratic equation.

Step3: Check if a == 0 and b == 0:

If true, **print** "Invalid Coefficients" and terminate the program.

Step4: Else if a == 0:

- The equation is linear (bx + c = 0).
- Calculate root = -c / b.
- **Print** the linear root.

Step5: Else (a ≠ 0):

- The equation is quadratic.
- Calculate the discriminant: disc = b^2 - 4ac.
- **Print** the discriminant.

- **If disc > 0:**
  - Calculate two real and distinct roots:
    - root1 = (-b + sqrt(disc)) / (2a)
    - root2 = (-b - sqrt(disc)) / (2a)
  - **Print** both roots.
- **Else if disc == 0:**
  - Calculate one real and repeated root:
    - root = -b / (2a)
  - **Print** the root.
- **Else (disc < 0):**
  - Calculate complex roots:
    - real = -b / (2a)
    - imag = sqrt(-disc) / (2a)
  - **Print** the complex roots in the form real ± imag i.

Step6: Stop

Program Code

```
#include <stdio.h>
#include <math.h>
int main()
 {
   float a, b, c, root1, root2, disc, real, imag;
   printf("Enter coefficients of quadratic equation (a, b, c): ");
   scanf("%f %f %f", &a, &b, &c);
   if (a == 0 && b == 0)
  {
     printf("Invalid Coefficients!\n");
   }
   else if (a == 0)
   {
     // Linear equation bx + c = 0
     root1 = -c / b;
     printf("Linear Equation with root: %.2f\n", root1);
    }
```

```
        Else
        {
          // Quadratic equation
          disc = b * b - 4 * a * c;
          printf("Discriminant = %.2f\n", disc);
          if (disc > 0)
          {
             root1 = (-b + sqrt(disc)) / (2 * a);
             root2 = (-b - sqrt(disc)) / (2 * a);
             printf("Two distinct real roots: %.2f and %.2f\n", root1, root2);
          }
          else if (disc == 0)
          {
             root1 = -b / (2 * a);
             printf("One real root: %.2f\n", root1);
          }
          else
          {
             real = -b / (2 * a);
             imag = sqrt(-disc) / (2 * a);
             printf("Complex roots: %.2f + %.2fi and %.2f - %.2fi\n", real, imag, real, imag);
          }
      }
    return 0;
}
```

Test Case	Input (a, b, c)	Expected Output	Explanation
1	1, -3, 2	Roots are real and distinct: 2.00, 1.00	D=9−8=1>0D = 9 - 8 = 1 > 0
2	1, -2, 1	Roots are real and equal: 1.00	D=4−4=0D = 4 - 4 = 0
3	1, 2, 5	Roots are complex: -1.00 ± 2.00i	D=4−20=−16<0D = 4 - 20 = -16 < 0
4	0, 2, 1	Linear Equation (a cannot be 0)	a = 0, root=-1/2=-0.5
5	2, 5, 3	Roots are real and distinct: -1.00, -1.50	D=25−24=1>0D = 25 - 24 = 1 > 0
6	0,0,1	Invalid Co-efficients	A=0 and b=0, so C=0

5. A sensor in a robotic arm needs to calculate the angle of rotation in real-time, but the hardware doesn't support built-in trigonometric functions. Develop a C program to approximate the value of sin(x) using a series expansion method for improved performance.

Problem Description

A sensor in a robotic arm must continuously compute the angle of rotation during movement. However, due to hardware limitations, built-in trigonometric functions (like sin()) are not available. To resolve this, a C program must be developed to approximate sin(x) using a series expansion, such as the Taylor Series.

This allows the robotic system to compute sine values efficiently and with reasonable accuracy using basic arithmetic operations only.

Input

A single real number x (in **degrees**) representing the angle for which sin(x) needs to be calculated.

Output

A real number representing the approximated value of sin(x), computed using the Taylor Series expansion.

Constraints

- $-100 \leq x \leq 100$
- Input angle in degree
- Accuracy should be acceptable for most robotics applications(error tolerance≤0.001
- Avoid using math library functions like sin( ), cos( ), or pow().

Method

- Use Taylor series expansion of sin(x)\sin(x)sin(x) around 0:
- $$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$
- The series alternates signs and involves odd powers of xxx divided by factorial of the power.
- We'll calculate terms iteratively until the desired precision or a fixed number of terms.

- Use a loop to sum terms.

Illustration

Example 1:

Input: x=30°, 5 terms

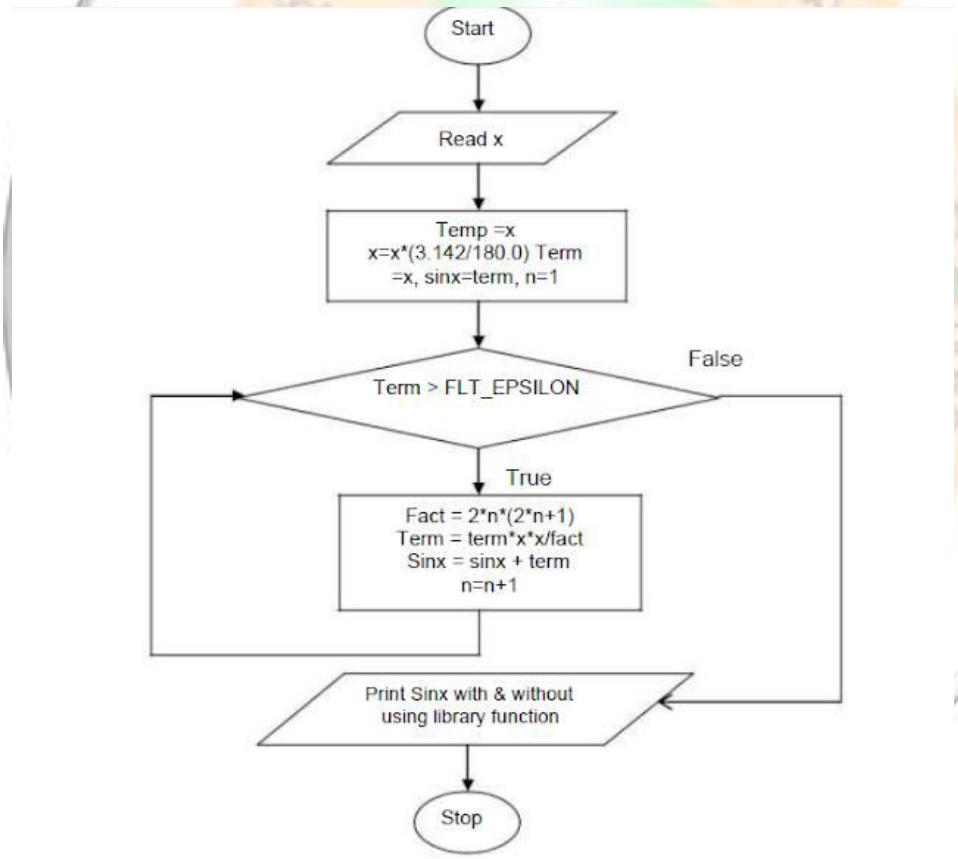
Expected Output: sin(x) using Tailor's series expansion ~0.5 (sin(30°) = 0.5)

Example 2:

Input: x=45°, 7 terms

Expected Output: sin(x) using Tailor's series expansion ~0.7071 (sin(45°) ≈ 0.7071)

Flowchart:



Algorithm

Step 1: Start

Step 2: Input angle x in degrees

Step 3: Convert degrees to radians:

$$x\_rad = x * \pi / 180$$

Step 4: Initialize variables:

- term = x_rad
- sum = 0
- sign = 1

Step 5: Loop through n terms for the desired accuracy

- Add current term * sign to sum
- Compute next term using formula:

term = term * x_rad * x_rad / ((2*i) * (2*i+1))

- Flip sign for next term (sign = -sign)

Step 6: Print final sum as approximate sin(x)

Step 7: Stop

Program Code

```
#include <stdio.h>
#include<math.h>
#define PI  3.14
void main()
{
   float sum,term,x,nume;
   int deg;
    i=2,fact=1;
   printf("Enter angle in degrees: ");
   scanf("%d", &deg);
   x=(deg*PI)/180;
   sum=x;
  nume=x;
  do
   {
     fact=fact*i*(i+1);
     nume = -nume * x * x ;
     term = nume/fact;
     sum += term;
     i+=2;
  } while (fabs(term)>=0.0001);
```

```
    printf("Approximated sin(%d) = %.2f\n", deg, sum);
}
```

Test Case	Input (Degrees, Terms)	Expected Output (approx)	Explanation
1	30°, 5 terms	~0.5	sin(30°) = 0.5
2	45°, 7 terms	~0.7071	sin(45°) ≈ 0.7071
3	90°, 10 terms	~1.0	sin(90°) = 1
4	0°, 5 terms	0.0	sin(0°) = 0
5	180°, 10 terms	~0.0 (close to zero, slight floating error)	sin(180°) ≈ 0

6. Develop a C program that accepts a course description string and a keyword from the user. Search whether the keyword exists within the course description using appropriate string functions. If found, display: "Keyword '' found in the course description." Otherwise, display: "Keyword '' not found in the course description.

Problem Description

Develop a C program that accepts a course description (string) and a keyword (string) from the user. The program should search the keyword inside the course description. If the keyword exists, print a confirmation message; otherwise, print that it is not found.

Input

- A string representing the course description (can be multiple words).
- A string representing the keyword to search for.

Output

- If keyword is found in the course description:
  Keyword '<keyword>' found in the course description.
- Otherwise:
  Keyword '<keyword>' not found in the course description.

Constraints

- Keyword search should be case-sensitive (or case-insensitive if you want — specify).
- Assume input strings are reasonable in length (e.g., less than 256 characters).
- Use standard C string functions.

Method

- Use fgets or scanf to read strings.
- Use the strstr() function to check if keyword is a substring of the course description.
- Display appropriate messages based on the result.

Illustration

Example 1:

Input: Course Description: Introduction to Computer Science and Programming

Keyword: Programming

Expected Output: Keyword 'Programming' found in the course description.

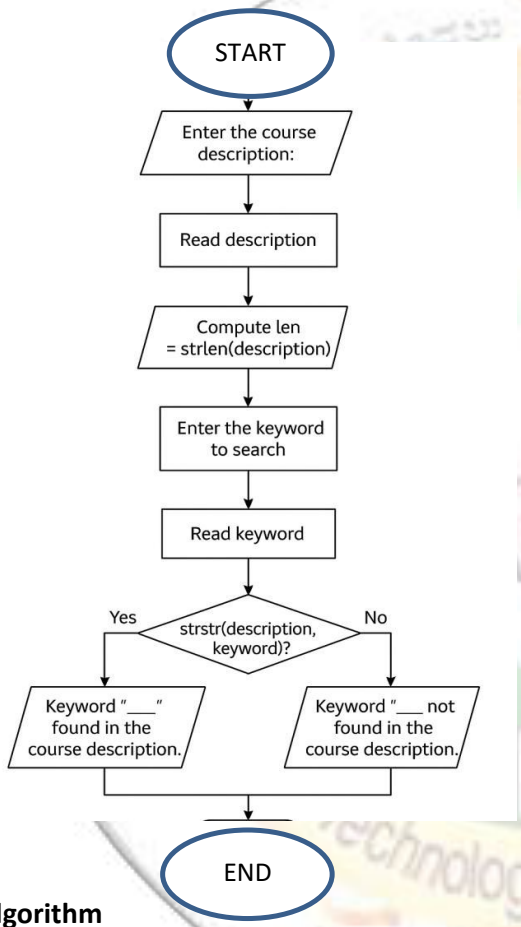
Example 2:

Input: Course Description: Fundamentals of Networking and Security

Keyword: Net

Expected Output: Keyword 'Net' found in the course description.

Flowchart:



Algorithm

Step 1: Start

Step 2: Input the course description (string)

Step 3: Input the keyword to search

Step 4: Use the strstr() function to check if the keyword exists in the course description

- If strstr() returns a non-NULL pointer → Keyword found
- Else → Keyword not found

Step 5: Display appropriate message

Step 6: Stop

Program Code

```
#include <stdio.h>
#include <string.h>
#define MAX 100
void main()
 {
   char description[MAX];
   char keyword[100];
   int len;
   printf("Enter the course description:\n");
   scanf("%s",description);
   len = strlen(description);
   printf("Enter the keyword to search: ");
   scanf("%s", keyword);
   if (strstr(description, keyword) )
    printf("Keyword '%s' found in the %s course description.\n",
keyword,description);
   else
      printf("Keyword '%s' not found in the %s course description.\n",
keyword,description);

   }
```

Test Case	Course Description	Keyword	Expected Output
1	Introduction to Computer Science and Programming	Programming	Keyword 'Programming' found in the course description.
2	Fundamentals of Networking and Security	Net	Keyword 'Net' found in the course description.

3	Advanced Mathematics and Its Applications	Math	Keyword 'Math' found in the course description.

7. Develop a C program that takes marks for three subjects as input. Use a function to check if the student has passed (minimum 40 marks in each subject). Display the average and whether the student passed or failed.

Problem Description

Develop a C program that takes marks for three subjects as input, checks if the student has passed (minimum 40 marks in each subject), and then displays the average marks and the pass/fail status.

Input

- Three integers representing marks in three subjects.

Output

- Display the average marks.
- Display whether the student **Passed** or **Failed** based on the criteria (all subjects have marks >= 40).

Constraints

- Marks should be in the range 0 to 100.
- Passing requires at least 40 marks in **each** subject.

Method

- Use a function hasPassed() that accepts the three marks and returns a boolean indicating pass/fail.
- Calculate average marks in the main program.
- Display the results accordingly.

Illustration

Example 1:

Input: Marks: 50, 60, 70

Expected Average: 60

Expected Output: Passed

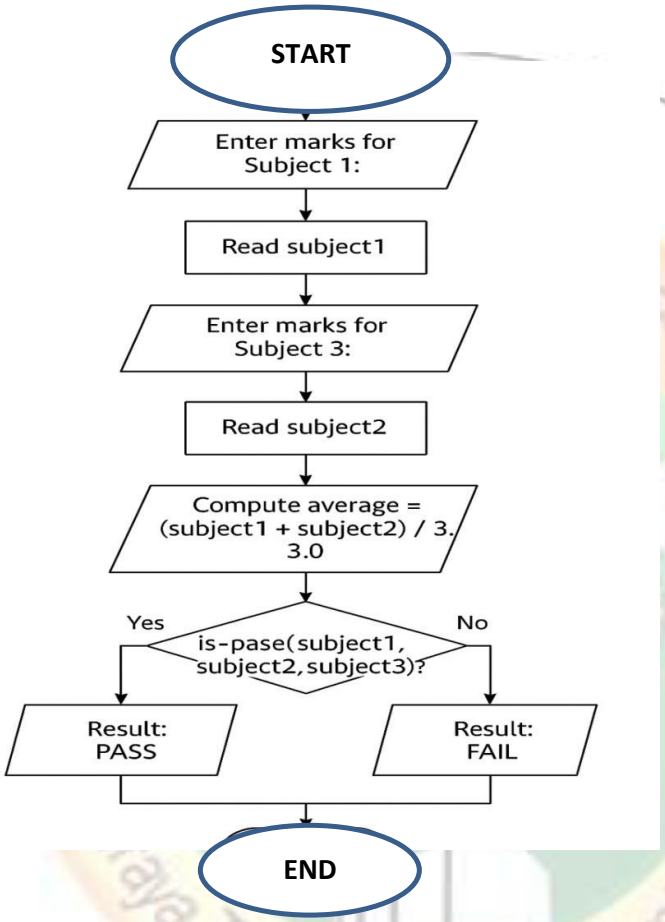
Example 2:

Input: Marks: 40, 39, 50

Expected Average: 43

Expected Output: Failed

Flowchart



Algorithm

Step 1: Start

Step 2: Input marks for three subjects: mark1, mark2, mark3

Step 3: Calculate the average:

average = (mark1 + mark2 + mark3) / 3.0

Step 4: Call a function isPassed(mark1, mark2, mark3) that returns:

- 1 if all marks ≥ 40
- 0 otherwise

Step 5: Display the average

Step 6: If return value is 1 → Print "Passed"

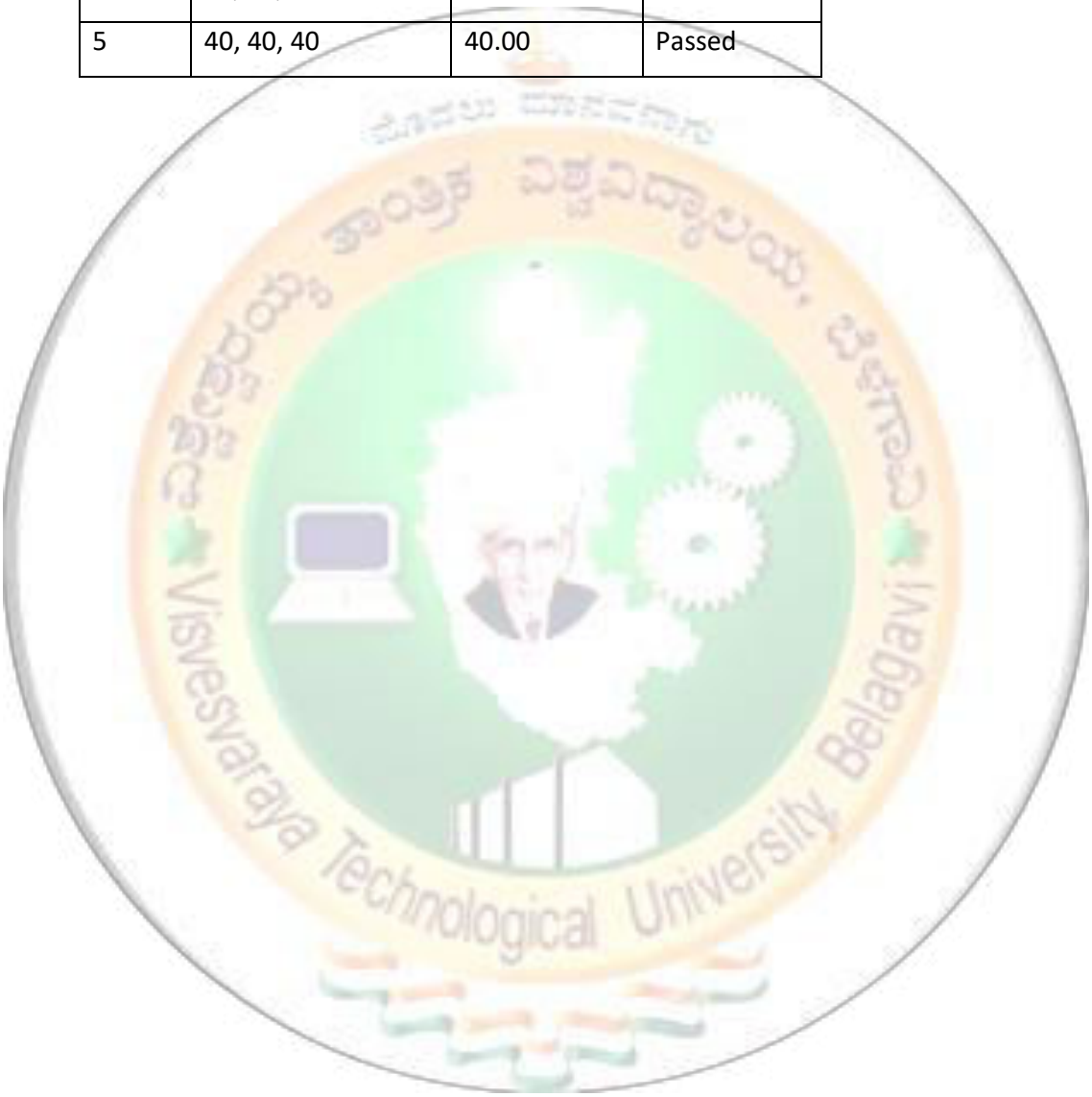
Else → Print “Failed”

Step 7: Stop

Program Code

```
#include <stdio.h>
int isPassed(int m1, int m2, int m3)
{
   if (m1 >= 40 && m2 >= 40 && m3 >= 40)
     return 1;
  else return 0;
}
void main()
{
   int subject1, subject2, subject3;
   float average;
   printf("Enter marks for Subject 1: ");
   scanf("%d", &subject1);
   printf("Enter marks for Subject 2: ");
   scanf("%d", &subject2);
   printf("Enter marks for Subject 3: ");
   scanf("%d", &subject3);
   average = (subject1 + subject2 + subject3) / 3.0;
   printf("Average Marks: %.2f\n", average);
   if (isPassed(subject1, subject2, subject3))
      printf("Result: PASS\n");
  else
      printf("Result: FAIL\n");
}
```

Test Case	Input Marks (Sub1, Sub2, Sub3)	Expected Average	Expected Result
1	50, 60, 70	60.00	Passed
2	40, 39, 50	43.00	Failed
3	80, 90, 100	90.00	Passed
4	30, 40, 50	40.00	Failed
5	40, 40, 40	40.00	Passed



8. In an ATM system, two account balances need to be swapped temporarily for validation. Develop a C program that accepts two balances and uses a function with pointers to swap them. Display the balances before and after swapping.

Problem Description

In an ATM system, you need to temporarily swap two account balances for validation. The program will accept two balances, swap them using a function with pointers, and display the balances before and after swapping.

Input

- Two floating-point numbers representing the two account balances.

Output

- Display the balances before swapping.
- Display the balances after swapping.

Constraints

- Balances can be positive floating-point numbers (including zero).
- Use pointers in the swap function to modify the original variables.

Method

- Use a function swapBalances that takes pointers to two balances.
- Swap the values inside the function.
- Display before and after swapping in main().

Illustration

Example 1:

Input: Balances: 1000.50, 2000.75 (before swapping)

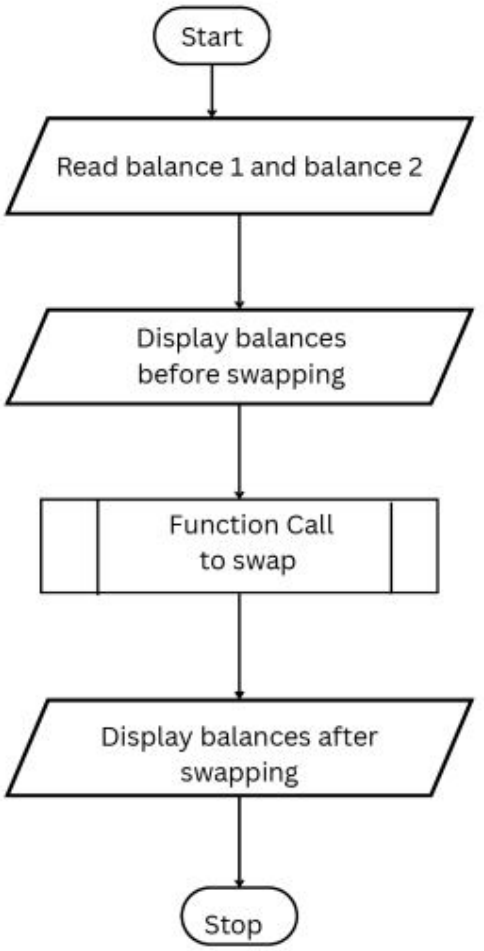
Expected Output: Account 1: 2000.75, Account 2: 1000.50(after swapping)

Example 2:

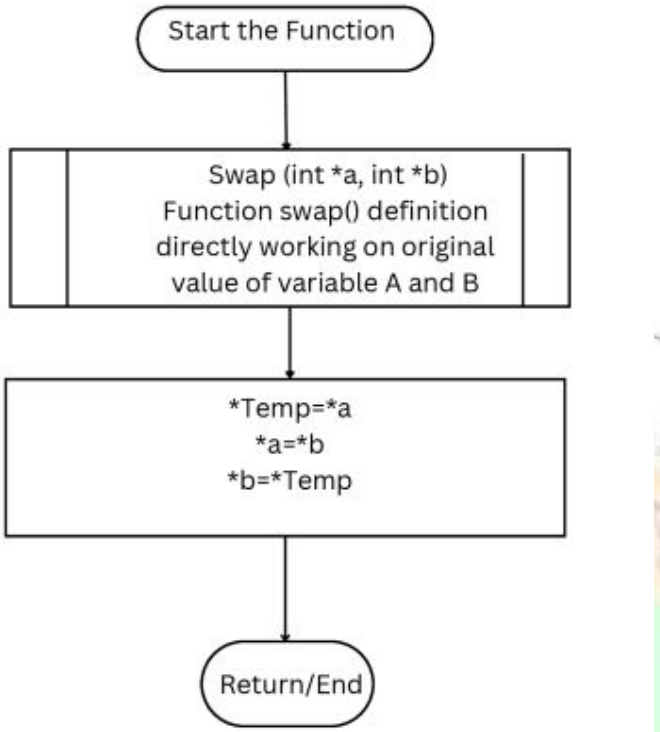
Input: Balances: -100.00, 100.00 (before swapping)

Expected Output: Account 1: 100.00, Account 2: -100.00(after swapping)

Flowchart



(b)Main Program



(a)Function

Algorithm

Step 1: Start

Step 2: Input two account balances: balance1 and balance2

Step 3: Display balances **before swapping**

Step 4: Call a function swap(&balance1, &balance2) that:

- Uses a temporary variable and pointer dereferencing to swap the values

Step 5: Display balances **after swapping**

Step 6: Stop

Program Code

```
#include <stdio.h>
void swap(int *a, int *b)
 {
   int temp = *a;
   *a = *b;
   *b = temp;
}
```

```
void main()
{
    int balance1, balance2;
    printf("Enter balance for Account 1: ");
    scanf("%d", &balance1);
    printf("Enter balance for Account 2: ");
    scanf("%d", &balance2);
    printf("\nBefore Swapping:\n");
    printf("Account 1 Balance: %d\n", balance1);
    printf("Account 2 Balance: %d\n", balance2);
    swap(&balance1, &balance2);
    printf("\nAfter Swapping:\n");
    printf("Account 1 Balance: %d\n", balance1);
    printf("Account 2 Balance: %d\n", balance2);
}
```

Test Case	Input Balances (Account 1, Account 2)	Expected Output After Swapping
1	1000.50, 2000.75	Account 1: 2000.75, Account 2: 1000.50
2	0.00, 500.00	Account 1: 500.00, Account 2: 0.00
3	-100.00, 100.00	Account 1: 100.00, Account 2: -100.00
4	12345.67, 12345.67	Account 1: 12345.67, Account 2: 12345.67

PART B

Program 1: Book Availability in Library (Binary Search)

Problem Statement

A college library has a digital bookshelf system where each book is assigned a unique Book ID. The bookshelf is organized in ascending order of Book IDs. The task is to develop a C program to quickly find whether a book with a specific Book ID is available in the shelf using binary search.

Problem Description

Input:

- Number of books (n).
- List of Book IDs in ascending order.
- A Book ID (key) to be searched.

Output:

- Message indicating whether the book is available or not.

Constraints:

- Number of books > 0.
- Book IDs must be in ascending order.
- Book ID to search should be a valid integer.

Method:

- Use binary search algorithm to efficiently check whether the key exists in the array.

Illustration

Example 1:

Input:

Number of Books = 4

Book IDs = 101 105 110 115

Search ID = 110

Output:

Book is available.

Example 2:

Input:

Number of Books = 4

Book IDs = 101 105 110 115

Search ID = 120

Output:

Book is not available.

Methodology

1. Read the total number of books.
2. Input the Book IDs in ascending order into an array.
3. Input the Book ID to be searched.
4. Initialize low = 0, high = n-1, and found = 0.
5. Apply binary search:
    - Compute mid = (low + high) / 2.
    - If books[mid] == key, set found = 1 and break.
    - If books[mid] < key, adjust low = mid + 1.
    - Else, adjust high = mid - 1.
6. If found == 1, display "Book is available". Otherwise, display "Book is not available".

Algorithm

1. Start
2. Input n.
3. Input n Book IDs into array books[].
4. Input key.
5. Set low = 0, high = n-1, found = 0.
6. Repeat while low <= high:
    - mid = (low + high) / 2.
    - If books[mid] == key: set found = 1 and break.
    - Else if books[mid] < key: low = mid + 1.
    - Else: high = mid - 1.
7. If found == 1: print "Book is available".
8. Else: print "Book is not available".
9. Stop

Program

```
#include <stdio.h>
int main()
{
   int n, i, key, low, high, mid, found = 0;
   printf("Enter number of books: ");
   scanf("%d", &n);
   int books[n];
   printf("Enter %d Book IDs in ascending order:\n", n);
   for(i = 0; i < n; i++)
   scanf("%d", &books[i]);
   printf("Enter Book ID to search: ");
   scanf("%d", &key);
   low = 0;
   high = n - 1;
   while(low <= high)
   {
      mid = (low + high) / 2;
      if(books[mid] == key)
      {
         found = 1;
         break;
      }
      else if(books[mid] < key)
         low = mid + 1;
      else
         high = mid - 1;
   }
   if(found)
      printf("Book is available.\n");
   else
      printf("Book is not available.\n");
   return 0;
}
```

Program 2: Sorting Scores in Descending Order (Bubble Sort)

Problem Statement

A sports teacher has recorded the scores of students in a 100-meter race. To prepare the result sheet, the teacher wants the scores arranged in descending order (from highest to lowest). The task is to develop a C program to sort the scores using Bubble Sort.

Problem Description

Input:

- Number of students (n).
- Scores of students in an array (scores[n]).

Output:

- Scores displayed in descending order.

Constraints:

- Number of students > 0.
- Scores must be integers.

Method:

- Use Bubble Sort algorithm to sort scores in descending order.

Illustration

Example 1:

Input:

Number of Students = 4

Scores = 12 45 32 10

Output:

45 32 12 10

Example 2:

Input:

Number of Students = 4

Scores = 100 95 80 90

Output:

100 95 90 80

Example 3:

Input:

Number of Students = 3

Scores = 5 5 5

Output:

5 5 5

Methodology

1. Read the number of students.
2. Input their scores into an array.
3. Use Bubble Sort algorithm:
    - Compare adjacent elements.
    - If the earlier element is smaller than the next, swap them.
    - Repeat until the entire array is sorted in descending order.
4. Print the sorted scores.

Algorithm

1. Start
2. Input number of students n.
3. Input n scores into array scores[].
4. For i = 0 to n-2:
    - For j = 0 to n-i-2:
        - If scores[j] < scores[j+1], swap them.
5. Print sorted scores in descending order.
6. Stop

Program

```
#include <stdio.h>
int main()
{
   int n, i, j, temp;
   printf("Enter number of students: ");
   scanf("%d", &n);
```

```
    int scores[n];
    printf("Enter the scores:\n");
    for(i = 0; i < n; i++)
        scanf("%d", &scores[i]);
    // Bubble Sort (Descending Order)
    for(i = 0; i < n-1; i++)
    {
        for(j = 0; j < n-i-1; j++)
        {
            if(scores[j] < scores[j+1])
            {
                temp = scores[j];
                scores[j] = scores[j+1];
                scores[j+1] = temp;
            }
        }
    }

    printf("Scores in descending order:\n");
    for(i = 0; i < n; i++)
        printf("%d ", scores[i]);
    return 0;
}
```

Program 3: Warehouse Revenue Calculation

Problem Statement

A small warehouse tracks many units of different products shipped from multiple branches. Another dataset shows how much revenue each product generates per unit. The task is to develop a C program that combines these datasets to calculate the total revenue generated by each branch.

Problem Description

Input:

- Number of branches (branches).
- Number of products (products).
- A 2D array units[branches][products] containing the number of units of each product shipped by each branch.
- An array revenue[products] containing the revenue generated per unit of each product.

Output:

- The total revenue generated by each branch.

Constraints:

- Number of branches > 0.
- Number of products > 0.
- Units and revenue values should be non-negative integers.

Method:

- Use a 2D array for storing product units shipped by each branch.
- Use a 1D array for revenue per unit.
- Multiply and accumulate values to compute branch-wise total revenue.

Illustration

Branches=3 products=2

Branch1[product1, product2]= [20 10], Branch2[product1, product2]=[20 30],

Branch3[product1, product2]=[50 20]

Revenue per unit for each product:

revenue[product1, product2]=[20 30]

Output:

Branch 1 : branch1*revenue=20*20+10*30**=700**

Branch 2 : branch2*revenue=20*20+30*30**=1300**

Branch 3 : branch3*revenue=50*20+20*30**=1600**

Example 1:

Input:

Enter number of branches: 3

Enter number of products: 2

Enter number of units shipped by each branch for each product:

Branch 1:

Product 1 units: 20

Product 2 units: 10

Branch 2:

Product 1 units: 20

Product 2 units: 30

Branch 3:

Product 1 units: 50

Product 2 units: 20

Enter revenue per unit for each product:

Product 1 revenue: 20

Product 2 revenue: 30

Output:

Total revenue per branch:

Branch 1: 700

Branch 2: 1300

Branch 3: 1600

Methodology

1. Read the number of branches and products.
2. Accept the number of units shipped by each branch for each product and store them in a 2D array.
3. Accept the revenue per unit for each product and store them in a 1D array.
4. For each branch, initialize a total revenue counter to 0.

5. Multiply the units shipped by each product with its corresponding revenue and accumulate the result.
6. Display the total revenue for each branch.

Algorithm

1. Start
2. Input number of branches (branches).
3. Input number of products (products).
4. Declare units[branches][products] and revenue[products].
5. Input units shipped for each branch and product.
6. Input revenue per unit for each product.
7. For each branch i:
    - Initialize total = 0.
    - For each product j:
        - Compute total += units[i][j] * revenue[j].
    - Print total revenue for branch i.
8. Stop

Program

```
#include <stdio.h>
int main() {
   int branches, products, i, j;
   printf("Enter number of branches: ");
   scanf("%d", &branches);
   printf("Enter number of products: ");
   scanf("%d", &products);
   int units[branches][products];   // now branches & products are known
   int revenue[products];
     printf("\nEnter number of units shipped by each branch for each product:\n");
   for(i = 0; i < branches; i++)
   {
      printf("Branch %d:\n", i+1);
      for(j = 0; j < products; j++)
      {
```

```
            printf("  Product %d units: ", j+1);
            scanf("%d", &units[i][j]);
        }
    }
    // Input revenue per product
    printf("\nEnter revenue per unit for each product:\n");
    for(j = 0; j < products; j++)
    {
        printf("Product %d revenue: ", j+1);
        scanf("%d", &revenue[j]);
    }
    printf("Total revenue per branch:\n");
    for (i = 0; i < branches; i++)
    {
        int total = 0;
        for (j = 0; j < products; j++)
        {
            total += units[i][j] * revenue[j];
        }
        printf("Branch %d: %d\n", i + 1, total);
    }
    return 0;
}
```

Program 4: Mobile Contact Manager (Concatenating Names)

Problem Statement

A basic mobile contact manager stores first and last names separately. For displaying the full names in the contact list, the system needs to join them manually. Additionally, the system must check the length of each full name to ensure it fits within the screen. The task is to implement these operations in C without using built-in string functions.

Problem Description

Input:

- First name (string).
- Last name (string).

Output:

- Full name formed by joining first and last names with a space.
- Length of the full name.
- Message whether the name fits the screen (length ≤ 20) or not.

Constraints:

- The first and last name must only contain valid characters (no spaces within).
- Full name length must be checked against the screen size (20 characters).

Method:

- Use character arrays to store first name, last name, and full name.
- Concatenate manually using loops (no built-in functions like strcat or strlen).
- Maintain a counter to track the total length of the full name.

Illustration

Example 1:

Input:

First Name: John

Last Name: Doe

Output:

Full Name: John Doe

Length: 8

Name fits the screen.

Example 2:

Input:

First Name: Alexander

Last Name: Johnson

Output:

Full Name: Alexander Johnson

Length: 18

Name fits the screen.

Example 3:

Input:

First Name: Supercalifragilistic

Last Name: Longname

Output:

Full Name: Supercalifragilistic Longname

Length: 29

Name too long for screen.

Methodology (Description)

1. Input first name and last name separately.
2. Initialize an index and counter variable to build the full name manually.
3. Copy characters of the first name into the full name array.
4. Insert a space character between the two names.
5. Copy characters of the last name into the full name array.
6. Terminate the full name with '\0'.
7. Print the full name and its length.
8. Check if the length is greater than 20:
    - If yes → Print **"Name too long for screen."**
    - Otherwise → Print **"Name fits the screen."**

Algorithm

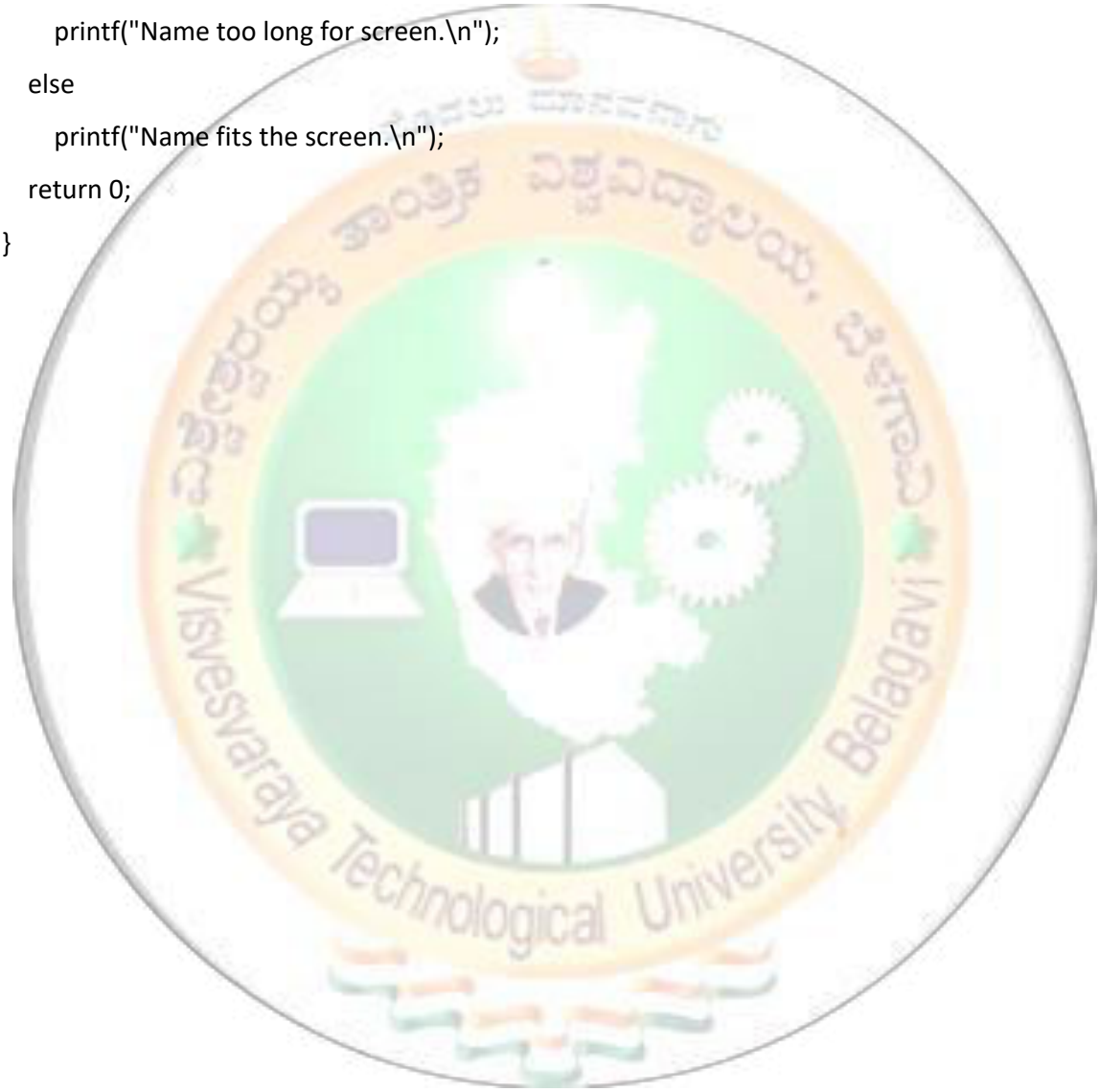
1. Start

2. Input first name into first[].
3. Input last name into last[].
4. Initialize variables:
    - i = 0, j = 0, length = 0.
5. Copy characters of first[] into full[], incrementing length.
6. Insert a space character into full[length]. Increment length.
7. Copy characters of last[] into full[], incrementing length.
8. Insert '\0' at the end of full[].
9. Print the full name and its length.
10. If length > 20 → Print **"Name too long for screen."**
11. Else → Print **"Name fits the screen."**
12. Stop

Program

```
#include <stdio.h>
int main()
{
  char first[50], last[50], full[100];
  int i = 0, j = 0, length = 0;
  printf("Enter first name: ");
  scanf("%s", first);
  printf("Enter last name: ");
  scanf("%s", last);
  // Copy first name
  while(first[i] != '\0')
  {
    full[length++] = first[i++];
  }
  // Add space
  full[length++] = ' ';
  // Copy last name
  while(last[j] != '\0')
  {
    full[length++] = last[j++];
```

```
    }
    // Terminate string
    full[length] = '\0';
    // Output
    printf("Full Name: %s\n", full);
    printf("Length: %d\n", length);
    if(length > 20)
        printf("Name too long for screen.\n");
    else
        printf("Name fits the screen.\n");
    return 0;
}
```



Program 5: Currency Exchange Simulation (Call by Value and Call by Reference)

Problem Statement

A currency exchange booth allows users to convert between two currencies. Before confirming the exchange, the system simulates a swap of the values (preview) without actually changing the original data. In other cases, it updates the actual values (permanent swap). The task is to implement both behaviors using Call by Value and Call by Reference in C.

Problem Description

Input:

- Two integer values representing two currencies (x, y).

Output:

- Preview swap result using **Call by Value**.
- Actual swap result using **Call by Reference**.

Constraints:

- Input values must be integers.
- In Call by Value → original values must remain unchanged.
- In Call by Reference → original values must be modified.

Method:

- Implement two separate functions:
  - swapByValue(int a, int b)
  - swapByReference(int *a, int *b)

Illustration

Example 1:

Input:

x = 10, y = 20

Output:

Preview Swap (By Value): 20 10

After Call by Value: 10 20

Actual Swap (By Reference): 20 10

After Call by Reference: 20 10

Example 2:

Input:

x = 50, y = 75

Output:

Preview Swap (By Value): 75 50

After Call by Value: 50 75

Actual Swap (By Reference): 75 50

After Call by Reference: 75 50

Methodology

1. Input two currency values.
2. Perform **preview swap** by passing values to a function (Call by Value).
3. Inside function, swap values and display result (original values in main remain unchanged).
4. Perform **actual swap** by passing addresses (Call by Reference).
5. Inside function, swap values by dereferencing pointers (original values are updated).
6. Print results after both operations.

Algorithm

1. Start
2. Input values x and y.
3. Call swapByValue(x, y) and display preview result.
4. Print x and y after call by value.
5. Call swapByReference(&x, &y) and display actual result.
6. Print updated x and y.
7. Stop

Program

```
#include <stdio.h>
void swapByValue(int a, int b)
{
   int temp = a;
```

```
    a = b;
    b = temp;
    printf("Preview Swap (By Value): %d %d\n", a, b);
}
void swapByReference(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
    printf("Actual Swap (By Reference): %d %d\n", *a, *b);
}
int main()
{
    int x, y;
    printf("Enter two currency values: ");
    scanf("%d %d", &x, &y);
    swapByValue(x, y);
    printf("After Call by Value: %d %d\n", x, y);
    swapByReference(&x, &y);
    printf("After Call by Reference: %d %d\n", x, y);
    return 0;
}
```

Program 6: Library Book Details (Structures in C)

Problem Statement

A local library needs to store and display details of its books, including title, author, and year of publication. The task is to design a structure that can hold these details and develop a C program to display the list of books entered.

Problem Description

Input:

- Number of books (n).
- For each book:
  - Title (string).
  - Author (string).
  - Year of publication (integer).

Output:

- Details of each book in the format:
- Book i: Title by Author (Year)

Constraints:

- Number of books > 0.
- Title and author must be strings without spaces (for simple input).
- Year must be a positive integer.

Method:

- Use struct Book with fields:
  - title[50]
  - author[50]
  - year

Illustration

Example 1:

Input:

Number of Books = 2

Title: CProgramming, Author: Dennis, Year: 1978

Title: AI, Author: Russell, Year: 2010

Output:

Book 1: CProgramming by Dennis (1978)

Book 2: AI by Russell (2010)

Example 2:

Input:

Number of Books = 1

Title: DBMS, Author: Navathe, Year: 2015

Output:

Book 1: DBMS by Navathe (2015)

Methodology

1. Define a structure Book with title, author, and year.
2. Read the number of books.
3. Create an array of structures to store book details.
4. For each book, input title, author, and year.
5. Display all book details in the specified format.

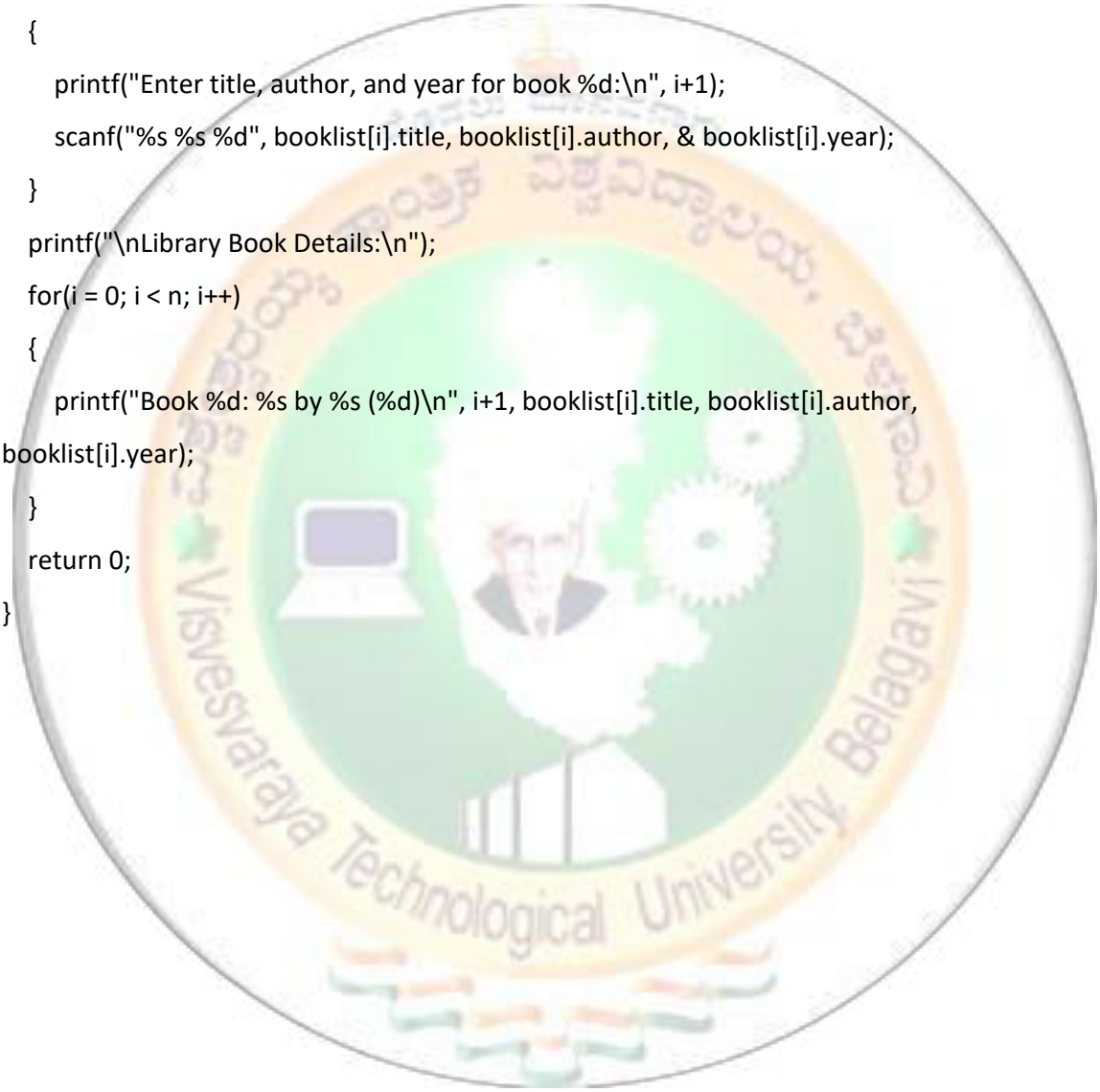
Algorithm

1. Start
2. Define structure Book with fields: title, author, year.
3. Input number of books n.
4. Declare array of structure books[n].
5. For each book: input title, author, and year.
6. For each book: display "Book i: title by author (year)".
7. Stop

Program

```
#include <stdio.h>
struct Book
{
   char title[50];
   char author[50];
   int year;
};
```

```
int main()
{
   int n, i;
   printf("Enter number of books: ");
   scanf("%d", &n);
   struct Book booklist[n];
   for(i = 0; i < n; i++)
   {
      printf("Enter title, author, and year for book %d:\n", i+1);
      scanf("%s %s %d", booklist[i].title, booklist[i].author, & booklist[i].year);
   }
   printf("\nLibrary Book Details:\n");
   for(i = 0; i < n; i++)
   {
      printf("Book %d: %s by %s (%d)\n", i+1, booklist[i].title, booklist[i].author,
booklist[i].year);
   }
   return 0;
}
```



CONTRIBUTERS

1. Dr. Demian Antony Dmello, Professor – Canara Engineering College, Mangalore.
2. Dr. Ravikumar G K – Principal, BGS College of Engineering and Technology, Bangalore.
3. Dr. S Sampath – Professor, AIT, Chikkamagaluru.
4. Dr. Kavitha Sooda – Professor, BMS College of Engineering, Bangalore.
5. Dr. Jagadisha N – Professor, SJBIT, Bangalore.
6. Dr. Suresha D – Professor, SIT, Mangalore.
7. Ms. Tara B B – Asst. Professor, Canara Engineering College, Mangalore.
8. Ms. Akshatha G Baliga – Asst. Professor, Canara Engineering College, Mangalore.
9. Ms. Meghashree – Asst. Professor, Canara Engineering College, Mangalore.