EXPERIMENT3.1: CONDITIONAL STATEMENTS

Q1. Write a program to check if the triangle is valid or not.

If valid, check whether it is Isosceles, Equilateral, Right-angled, or Scalene.

Take the sides as input from the user.*

Aim

To determine if a triangle with given sides is valid and classify its type.

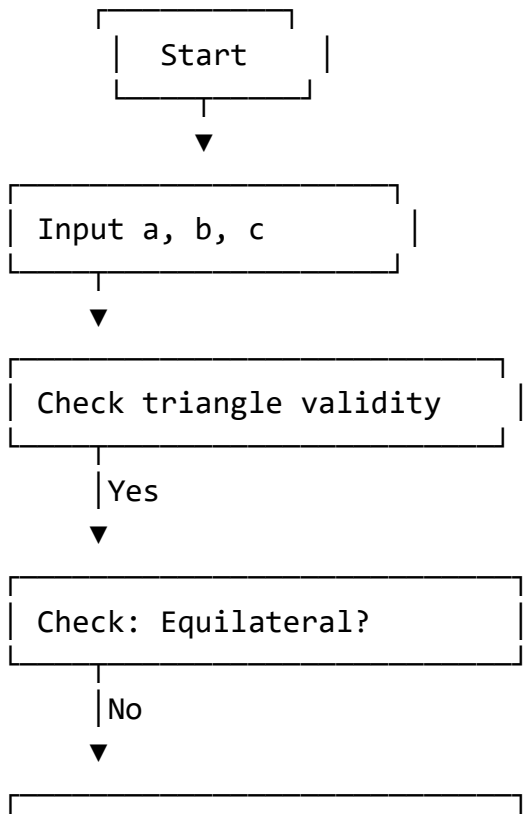
Algorithm

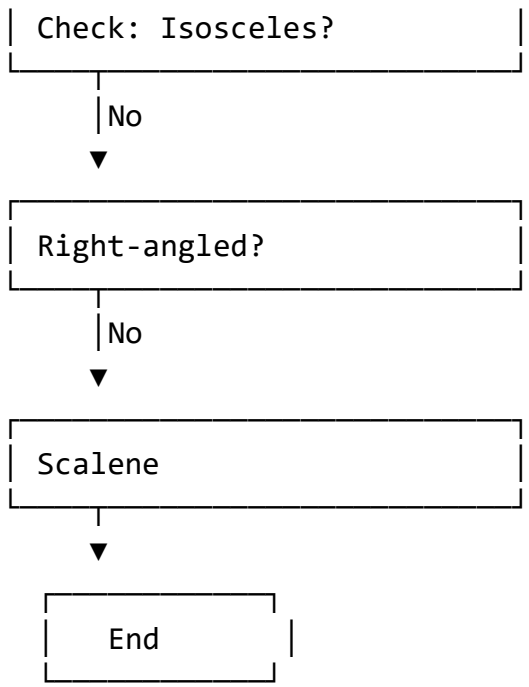
1. Start
2. Input sides a, b, c
3. Check **Triangle Validity Condition**:
    a. If (a + b > c) AND (b + c > a) AND (a + c > b)
4. If NOT valid → print “Invalid triangle”
5. Else:
    a. If all sides equal → Equilateral
    b. Else if two sides equal → Isosceles
    c. Else if ($a^2 + b^2 = c^2$) or similar → Right-angled
    d. Else → Scalene
6. End

Pseudocode

```
BEGIN
  INPUT a, b, c
  IF a + b > c AND b + c > a AND a + c > b THEN
        IF a = b AND b = c THEN PRINT "Equilateral"
        ELSE IF a = b OR b = c OR a = c THEN PRINT "Isosceles"
        ELSE IF (a*a + b*b = c*c) OR ... THEN PRINT "Right Angled"
        ELSE PRINT "Scalene"
  ELSE
        PRINT "Invalid Triangle"
END
```

Flowchart



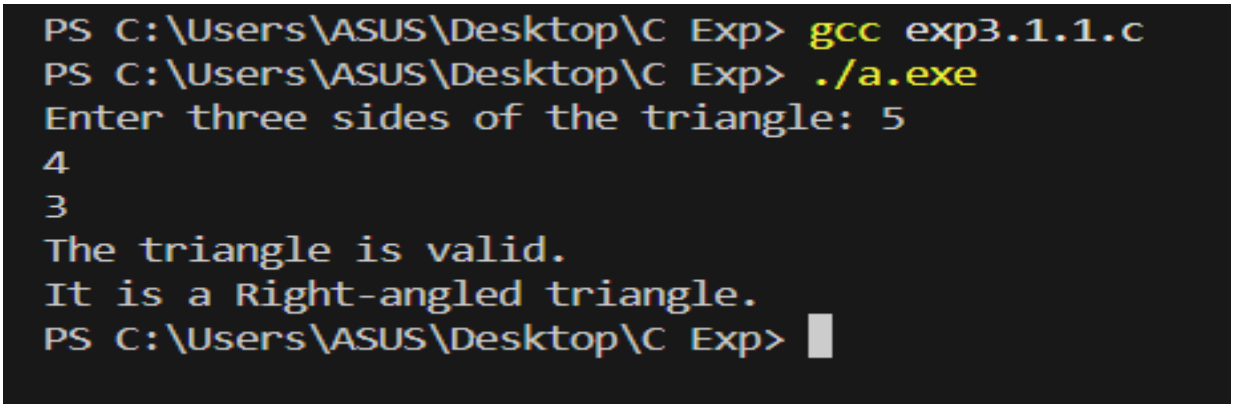


C Program

```
#include <stdio.h>

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

    if (a + b > c && b + c > a && a + c > b) {
        printf("Triangle is valid\n");

        if (a == b && b == c)
            printf("Equilateral Triangle");
        else if (a == b || b == c || a == c)
            printf("Isosceles Triangle");
        else if ((a*a + b*b == c*c) ||
                 (b*b + c*c == a*a) ||
                 (a*a + c*c == b*b))
```

```
            printf("Right Angled Triangle");
        else
            printf("Scalene Triangle");
    } else {
        printf("Invalid Triangle");
    }

    return 0;
}
```

Output



Conclusion

The program correctly identifies a valid triangle and classifies its type.

Q2. Write a program to compute BMI and print values according to categories.

BMI Formula:

[

BMI = \frac{Weight(kg)}{Height(m)^2}

]

Aim

To compute BMI and categorize it as per the given BMI ranges.

BMI Categories

BMI Range	Category
< 15	Starvation
15 – 17.5	Anorexic
17.5 – 18.5	Underweight
18.5 – 25	Ideal
25 – 30	Overweight
30 – 40	Obese
> 40	Morbidly Obese

Algorithm

1. Start
2. Input weight and height
3. Compute BMI

4. Check ranges using if-else ladder
5. Display the category
6. End

Pseudocode

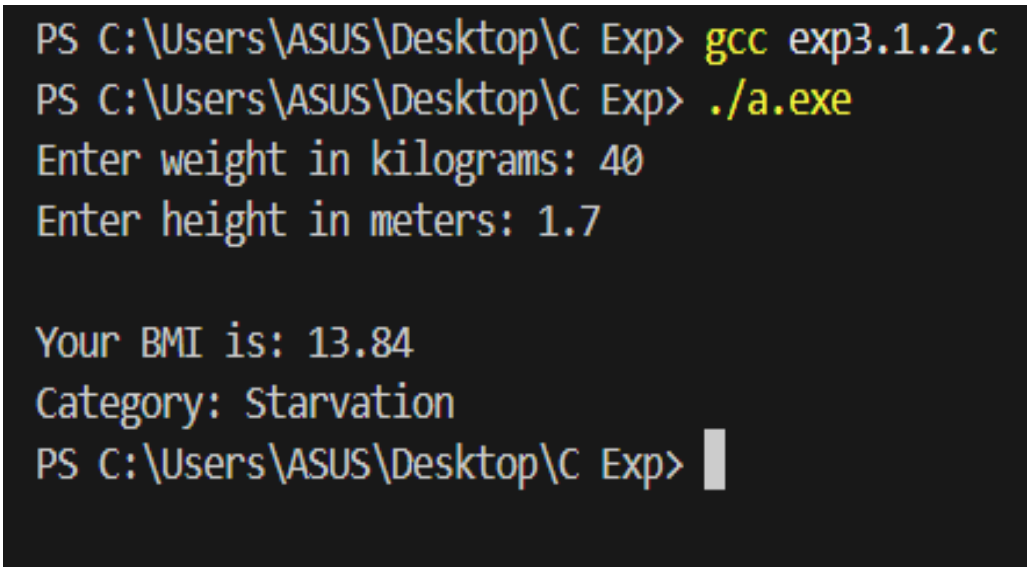
```
BEGIN
  INPUT weight, height
  BMI = weight/(height*height)
  IF BMI < 15 THEN Starvation
  ELSE IF BMI < 17.5 THEN Anorexic
  ELSE IF BMI < 18.5 THEN Underweight
  ELSE IF BMI < 25 THEN Ideal
  ELSE IF BMI < 30 THEN Overweight
  ELSE IF BMI < 40 THEN Obese
  ELSE Morbidly Obese
END
```

C Program

```
#include <stdio.h>

int main() {
    float weight, height, bmi;

    printf("Enter weight (kg): ");
    scanf("%f", &weight);

    printf("Enter height (m): ");
    scanf("%f", &height);

    bmi = weight / (height * height);

    printf("BMI = %.2f\n", bmi);
```

```
    if (bmi < 15)
        printf("Starvation");
    else if (bmi < 17.5)
        printf("Anorexic");
    else if (bmi < 18.5)
        printf("Underweight");
    else if (bmi < 25)
        printf("Ideal");
    else if (bmi < 30)
        printf("Overweight");
    else if (bmi < 40)
        printf("Obese");
    else
        printf("Morbidly Obese");

    return 0;
}
```

Output



Conclusion

BMI is calculated correctly and categorized according to the given medical ranges.

Q3. Write a program to check if three points (x1,y1), (x2,y2), (x3,y3) are collinear.

Aim

To check whether 3 points lie on the same straight line using the area method.

Method (Area of Triangle = 0)

Points are collinear if:

[

(x1(y2 - y3) + x2(y3 - y1) + x3(y1 - y2)) = 0

]

Algorithm

1. Input coordinates of the three points
2. Apply the formula
3. If expression = 0 → Collinear
4. Else → Not Collinear

Pseudocode

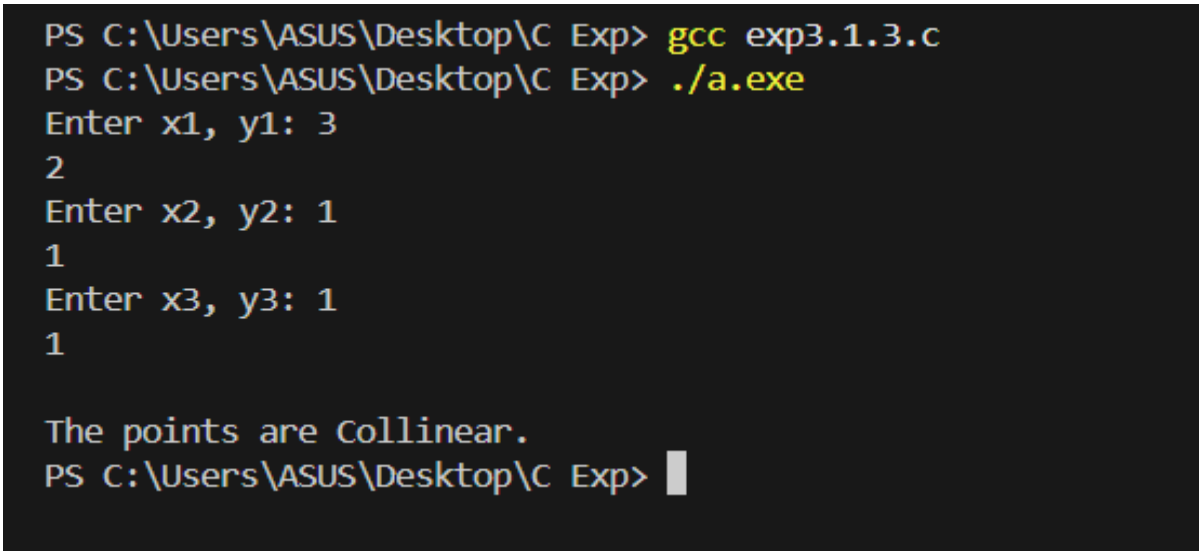
```
BEGIN
  INPUT x1,y1,x2,y2,x3,y3
  compute A = x1(y2–y3)+x2(y3–y1)+x3(y1–y2)
  IF A == 0 THEN collinear
  ELSE not collinear
END
```

C Program

```
#include <stdio.h>

int main() {
    int x1, y1, x2, y2, x3, y3;
    int area;

    printf("Enter three points (x y): ");
    scanf("%d %d %d %d %d %d", &x1,&y1,&x2,&y2,&x3,&y3);

    area = x1*(y2-y3) + x2*(y3-y1) + x3*(y1-y2);

    if (area == 0)
        printf("Points are collinear");
    else
        printf("Points are NOT collinear");
```

```
    return 0;
}
```

Output



Conclusion

The program correctly determines collinearity using the triangle area formula.

Q4. According to Gregorian calendar, 01-01-1900 was Monday.

If any year is input, find the day on 01-01 of that year.**

Aim

To determine the weekday of 1st January of any input year.

Logic

Count total days from 1900 to given year:

- Add 365 days per normal year
- Add 366 days per leap year
- Day index = total_days % 7

Mapping:

0 → Monday

1 → Tuesday

2 → Wednesday

3 → Thursday

4 → Friday

5 → Saturday

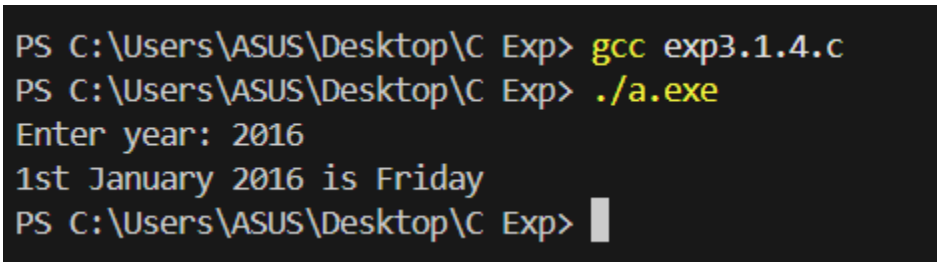
6 → Sunday

C Program

```
#include <stdio.h>

int main() {
    int year, y, days = 0;
```

```
    printf("Enter year: ");
    scanf("%d", &year);

    for (y = 1900; y < year; y++) {
        if ((y % 4 == 0 && y % 100 != 0) || (y % 400 == 0))
            days += 366;
        else
            days += 365;
    }

    int day = days % 7;

    switch(day) {
        case 0: printf("Monday"); break;
        case 1: printf("Tuesday"); break;
        case 2: printf("Wednesday"); break;
        case 3: printf("Thursday"); break;
        case 4: printf("Friday"); break;
        case 5: printf("Saturday"); break;
        case 6: printf("Sunday"); break;
    }

    return 0;
}
```

Output



Conclusion

The program correctly computes the weekday for 1-Jan of any given year.

Q5. Using the ternary operator, input the length & breadth of at least 3 rectangles and print which rectangle has the highest perimeter.

Aim

To find which of three rectangles has the largest perimeter using a ternary operator.

Algorithm

1. Input length & breadth of 3 rectangles
2. Compute perimeters
3. Use ternary operator to compare
4. Display rectangle number with highest perimeter

C Program

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

```
int main() {
    int l1,b1,l2,b2,l3,b3;
    int p1, p2, p3;

    printf("Enter length & breadth of rectangle 1: ");
    scanf("%d %d", &l1, &b1);

    printf("Enter length & breadth of rectangle 2: ");
    scanf("%d %d", &l2, &b2);

    printf("Enter length & breadth of rectangle 3: ");
    scanf("%d %d", &l3, &b3);

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

    int max = (p1 > p2 && p1 > p3) ? 1 :
              (p2 > p3) ? 2 : 3;

    printf("Rectangle %d has the highest perimeter.\n", max);

    return 0;
}
```

Output

.

```
PS C:\Users\ASUS\Desktop\C Exp> gcc exp3.1.5.c
PS C:\Users\ASUS\Desktop\C Exp> ./a.exe
Enter length and breadth of Rectangle 1: 4
3
Enter length and breadth of Rectangle 2: 6
5
Enter length and breadth of Rectangle 3: 8
7

Perimeter of Rectangle 1 = 14.00
Perimeter of Rectangle 2 = 22.00
Perimeter of Rectangle 3 = 30.00

Highest Perimeter = 30.00
Rectangle 3 has the highest perimeter.
PS C:\Users\ASUS\Desktop\C Exp>
```

Conclusion

Using the ternary operator simplifies conditional comparisons to find the rectangle with the highest perimeter.

.