EXPERIMENT-3.1

(CONDITIONAL STATEMENTS)

AIM:Write a program to take check if the triangle is valid or not. If the validity is established, do check if the triangle is isosceles, equilateral, right angle, or scalene. Take sides of the triangle as input from a user.

CODE:

```
//WRITE A PROGRAM TO TAKE CHECK IF THE TRIANGLE IS VALID OR NOT.IF THE VALIDITY IS ESTABLISHED, DO CHECK IF THE TRIANGLE IS ISOSCELES,EQUILATERAL,RIGHT ANGLE, OR SCALENE. TAKE SIDES OF THE TRIANGLE AS INPUT FROM A USER.
#include <stdio.h>
int main()
{
int a, b, c;//declaration of 3 sides
printf("Enter 3 sides of triangle:\n");
scanf("%d %d %d", &a, &b, &c);
if (a + b > c && a + c > b && b + c > a)
{
printf("Triangle is valid.\n");
if (a == b && b == c)
printf("Equilateral Triangle\n");
else if (a == b || b == c || a == c)
printf("Isosceles Triangle\n");
else if (a*a + b*b == c*c || a*a + c*c == b*b || b*b + c*c == a*a)
printf("Right-angled Triangle\n");
else
printf("Scalene Triangle\n");
}
else
{
printf("Not a valid Triangle\n");
}
return 0;
}
```

OUTPUT 1:

Enter 3 sides of triangle:

0

0

0

Invalid input

OUTPUT 2:

Enter 3 sides of triangle:

-213

1

123

Not a valid Triangle

OUTPUT 3:

Enter 3 sides of triangle:

12

13

14

Triangle is valid.

Scalene Triangle

QUESTION-2:Write a program to compute the BMI Index of the person and print the BMI values as per the following ranges. You can use the

following formula to computeBMI=weight(kgs)/Height(Mts)*Height(Mts).

CODE:

```
//WRITE A PROGRAM TO COMPUTE THE BMI INDEX OF THE PERSON AND PRINT THE BMI VALUES AS PER RANGES.YOU CAN USE THE FOLLOWING FORMULA TO COMPUTE BMI[WEIGHT(kgs)/HEIGHT(Mts)*HEIGHT(Mts(

#include <stdio.h>
int main()
{
float weight, height, bmi;//declaration
printf("Enter weight (kg): ");
scanf("%f", &weight);
printf("Enter height (m): ");
scanf("%f", &height);

bmi = weight / (height * height);//calculating bmi

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

if (bmi < 15)
printf("Starvation\n");
else if (bmi >= 15.1 && bmi <= 17.5)
printf("Anorexic\n");
else if (bmi >= 17.6 && bmi <= 18.5)
printf("Underweight\n");
else if (bmi >= 18.6 && bmi <= 24.9)
printf("Ideal\n");
else if (bmi >= 25 && bmi <= 25.9)
```

```
printf("Overweight\n");
else if (bmi >= 30 && bmi <= 39.9)
printf("Obese\n");
else if (bmi >= 40)
printf("Morbidity Obese\n");

return 0;
}
```

QUESTION-3: Write a program to check if three points(x1,y1),(x2,y2) and(x3,y3) are collinear or not.

CODE:

```
//WRITE A PROGRAM TO CHECK IF THREE POINTS(X1,Y1),(X2,Y2) AND (X3,Y3) ARE COLLINEAR OR NOT.

#include <stdio.h>
int main()
{
float x1, y1, x2, y2, x3, y3;//declaration
printf("Enter x1 y1: \n");
scanf("%f %f", &x1, &y1);
printf("Enter x2 y2: \n");
scanf("%f %f", &x2, &y2);
printf("Enter x3 y3: \n");
scanf("%f %f", &x3, &y3);
if((x1==x2 && y1==y2)||(x1==x3 && y1==y3)||(x2==x3 && y2==y3))
{
 printf("Two or more points are identical. Degenerate  case.\n");
}
else
{
 if ((y2 - y1) * (x3 - x2) == (y3 - y2) * (x2 -          x1))//condition of collinearity
 printf("Points are Collinear\n");
 else
 printf("Points are NOT Collinear\n");
}
return 0;
}
```

OUTPUT 1:

Enter x1 y1:

3

3

Enter x2 y2:

3

3

Enter x3 y3:

3

3

Two or more points are identical. Degenerate case.

OUTPUT 2:

Enter x1 y1:

3.12

1.3

Enter x2 y2:

2.2

2.2

Enter x3 y3:

-3.2

-2.1

Points are NOT Collinear

OUTPUT 3:

Enter x1 y1:

1

1

Enter x2 y2:

2

2

Enter x3 y3:

3

3

Points are Collinear

QUESTION-4:According to the gregorian calendar, it was Monday on the date 01/01/01. If Any year is input through the keyboard write a program to find out what is the day on 1st January of this year.

CODE:

```
/*ACCORDING TO THE GREGORIAN CALENDER,IT WAS MONDAY ON THE DATE 01/01/01. IF ANY YEAR IS INPUT THROUGH THE KEYBOARD WRITE A PROGRAM TO FIND OUT WHAT IS THE DAY ON 1ST JANUARY OF THIS YEAR*/
#include <stdio.h>
int main()
{
int year, days, day;//declaration
printf("Enter year: ");
scanf("%d", &year);
if(year>0)
{
days = (year - 1) * 365 + (year - 1) / 4 - (year - 1) / 100 + (year - 1) / 400;//calculating total number of days
```

```
day = (days + 1) % 7;//calculating weekday
printf("On 1st Jan %d, it was ", year);
if (day == 1)
printf("Monday\n");
else if (day == 2)
printf("Tuesday\n");
else if (day == 3)
printf("Wednesday\n");
else if (day == 4)
printf("Thursday\n");
else if (day == 5)
printf("Friday\n");
else if (day == 6)
printf("Saturday\n");
else printf("Sunday\n");
}
else
printf("input is invalid");
return 0;
}
```

OUTPUT 1:

Enter year: 0

input is invalid

OUTPUT 2:

Enter year: 1

On 1st Jan 1, it was Monday

[Note:Gregorian Calendar starts from 1]

QUESTION-5:Write a program using ternary operator, the user should in put the length and breadth of a rectangle, one has to find out which

rectangle has the highest perimeter. The minimum number of rectangles should be three.

CODE:

```
/*WRITE A PROGRAM USING TERNARY OPERATOR, THE USER SHOULD INPUT THE LENGTH AND BREADTH OF A RECTANGLE, ONE HAS TO FIND OUT WHICH RECTANGLE HAS THE HIGHEST PERIMETER.THE MINIMUN NUMBER OF RECTANGLES SHOULD BE THREE*/

#include <stdio.h>
int main()
{
Float l1, b1, l2, b2, l3, b3;//declaration
float p1, p2, p3, max;
printf("Enter length and breadth of rectangle 1: \n");
scanf("%f %f", &l1, &b1);
printf("Enter length and breadth of rectangle 2: \n");
scanf("%f %f", &l2, &b2);
printf("Enter length and breadth of rectangle 3: \n");
scanf("%f %f", &l3, &b3);
if (l1 <= 0 || b1 <= 0 || l2 <= 0 || b2 <= 0 || l3 <= 0 || b3 <= 0)
{
printf(" Invalid input! Rectangle sides must be positive.\n");
return 0;
}
p1 = 2 * (l1 + b1);
p2 = 2 * (l2 + b2);
p3 = 2 * (l3 + b3);
if(p1==p2 || p2==p3 ||p1==p3)
{
printf("All perimeters are equal");
}
max = (p1 > p2) ? ((p1 > p3) ? 1:3) : ((p2 > p3)? 2 : 3);
printf("Highest perimeter is of rectangle %d\n", max);
return 0;
}
```

OUTPUT:

Enter length and breadth of rectangle 1:

45

1.5

Enter length and breadth of rectangle 2:

10.5

23.4

Enter length and breadth of rectangle 3:

27.6

57.8

Highest perimeter is of rectangle 3.000000

EXPERIMENT-3.2

(Loops (Enhanced with Edge Case Handling))

Aim:

To implement different loop-based programs in C and handle all possible edge cases effectively.

Theory:

Loops are used to execute a block of code repeatedly until a specified condition becomes false. Common types are for, while, and do-while loops.

Program 1: Count Positive, Negative, and Zero Numbers

Code:

```c
#include <stdio.h>
int main()
{
int num, pos = 0, neg = 0, zero = 0;
char choice;
do {
printf("Enter your number: ");
if (scanf("%d", &num) != 1)

{

printf("Invalid input! Please enter an integer.\n");
while (getchar() != '\n');//clears buffer so that scanf takes fresh input
continue;


}
if (num > 0)
pos++;
else if (num < 0)
neg++;
else
zero++;
printf("Do you want to continue (y/n)? ");
scanf(" %c", &choice);
} while (choice == 'y' || choice == 'Y');
printf("\nCount Summary:\n");
printf("Positive numbers: %d\n", pos);
printf("Negative numbers: %d\n", neg);
printf("Zeroes: %d\n", zero);
return 0;
}
```

Edge Case Inputs and Outputs:

Input Sequence	Output
5, -2, 0, n	Positive = 1, Negative = 1, Zero = 1
0, 0, 0, n	Positive = 0, Negative = 0, Zero = 3
-1, -2, -3, n	Positive = 0, Negative = 3, Zero = 0

Program 2: Multiplication Table

Code:

```c
#include <stdio.h>
int main()
{
int n;
printf("Enter a number for multiplication table: ");
if (scanf("%d", &n) != 1)
{
printf("Invalid input! Please enter an integer.\n");
return 0;
}
if (n == 0)
{
printf("Multiplying by 0 always gives 0.\n");
return 0;
}
for (int i = 1; i <= 10; i++)
{
printf("%d * %d = %d\n", n, i, n * i);
}
return 0;
}
```

Edge Case Demonstration:

Input	Output
5	Prints 5 × 1 = 5 ... 5 × 10 = 50
0	Prints 0 × 1 = 0 ... (all zeroes)
-3	Prints -3 × 1 = -3 ... -3 × 10 = -30

Program 3: Pattern Generation

Code(a):

```c
#include <stdio.h>

int main()
{
 int num = 1;
 printf("Pattern A:\n");
 for (int i = 1; i <= 3; i++)
 {
  for (int j = 1; j <= i; j++)
  {
   printf("%d", num++);
  }
  printf("\n");
 }
 return 0;
}
```

Output:

Pattern A:

1
23
456

Code(b):

```c
#include <stdio.h>

int main()
{
 int rows = 5;
 printf("Pattern B:\n");

 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);
 }
 printf("\n");
 }
 return 0;
}
```

Pattern B:

1
11
121
1331
14641

Program 4: Population Growth

Code:

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

```
{
double population = 100000; //declaration
double rate = 0.10;
printf("Yearly Population Growth:\n");
for (int year = 1; year <= 10; year++)
{
population=population+population * rate;
printf("Year %d: %.0lf\n", year, population);
}
return 0;
}
```

Edge Case Test:

Initial Population	Rate	Years	Result
100000	10%	10	Prints population from 110000 up to 259374
0	10%	10	Always 0
100000	0%	10	Constant 100000 each year

Program 5: Ramanujan Numbers

Code:

```
#include <stdio.h>
#include <math.h>
int main()
{
 int limit;
 printf("Enter limit to find Ramanujan numbers: ");
 if (scanf("%d", &limit) != 1 || limit <= 0)
 {
  printf("Invalid limit! Please enter a positive integer.\n");
```

```
 return 0;
}
 printf("Ramanujan Numbers up to %d:\n", limit);
 for (int a = 1; a * a * a < limit; a++)
 {
  for (int b = a + 1; b * b * b < limit; b++)
  {
   int sum1 = a * a * a + b * b * b;
   for (int c = a + 1; c * c * c < limit; c++)
   {
    for (int d = c + 1; d * d * d < limit; d++)
    {
     int sum2 = c * c * c + d * d * d;
     if (sum1 == sum2 && sum1 <= limit)
     {
      printf("%d = %d^3 + %d^3 = %d^3 + %d^3\n", sum1, a, b, c, d);
     }
    }
   }
  }
 }
return 0;
}
```

Edge Case Inputs and Outputs:

Input	Output
2000	$1729 = 1^3 + 12^3 = 9^3 + 10^3$
0	Invalid limit message
-100	Invalid limit message

Observations:

1. Programs correctly validate user inputs and handle edge cases like negative numbers, zeros, and invalid entries.
2. Each loop type (for, while, do-while) is implemented appropriately based on the problem.
3. Output formatting ensures clarity and correctness.

Common Errors and Fixes:

Error	Cause	Correction
Infinite loop	Missing input validation	Added proper scanf checks and input flushing
Incorrect pattern alignment	Missing newline or wrong loop range	Adjusted nested loop conditions
Wrong output for population	Integer overflow	Used double data type