Subject: Professional Development Skills	**Lecture - 2**	**Date: 24.12.2024 / Tuesday**	**No. of Programs : 02**

1. Ramesh’s basic salary is input, his dearness allowance is 40% of basic salary, and house rent allowance is 20% of basic salary.

C

```
#include <stdio.h>

int main() {
    float basic_salary, da, hra, gross_salary;

    // Input basic salary
    printf("Enter Ramesh's basic salary: ");
    scanf("%f", &basic_salary);

    // Calculate DA and HRA
    da = 0.4 * basic_salary;
    hra = 0.2 * basic_salary;

    // Calculate gross salary
    gross_salary = basic_salary + da + hra;

    // Output results
 printf("Dearness Allowance: %.2f\n", da);
printf("House Rent Allowance: %.2f\n", hra);
    printf("Gross Salary: %.2f\n", gross_salary);

    return 0;
}
```

JAVA

```
import java.util.Scanner;

public class SalaryCalculation {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Input basic salary
        System.out.print("Enter Ramesh's basic salary: ");
        double basicSalary = scanner.nextDouble();

        // Calculate DA and HRA
        double da = 0.4 * basicSalary;
        double hra = 0.2 * basicSalary;

        // Calculate gross salary
        double grossSalary = basicSalary + da + hra;

        // Output results
     System.out.printf("Dearness Allowance: %.2f\n", da);
System.out.printf("House Rent Allowance: %.2f\n", hra);
        System.out.printf("Gross Salary: %.2f\n", grossSalary);

        scanner.close();
    }
}
```

PYTHON

```
# Input basic salary
basic_salary = float(input("Enter Ramesh's basic salary: "))

# Calculate DA and HRA
da = 0.4 * basic_salary
hra = 0.2 * basic_salary

# Calculate gross salary
gross_salary = basic_salary + da + hra

# Output results
print(f"Dearness Allowance: {da:.2f}")
print(f"House Rent Allowance: {hra:.2f}")
print(f"Gross Salary: {gross_salary:.2f}")
```

1. **Time Complexity:**
   - Calculations (DA, HRA, Gross Salary) involve constant-time operations: **O(1).**
   - Input and output operations are also **O(1).**
   - **Overall Time Complexity: O(1).**
2. **Space Complexity:**
   - Variables used: basic_salary, da, hra, gross_salary (constant memory).

- **Overall Space Complexity: O(1).**

Each program has constant time and space complexity since the operations and memory usage do not scale with input size.

Why %.2f is Used

1. **%f:**
    - The %f format specifier outputs floating-point numbers by default with six digits after the decimal point (e.g., 123.456789).
    - If you use %f, it will display the result with unnecessary precision, which may not look neat or practical for salary calculations.
2. **%.2f:**
    - The .2 part specifies that only **two decimal places** should be displayed.
    - It is commonly used in financial or practical applications to show currency or measurements, as more than two decimal places are usually not meaningful.

Example Without and With %.2f

Let’s assume da = 123.456789.

- Using %f:

  ```
  printf("%f\n", da);
  ```

  Output:

  ```
  123.456789
  ```

- Using %.2f:

  arduino

  ```
  printf("%.2f\n", da);
  ```

  Output:

  ```
  123.46
  ```

  The value is rounded to two decimal places for better readability.

If you do not use %.2f but only %f, the output will include six decimal places by default, which is generally less desirable for formatted financial results.

what if use give %.3f..

If you use %.3f instead of %.2f, the output will display **three digits after the decimal point** instead of two. This simply changes the level of precision for the floating-point value being displayed.

Example: Using %.3f

Let’s assume a value of 123.456789.

```
printf("%.3f\n", 123.456789);
```

- **Output**: 123.457

Explanation

1. The .3 specifies that the number should be rounded to **three decimal places**.
2. If the number has more than three decimal places, the value will be **rounded**:
    - Example: 123.456789 becomes 123.457.
3. If the number has fewer than three decimal places, zeros will be appended to ensure exactly three decimal places:
    - Example: 123.4 becomes 123.400.

Comparison Between %.2f and %.3f

Specifier	Input Value	Output
%f	123.456789	123.456789
%.2f	123.456789	123.46
%.3f	123.456789	123.457
%.3f	123.4	123.400

When to Use %.3f

- You would use %.3f when higher precision is necessary, such as in scientific calculations or scenarios where three decimal places are meaningful.
- In the given salary problem, two decimal places (%.2f) are more practical, but you can choose %.3f if you want to display the results with three decimal places for added precision.

2. The distance between two cities (in km.) is input. Program to convert and print this distance in meters, feet, inches and centimeters

C

```
#include <stdio.h>

int main() {
   float distance_km, distance_m, distance_ft,
distance_in, distance_cm;

   // Input distance in kilometers
   printf("Enter the distance between two cities
(in km): ");
   scanf("%f", &distance_km);

   // Convert to other units
   distance_m = distance_km * 1000;
// Meters
   distance_cm = distance_m * 100;
 // Centimeters
   distance_ft = distance_m * 3.28084;
// Feet
   distance_in = distance_ft * 12;
// Inches

   // Print results
   printf("Distance in meters: %.2f m\n",
distance_m);
   printf("Distance in centimeters: %.2f cm\n",
distance_cm);
   printf("Distance in feet: %.2f ft\n",
distance_ft);
   printf("Distance in inches: %.2f in\n",
distance_in);

   return 0;
}
```

JAVA

```
import java.util.Scanner;

public class DistanceConverter {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      // Input distance in kilometers
      System.out.print("Enter the distance between two
cities (in km): ");
      double distanceKm = scanner.nextDouble();

      // Convert to other units
      double distanceM = distanceKm * 1000;
// Meters
      double distanceCm = distanceM * 100;
// Centimeters
      double distanceFt = distanceM * 3.28084;
// Feet
      double distanceIn = distanceFt * 12;
// Inches

      // Print results
      System.out.printf("Distance in meters: %.2f m\n",
distanceM);
      System.out.printf("Distance in centimeters: %.2f
cm\n", distanceCm);
      System.out.printf("Distance in feet: %.2f ft\n",
distanceFt);
      System.out.printf("Distance in inches: %.2f in\n",
distanceIn);

      scanner.close();
   }
}
```

PYTHON

```
# Input distance in kilometers
distance_km = float(input("Enter the distance
between two cities (in km): "))

# Convert to other units
distance_m = distance_km * 1000
# Meters
distance_cm = distance_m * 100
# Centimeters
distance_ft = distance_m * 3.28084
# Feet
distance_in = distance_ft * 12          # Inches

# Print results
print(f"Distance in meters: {distance_m:.2f} m")
print(f"Distance in centimeters:
{distance_cm:.2f} cm")
print(f"Distance in feet: {distance_ft:.2f} ft")
print(f"Distance in inches: {distance_in:.2f} in")
```

Time Complexity

1. **Input Operation:** Reading the distance in kilometers takes **O(1).**
2. **Conversion Calculations:** Each conversion (meters, centimeters, feet, inches) involves simple arithmetic operations, each taking **O(1)**. There are **4 conversions**, so this part is also **O(1).**
3. **Output Operations:** Printing the results involves constant-time operations, O(1). **Overall Time Complexity: O(1).**

Space Complexity

1. The program uses variables to store:
   - The input (distance_km).
   - Converted distances (distance_m, distance_cm, distance_ft, distance_in).
2. These are constant-sized variables, and no additional data structures are used. **Overall Space Complexity: O(1).**