Subject: Professional Development Skills	Lecture - 4	Date: 26.12.2024 / Thursday	No. of Programs : 04

Scenario: Paper of size A0 has dimensions 1189 mm * 841 mm. each subsequent size A(n) is defined as A(n-1) cut in half parallel to its shorter sides.

Explanation:
Imagine you have a big rectangle paper called **A0**. Now, let's do something fun with it:

1. **Cut it in half**:
   You take the paper and cut it in the middle, across the shorter side. Now you have two smaller pieces. These are called **A1**.
2. **Keep cutting**:
   If you take one of those smaller pieces (A1) and cut it in half the same way, you get two even smaller pieces called **A2**.
3. **It's like magic**:
   No matter how many times you cut the paper in half, the new pieces always look like a smaller version of the original big one. They're always the same shape.
4. **Names get bigger, but paper gets smaller**:
   - A0 is the biggest.
   - A1 is smaller.
   - A2 is even smaller.
   - And so on...

It's a simple trick to make paper sizes that all match perfectly, even if they're tiny or huge!

Paper Size	Dimensions (mm)	Area (m^2)
A0	1189 × 841	1.000
A1	841 × 594	0.500
A2	594 × 420	0.250
A3	420 × 297	0.125
A4	297 × 210	0.0625

C

```c
#include <stdio.h>

int main() {
    int n;
    double length = 1189, width = 841; // A0 dimensions
    printf("Enter the maximum paper size  ");
    scanf("%d", &n);

    // Calculate and print A0
printf("A0: %.0lf mm x %.0lf mm\n", length, width);

    // Calculate and print A1
    if (n >= 1) {
        double temp = length;
        length = width;
        width = temp / 2;
```

JAVA

```java
public class PaperSizes {
    public static void main(String[] args) {
        java.util.Scanner scanner = new java.util.Scanner(System.in);
        System.out.print("Enter the maximum paper size : ");
        int n = scanner.nextInt();
        scanner.close();

        double length = 1189, width = 841; // A0 dimensions

        // Calculate and print A0
        System.out.printf("A0: %.0f mm x %.0f mm\n", length, width);

        // Calculate and print A1
        if (n >= 1) {
            double temp = length;
            length = width;
            width = temp / 2;
```

PYTHON

```python
n = int(input("Enter the maximum paper size : "))
length, width = 1189, 841  # A0 dimensions

# Calculate and print A0
print(f"A0: {int(length)} mm x {int(width)} mm")

# Calculate and print A1
if n >= 1:
    temp = length
    length = width
    width = temp / 2
    print(f"A1: {int(length)} mm x {int(width)} mm")

# Calculate and print A2
if n >= 2:
    temp = length
    length = width
```

```c
printf("A1: %.0lf mm x %.0lf mm\n", length, width);
    }

    // Calculate and print A2
    if (n >= 2) {
        double temp = length;
        length = width;
        width = temp / 2;
printf("A2: %.0lf mm x %.0lf mm\n", length, width);
    }

    // Calculate and print A3
    if (n >= 3) {
        double temp = length;
        length = width;
        width = temp / 2;
printf("A3: %.0lf mm x %.0lf mm\n", length, width);
    }
    return 0;
}
```

```java
        System.out.printf("A1: %.0f mm x %.0f mm\n", length, width);
    }

    // Calculate and print A2
    if (n >= 2) {
        double temp = length;
        length = width;
        width = temp / 2;
        System.out.printf("A2: %.0f mm x %.0f mm\n", length, width);
    }

    // Calculate and print A3
    if (n >= 3) {
        double temp = length;
        length = width;
        width = temp / 2;
        System.out.printf("A3: %.0f mm x %.0f mm\n", length, width);
    }
  }
}
```

```python
    width = temp / 2
    print(f"A2: {int(length)} mm x {int(width)} mm")

# Calculate and print A3
if n >= 3:
    temp = length
    length = width
    width = temp / 2
    print(f"A3: {int(length)} mm x {int(width)} mm")
```

Time Complexity

Since there are no loops or functions, the number of operations is directly proportional to nnn (the number of paper sizes).

Time Complexity: O(n).

Space Complexity

Only a few variables are used for calculations.

Space Complexity: O(1).

If a four digit number is input, write a c, java and python program to calculate the sum of its digits

C

```c
#include <stdio.h>

int main() {
    int num, sum = 0;
    printf("Enter a four-digit number: ");
    scanf("%d", &num);

    if (num < 1000 || num > 9999) {
        printf("Please enter a valid four-digit number.\n");
        return 1;
    }

    // Extract and sum digits
    sum += num % 10;          // Extract last digit
    num /= 10;                // Remove last digit
    sum += num % 10;          // Extract next digit
    num /= 10;                // Remove last digit
    sum += num % 10;          // Extract next digit
    num /= 10;                // Remove last digit
```

JAVA

```java
import java.util.Scanner;

public class DigitSum {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a four-digit number: ");
        int num = scanner.nextInt();
        scanner.close();

        if (num < 1000 || num > 9999) {
            System.out.println("Please enter a valid four-digit
number.");
            return;
        }

        int sum = 0;
        // Extract and sum digits
        sum += num % 10;          // Extract last digit
        num /= 10;                // Remove last digit
```

PYTHON

```python
num = int(input("Enter a four-digit number: "))

if num < 1000 or num > 9999:
    print("Please enter a valid four-digit number.")
else:
    # Extract and sum digits
    sum_digits = 0
    sum_digits += num % 10          # Extract last digit
    num //= 10                      # Remove last digit
    sum_digits += num % 10          # Extract next digit
    num //= 10                      # Remove last digit
    sum_digits += num % 10          # Extract next digit
    num //= 10                       # Remove last digit
    sum_digits += num % 10      # Extract last remaining digit

    print("Sum of the digits:", sum_digits)
```

```
    sum += num % 10;            // Extract last remaining digit

    printf("Sum of the digits: %d\n", sum);

    return 0;
}
```

```
        sum += num % 10;                // Extract next digit
        num /= 10;                      // Remove last digit
        sum += num % 10;                // Extract next digit
        num /= 10;                      // Remove last digit
        sum += num % 10;                // Extract last remaining digit

        System.out.println("Sum of the digits: " + sum);
    }
}
```

Time Complexity

- The program involves a fixed number of operations (4 modulo and division operations for a four-digit number).
- **Time Complexity**: **O(1),** as the number of operations does not depend on the input size.

Space Complexity

- Only a few variables are used (e.g., num, sum, sum_digits).
- No additional memory is allocated for data structures.
- **Space Complexity: O(1).**

If a four digit number is input, write a c, java and python program to reverse the number

C

```
#include <stdio.h>

int main() {
    int num, reversed = 0;

    printf("Enter a four-digit number: ");
    scanf("%d", &num);

    if (num < 1000 || num > 9999) {
        printf("Please enter a valid four-digit number.\n");
        return 1;
    }

    // Reverse the number
    reversed += num % 10;  // Extract last digit
    num /= 10;             // Remove last digit
    reversed = reversed * 10 + num % 10;  // Add next digit
    num /= 10;             // Remove last digit
    reversed = reversed * 10 + num % 10;  // Add next digit
    num /= 10;             // Remove last digit
    reversed = reversed * 10 + num % 10;
// Add last remaining digit

    printf("Reversed number: %d\n", reversed);

    return 0;
}
```

JAVA

```
import java.util.Scanner;

public class ReverseNumber {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a four-digit number: ");
        int num = scanner.nextInt();
        scanner.close();

        if (num < 1000 || num > 9999) {
      System.out.println("Please enter a valid four-digit number.");
            return;
        }

        int reversed = 0;

        // Reverse the number
        reversed += num % 10;  // Extract last digit
        num /= 10;             // Remove last digit
        reversed = reversed * 10 + num % 10;  // Add next digit
        num /= 10;             // Remove last digit
        reversed = reversed * 10 + num % 10;  // Add next digit
        num /= 10;             // Remove last digit
        reversed = reversed * 10 + num % 10;
// Add last remaining digit

        System.out.println("Reversed number: " + reversed);
    }}
```

PYTHON

```
num = int(input("Enter a four-digit number: "))

if num < 1000 or num > 9999:
    print("Please enter a valid four-digit number.")
else:
    # Reverse the number
    reversed_num = 0
    reversed_num += num % 10  # Extract last digit
    num //= 10                # Remove last digit
    reversed_num = reversed_num * 10 + num % 10
# Add next digit
    num //= 10                # Remove last digit
    reversed_num = reversed_num * 10 + num % 10
# Add next digit
    num //= 10                # Remove last digit
    reversed_num = reversed_num * 10 + num % 10
# Add last remaining digit

    print("Reversed number:", reversed_num)
```

Time Complexity

- Each program performs a fixed number of steps (4 modulo and division operations for the four digits).
- **Time Complexity: O(1),** as the number of steps does not depend on the input size.

Space Complexity

- The programs use a constant number of variables (num, reversed_num).
- No additional data structures are required.
- **Space Complexity: O(1).**

if four digit number is input, write a c, java and python program to obtain the sum of first and last digit of the given number.

C

```c
#include <stdio.h>

int main() {
    int num, first_digit, last_digit, sum;

    printf("Enter a four-digit number: ");
    scanf("%d", &num);

    if (num < 1000 || num > 9999) {
        printf("Please enter a valid four-digit number.\n");
        return 1;
    }

    // Find the last digit (using modulo)
    last_digit = num % 10;

    // Find the first digit (using division)
    first_digit = num / 1000;

    // Calculate the sum of first and last digits
    sum = first_digit + last_digit;

    printf("Sum of the first and last digit: %d\n", sum);

    return 0;
}
```

JAVA

```java
import java.util.Scanner;

public class SumFirstLastDigit {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a four-digit number: ");
        int num = scanner.nextInt();
        scanner.close();

        if (num < 1000 || num > 9999) {
            System.out.println("Please enter a valid four-digit
number.");
            return;
        }

        // Find the last digit (using modulo)
        int lastDigit = num % 10;

        // Find the first digit (using division)
        int firstDigit = num / 1000;

        // Calculate the sum of first and last digits
        int sum = firstDigit + lastDigit;

        System.out.println("Sum of the first and last digit: " + sum);
    }
}
```

PYTHON

```python
num = int(input("Enter a four-digit number: "))

if num < 1000 or num > 9999:
    print("Please enter a valid four-digit number.")
else:
    # Find the last digit (using modulo)
    last_digit = num % 10

    # Find the first digit (using division)
    first_digit = num // 1000

    # Calculate the sum of first and last digits
    sum_digits = first_digit + last_digit

    print("Sum of the first and last digit:", sum_digits)
```

Time Complexity

- The program performs a constant number of operations:
    1. One modulo operation to find the last digit.
    2. One division operation to find the first digit.
- Simple arithmetic to calculate the sum. **Time Complexity: O(1),** as the operations do not depend on the input size.

Space Complexity

- Only a fixed number of variables are used (num, first_digit, last_digit, sum_digits). **Space Complexity: O(1).**