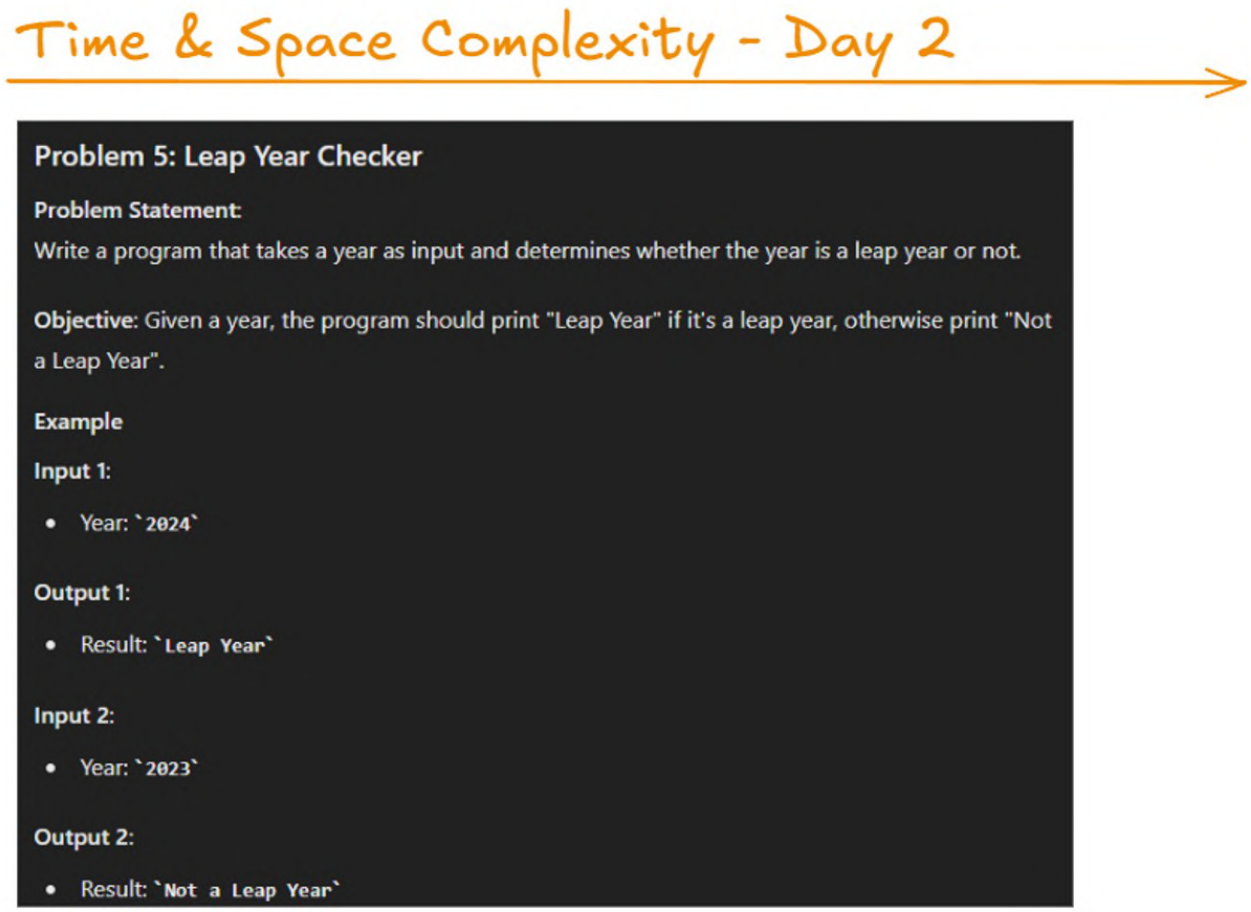
Time & Space Complexity - Day 2

03 September 2024 15:54




```
import java.util.Scanner;
public class LeapYearChecker {
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        System.out.println("Enter a Year:");
        int year = scn.nextInt();
        if(year % 4 ==0){
            if(year % 100 ==0){
                if(year % 400 ==0){
                    System.out.println("Leap Year");
                }else{
                    System.out.println("Not a Leap Year");
                }
            }else{
                System.out.println("Leap Year");
            }
        }else{
            System.out.println("Not A Leap Year");
```

```
        }
    }else{
        System.out.println("Not A Leap Year");
    }
    }
}
```

Problem 6: Is a Number Prime?

Problem Statement:

Write a program that takes an integer as input and determines whether the number is prime.

Objective: Given an integer, the program should print "Prime" if the number is a prime number, otherwise print "Not Prime".

Example

Input 1:

- Number: `7`

Output 1:

- Result: `Prime`

Input 2:

- Number: `10`

Output 2:

- Result: `Not Prime`

```
import java.util.Scanner;
public class PrimeChecker {
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int num = scn.nextInt();
        boolean isPrime = true;
        if(num <=1){
            isPrime = false;
        }else{
            for(int i = 2; i <= Math.sqrt(num); i++){
                if(num % i ==0){
                    isPrime = false;
                    return;
                }
            }
        }

        if(isPrime){
            System.out.println(num + " is a prime number.");
        }else{
            System.out.println(num + " is not a prime number.");
```

```
            System.out.println(num + " is a prime number.");
        }else{
            System.out.println(num + " is not a prime number.");
        }
    }
}
```

Problem 7: Factorial of a Number

Problem Statement:

Write a program that takes an integer as input and calculates its factorial.

Objective: Given an integer, the program should print the factorial of that number.

Example

Input 1:

- Number: `5`

Output 1:

- Result: `120` (since $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$)

Input 2:

- Number: `3`

Output 2:

- Result: `6` (since $3! = 3 \times 2 \times 1 = 6$)

```
import java.util.Scanner;
public class Factorial {
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);

        System.out.println("Enter an number:");
        int num = scn.nextInt();
        int fact = 1;

        if(num ==0 || num == 1){
            fact = 1;
         }else{
            for(int i = 1 ; i<= num ; i++){
                fact *= i;
            }
        }
        System.out.println("The Factorial of " +
            num + " is: " + fact);
    }
```

```
            num + " is: " + fact);
    }
}
```

Problem 8: Print Multiplication Table

Problem Statement:

Write a program that takes an integer as input and prints the multiplication table for that number up to 10.

Objective: Given an integer, the program should print its multiplication table up to 10.

Example

Input 1:

- Number: `7`

Output 1:

```
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
...
7 x 10 = 70
```

Input 2:

- Number: `3`

Output 2:

```
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
...
3 x 10 = 30
```

```java
import java.util.Scanner;
public class MultiplicationTable {
    public static void main(String[] args) {

        Scanner scn = new Scanner(System.in);
        System.out.println("Enter a number");
        int  num = scn.nextInt();

        for(int i = 1; i<=10 ; i++){
            System.out.println(num + " x " + i + " = " + (num * i));
        }
    }
}
```

```
Enter a number
11
11 x 1 = 11
11 x 2 = 22
11 x 3 = 33
11 x 4 = 44
11 x 5 = 55
11 x 6 = 66
11 x 7 = 77
11 x 8 = 88
```

```
11 x 6 = 66
11 x 7 = 77
11 x 8 = 88
11 x 9 = 99
11 x 10 = 110
```

Problem 9: Sum of Natural Numbers

Problem Statement:

Write a program that takes an integer `n` as input and calculates the sum of the first `n` natural numbers.

Objective: Given an integer `n`, the program should print the sum of the first `n` natural numbers.

Example

Input 1:

- `n = 5`

Output 1:

- Result: `15` (since $1 + 2 + 3 + 4 + 5 = 15$)

Input 2:

- `n = 10`

Output 2:

- Result: `55` (since $1 + 2 + \ldots + 10 = 55$)

```java
import java.util.Scanner;
public class SumOfNaturalNumber {
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        System.out.println("Enter a number:");
        int n = scn.nextInt();

        int sum =0;
        for(int i = 1; i<=n ; i++){
            sum +=i;
        }
        System.out.println(sum);
    }
}
```

Problem 10: Check If Last Digit Is Even

Problem Statement:

Write a program that takes an integer as input and checks whether the last digit of the number is even.

Objective: Given an integer, the program should print "Even" if the last digit is even, otherwise print "Odd".

Objective: Given an integer, the program should print "Even" if the last digit is even, otherwise print "Odd".

Example

Input 1:

- Number: `1234`

Output 1:

- Result: `Even`

Input 2:

- Number: `567`

Output 2:

- Result: `Odd`

```
import java.util.Scanner;
public class LastDigitChecker {
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);

        System.out.println("Enter a number");
        int num = scn.nextInt();

        int lastDigit = num % 10;

        if(lastDigit %2 == 0){
            System.out.println("The last digit is even");
        }else{
            System.out.println("The last digit is odd");
        }
    }
}
```

Time & Space Complexity

```
int lastDigit = num % 10;

if(lastDigit %2 == 0){
    System.out.println("The last digit is even");
}else{
```

```
if(lastDigit %2 == 0){
    System.out.println("The last digit is even");
}else{
    System.out.println("The last digit is odd");
}
```

Time Complexity - O(1)

```
import java.util.Scanner;
public class SumOfNaturalNumber {
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        System.out.println("Enter a number:");
        int n = scn.nextInt();

        int sum =0;
        for(int i = 1; i<=n ; i++){
            sum +=i;
        }
        System.out.println(sum);
    }
}
```

Time Complexity - O(N)

```
public static void main(String[] args) {
    int n =5;
    for(int i = 1 ; i<=n ; i++){
        System.out.println(i);
    }
}
```

Time Complexity - O(N)

```
int n =5;
for(int i = 1 ; i<=n ; i++){
    for(int j = 1; j<=n; j++){
        System.out.println(i + " ," + j);
    }
}
```

Time Complexity - O(N^2)

```
}
```

Time Complexity - O(N^2)

```
int n =16;
int x = 1;
while(x < n){
    x = x * 2;
    System.out.println(x);
}
```

Time Complexity - O(log N)

```
int n =16;
int x = 1;
for(int i = 1 ; i<=n ; i++){
    while(x < n){
        x = x * 2;
        System.out.println(x);
    }
}
```

O(N)

O(logN)

O(N) * O(LogN)

Time Complexity - O(Nlog N)

```
int n =16;
int x = 1;
for(int i = 1 ; i<=n ; i++){
    while(x < n){
        x = x * 2;
        System.out.println(x);
    }
}
```

O(NLogN)

```
for(int j = 1; j<=n j++){
    System.out.println(x:"Hello");
}
```

O(N)

O(NLogN) + O(N)

Time Complexity - O(Nlog N)

```
int n =16;
int x = 1;
for(int i = 1 ; i<=n ; i++){
    System.err.println(x);
}

for(int j = 1; j<=n j++){
    System.out.println(x:"Hello");
}
```

O(N)

O(N)

O(N) + O(N) = O(2N)

Time Complexity - O(N)

Problem Statement:

Write a program that prints a right-angled triangle of stars with `n` rows.

Example:

- Input: `n = 4`
- Output:

```markdown
*
**
```

```
*
**
***
****
```

```java
int n = 4;
for(int i = 1 ; i<=n ; i++){
    for(int j =1; j<=i ; j++){
        System.out.print("*");
    }
    System.out.println();
}
```

rows

columns

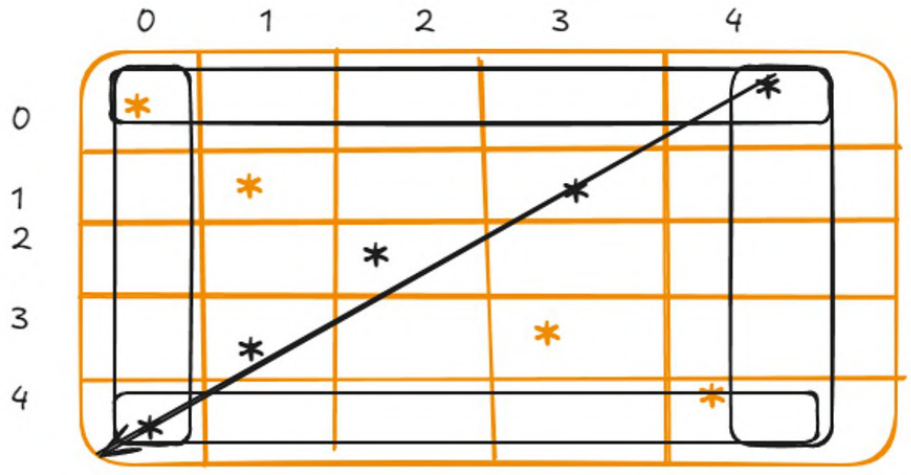
2

```
*
* *
* * *
* * * *
```

memory

n = 4;
i = 5;
j=1;

Time Complexity - O(N^2)
Space Complexity - O(1)



n*n

n= 5

i+j = n-1

What is Space Complexity?

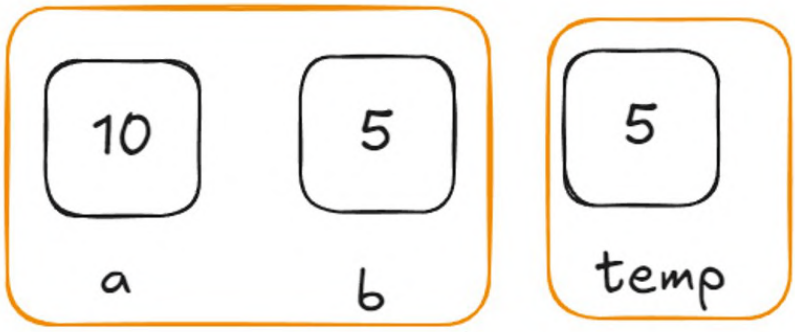
Space complexity refers to the amount of memory an algorithm uses relative to the input size. It measures the total space or memory required by an algorithm, including both the space needed for the input and any auxiliary space used during computation.

Space complexity is important because it helps us understand how much memory an algorithm will need, which is crucial in environments with limited resources.

Write a program to Swap Numbers.

```
a = 5;
b = 10;
```

```java
public static void main(String[] args)
    int a =5;
    int b =10;
    int temp = a;
    a = b;
    b = temp;
    System.out.println(a);
    System.out.println(b);
}
```

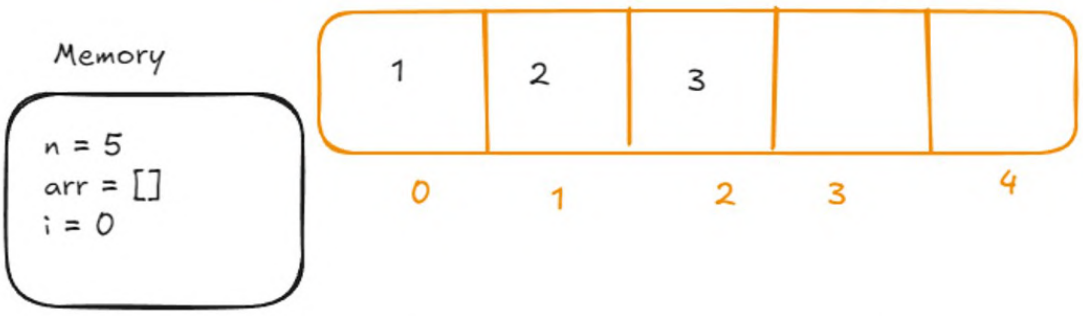


Time - O(1)
Space - O(1)

Space - O(1)

```
public class StoreNumbers {
    public static void main(String[] args) {
        int n = 5;
        int[] array = new int[n];
        for (int i = 0; i < n; i++) {
            array[i] = i + 1;
        }
        // Printing the array to verify
        for (int i = 0; i < n; i++) {
            System.out.print(array[i] + " ");
        }
    }
}
```

O(N)



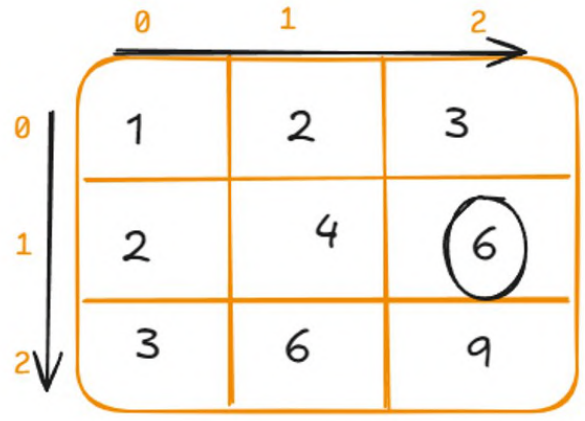
Time Complexity = O(N)
Space Complexity = O(N)

```
public class CreateMatrix {
    public static void main(String[] args) {
        int n = 3;
        int[][] matrix = new int[n][n];
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                matrix[i][j] = (i + 1) * (j + 1);
            }
        }
        // Printing the matrix to verify
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

O(N^2)

O(N^2)

```
}
```

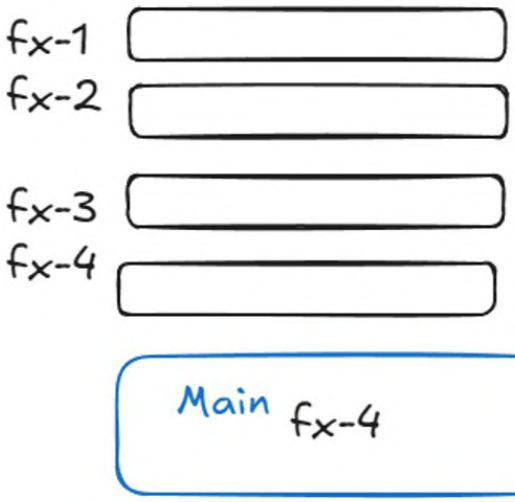


Space - O(N^2)

Time is 2 * O(N^2) where 2 is constant can be neglacted.

```
public class LogarithmicSpace {
    public static void main(String[] args) {
        int x = 16;
        divideAndConquer(x);
    }

    public static void divideAndConquer(int x) {
        if (x == 1) {
            return;
        } else {
            divideAndConquer(x / 2);
        }
    }
}
```



call Stack

a b c

_ _ _

2 * 2 * 2 = 8

2^3 = 8

length of the string is 3.

[.]
[a]
[a b]
[a b c]
[a c]
[b c]
[b]
[c]

= 8

Attendance Code: 0221C22D