**Name:** John Patrick F. Narvasa

**Section/Schedule:** 1 BSCS-2 / 4PM-7PM

**Program:** BS Computer Science

**Course:** Data Structures and Algorithms

**Problem:**

### Part 2: Code and determine the Time complexity of your solution(code).

Sample Output:

// O(1) - Constant Time

public static int getFirstElement(int[] array) {

return array[0];

}

| **Source Code:**  public class SquareAndCubeTable {  public static void main(String[] args) {  // the value of the range for the loop  int num = 10;  printTable(num);    }  // prints the table  // Overall: O(n) - Linear Time Complexity  public static void printTable(int n) {  int squareSum=0;  int cubeSum = 0;  System.out.println("Number " + "\tSquare" + "\tCube");  // O(n) - Linear Time Complexity  for (int i = 2; i <= n; i+=2) {  // O(1) - Constant Time Complexity  int square = i \* i;  int cube = i \* i \* i;  System.out.println(i + "\t" + square + "\t" + cube + "\n");  squareSum += square;  cubeSum += cube;  }  System.out.println("Total \t" + squareSum + "\t" + cubeSum);  }  } |
| --- |
| **Screenshot:** |

**Function:** printTable(int n)

**Time Complexity:** O(n)

**Reasoning:**

TheprintTable method used a simple loop. Firstly, it initializes what the value of the square and cube variables will be which performs a constant operation (O(1)). To print the following outputs, the loop iterates around the value that is given by num variable which performs a linear operation (O(n)). Although, the amount of steps the program made is not directly proportional to the input value from num since the loop increments by 2 (which is denoted by i +=2), we can say that the overall time complexity of the program is approximately O(n).