SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

AN AUTONOMOUS INSTITUTION

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

Department of Electronics and Communication Engineering

23CSB101-Object Oriented Programming

Assignment I

1. Given an array of integer's nums and an integer target, return indices of the two numbers such that they add up to target.You may assume that each input would have exactly one solution, and you may not use the same element twice. You can return the answer in any order.

   Input: nums = [2,7,11,15], target = 9

   Output: [0,1]

   Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].

2. Given an integer array nums sorted in non-decreasing order, remove the duplicates in-place such that each unique element appears only once. The relative order of the elements should be kept the same. Then return the number of unique elements in nums. Consider the number of unique elements of nums to be k, to get accepted, you need to do the following things: Change the array nums such that the first k elements of nums contain the unique elements in the order they were present in nums initially. The remaining elements of nums are not important as well as the size of nums. Return k

   Input: nums = [0,0,1,1,1,2,2,3,3,4]

   Output: 5, nums = [0,1,2,3,4,_,_,_,_,_]

   Explanation: Your function should return k = 2, with the first two elements of nums being 1 and 2 respectively.

   It does not matter what you leave beyond the returned k (hence they are underscores).

3. Given an integer num, repeatedly add all its digits until the result has only one digit, and return it.

Input: num = 38
Output: 2

Explanation: The process is
38 --> 3 + 8 --> 11
11 --> 1 + 1 --> 2
Since 2 has only one digit, return it.

4. You are given a positive integer array nums. The element sum is the sum of all the elements in nums. The digit sum is the sum of all the digits (not necessarily distinct) that appear in nums. Return the absolute difference between the element sum and digit sum of nums.
Input: nums = [1,15,6,3]
Output: 9

Explanation:
The element sum of nums is 1 + 15 + 6 + 3 = 25.
The digit sum of nums is 1 + 1 + 5 + 6 + 3 = 16.
The absolute difference between the element sum and digit sum is |25 - 16| = 9.

5. Given a positive integer n, find the sum of all integers in the range [1, n] inclusive that are divisible by 3, 5, or 7. Return an integer denoting the sum of all numbers in the given range satisfying the constraint.

Input: n = 7
Output: 21

Explanation: Numbers in the range [1, 7] that are divisible by 3, 5, or 7 are 3, 5, 6, 7.
The sum of these numbers is 21.

6. Given an integer number n, return the difference between the product of its digits and the sum of its digits.

Input: n = 234
Output: 15

Explanation:
Product of digits = 2 * 3 * 4 = 24
Sum of digits = 2 + 3 + 4 = 9
Result = 24 - 9 = 15

7. You are given a large integer represented as an integer array digit, where each digits[i] is the ith digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. The large integer does not contain any leading 0's. Increment the large integer by one and return the resulting array of digits.

   Input: digits = [1,2,3]
   Output: [1,2,4]

   Explanation: The array represents the integer 123.
   Incrementing by one gives 123 + 1 = 124.
   Thus, the result should be [1,2,4].

8. Given an integer x, return true if x is a palindrome, and false otherwise.

   Input: x = 121
   Output: true

   Explanation: 121 reads as 121 from left to right and from right to left.

9. Given a non-negative integer x, return the square root of x rounded down to the nearest integer. The returned integer should be non-negative as well. You must not use any built-in exponent function or operator.

   Input: x = 4
   Output: 2

   Explanation: The square root of 4 is 2, so we return 2.

10. Design a parking system for a parking lot. The parking lot has three kinds of parking spaces: big, medium, and small, with a fixed number of slots for each size. Implement the ParkingSystem class: ParkingSystem(int big, int medium, int small) Initializes object of the ParkingSystem class. The number of slots for each parking space are given as part of the constructor. bool addCar(int carType) Checks whether there is a parking space of carType for the car that wants to get into the parking lot. carType can be of three kinds: big, medium, or small, which are represented by 1, 2, and 3 respectively. A car can only park in a parking space of its carType. If there is no space available, return false, else park the car in that size space and return true.

    Input

["ParkingSystem", "addCar", "addCar", "addCar", "addCar"]

[[1, 1, 0], [1], [2], [3], [1]]

Output

[null, true, true, false, false]

11. You are building a library management system. A Book class has a method checkAvailability(), but you need different versions of this method to accommodate different types of books, such as PrintedBook and Ebook.

    Problem:

    Write a class Book with two overloaded methods:
    - checkAvailability(int bookId): Checks availability of a printed book based on its bookId.
    - checkAvailability(String bookTitle): Checks availability of an ebook based on its bookTitle.

    Implement the class and demonstrate method overloading by calling both methods with different inputs.

    Constraints:

    - Assume there is a predefined database of books in the library.

12. You are building a salary management system for a company. You need to create a method that calculates the bonus of an employee, and the bonus depends on the department they belong to. The department is an object passed as a parameter.

Problem:

- Create two classes: Employee and Department.
- Employee class should have an id, name, and salary.
- Department class should have departmentName and bonusMultiplier.
- Implement a method calculateBonus(Department department) in the Employee class that calculates the bonus by multiplying the employee's salary by the department's bonus multiplier.

Constraints:

- Bonus calculation: employeeSalary * departmentBonusMultiplier.

13. You are managing a fleet of vehicles for a transportation company. The system should be able to return a specific vehicle type based on its ID. This will help retrieve either a Car or Truck object.

Problem:

- Create a FleetManager class that contains a method getVehicleById(int vehicleId) which returns a Vehicle object.
- Define two classes, Car and Truck, that both extend from a common Vehicle superclass.
- Implement the getVehicleById method in FleetManager to return either a Car or Truck based on the given vehicleId.

Constraints:

- The system should differentiate between Car and Truck types and should use object polymorphism to return the appropriate vehicle object.

14. You are building a school system where different grades are represented by nested classes. You need to store students and their grades, with some classes being static and others being inner classes for flexibility.

Problem:

- Create a School class that contains a static class Grade, which represents a grade level (e.g., Grade 1, Grade 2).
- Inside Grade, create an inner class Student that holds student data such as name and marks.
- The School class should contain a method to add students to a specific grade.

Implement this system to manage students across various grade levels in the school.

Constraints:

- The Grade class should be static, and the Student class should be an inner class of Grade.

15. You are designing a system to model an animal kingdom. The system should allow different animal types to speak in their own unique way. The Animal class will define a basic method speak(), but different animal types, like Dog and Cat, should override this method to make specific sounds.

Problem:

- Create a base class Animal with a method speak() that prints a generic message like "The animal makes a sound."
- Create two subclasses, Dog and Cat, that override the speak() method.
- In the Dog class, use the super keyword to call the base class speak() method before adding the dog-specific sound.
- Demonstrate polymorphism by calling the speak() method for both Dog and Cat objects.

Constraints:

- speak() should be overridden in both Dog and Cat.
- The super keyword should be used in the Dog class.