National Institute of Technology Kurukshetra
Department of Computer Engineering
Object Oriented Programming using Java (CSPC – 205)
End Term Practical Examination, November 2024
SET - 1

Time : 2 hours **Max Marks : 20**

---

1) Write a program to accept the year of graduation from school as an integer value from the user. Using the Binary Search technique on the sorted array of integers given below, output the message **"Record exists"** if the value input is located in the array. If not, output the message **"Record does not exist"**.
array = {1982, 1986, 1987, 1993, 1996, 1999, 2003, 2006, 2007, 2009, 2010, }

2) Create a class Book with fields **title, author, price,** and **quantity.**

- Add two constructors:
  - For initializing all fields.
  - For initializing only title and author, with default values for price and quantity.
- Create a Bookstore class with a method **sellBook(String title, int quantity)** that:
  - Reduces the quantity of a book.
  - Throws a custom exception **OutOfStockException** if the requested quantity is unavailable.
- Use method overloading in **sellBook()** to allow selling by title only (default quantity is 1).
- Write a main method to add books to the store and demonstrate selling books with and without exceptions.

National Institute of Technology Kurukshetra
Department of Computer Engineering
Object Oriented Programming using Java (CSPC – 205)
End Term Practical Examination, November 2024
SET - 2

Time : 2 hours **Max Marks : 20**

---

1) Write a program to initialize the seven wonders of the world along with their locations in two different arrays. Search for a name of the country input by the user. If found, display the name of the country along with its wonder, otherwise display "Sorry Not Found".

Name	Location
Chichén Itzá	Mexico
Christ The Redeemer	Brazil
Taj Mahal	India
Great Wall of China	China
Machu Picchu	Peru
Petra	Jordan
Colosseum	Italy

2) Create an abstract class **Vehicle** with fields **brand** and **model** and methods **getServiceCost()** and **getDetails().**

- Define an interface **Insurable** with a method **calculateInsurance()**.
- Create subclasses **Car** and **Bike** that implement **getServiceCost()** with different logic.
- Implement the Insurable interface to calculate insurance based on vehicle type.
- Demonstrate polymorphism by creating Vehicle references for Car and Bike and calling their methods.

National Institute of Technology Kurukshetra
Department of Computer Engineering
Object Oriented Programming using Java (CSPC – 205)
End Term Practical Examination, November 2024
SET - 3

Time : 2 hours **Max Marks : 20**

---

1) Write a program to identify a special two-digit number. A special two-digit number is such that when the sum of its digits is added to the product of its digits, the result is equal to the original two-digit number.

Example:

Consider the number 59.

Sum of digits : 5 + 9 = 14

Product of its digits = 5 x 9 = 45

Sum of the sum of digits and product of digits = 14 + 45 = 59

2) Create a package ecommerce containing a class **Product** with fields **id, name, price,** and **quantity**.

Create a subclass **Electronics** that adds a field **warranty**. Override the **toString()** method to display product details.

In the main program, **add products to a cart** represented by a String[].

Use StringBuilder to concatenate and display the cart items.

Demonstrate method overriding by calling toString() on both Product and Electronics objects.

National Institute of Technology Kurukshetra
Department of Computer Engineering
Object Oriented Programming using Java (CSPC – 205)
End Term Practical Examination, November 2024
SET - 4

Time : 2 hours **Max Marks : 20**

---

1) Given an integer array **nums**, find the subarray with the largest **sum**, and return *its sum*.

Example 1:
Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
Explanation: The subarray [4,-1,2,1] has the largest sum 6.
Example 3:
Input: nums = [5,4,-1,7,8]
Output: 23
Explanation: The subarray [5,4,-1,7,8] has the largest sum 23.

2) Create a class **BankAccount** with fields **accountNumber**, **balance**, and **accountHolder**.

Add methods **deposit(double amount)** and **withdraw(double amount)** that:

- Throw a custom exception **InsufficientFundsException** if the withdrawal amount exceeds the balance.

Use polymorphism to override the **withdraw()** method in a subclass **SavingsAccount** to allow only 3 withdrawals per month.

Parse user input like **"Deposit: 500"** to extract the action and amount using **String** methods and **Double** wrapper class.

National Institute of Technology Kurukshetra
Department of Computer Engineering
Object Oriented Programming using Java (CSPC – 205)
End Term Practical Examination, November 2024
SET - 5

Time : 2 hours **Max Marks : 20**

---

1) You are given an integer array **nums**. You replace each element in nums with the **sum of its digits**. Return the **minimum element** in nums after all replacements.

Example 1:
Input: nums = [10,12,13,14]
Output: 1

Example 2:
Input: nums = [999,19,199]
Output: 10

Constraints:
1 <= nums.length <= 100
1 <= nums[i] <= 10^4

2) Design a class named Fan.

- The class contains: Three constants named **SLOW, MEDIUM,** and **FAST** with values **1, 2,** and **3** to denote the fan speed.
- An int data field named **speed** that specifies the speed of the fan (default SLOW).
- A boolean data field named **on** that specifies whether the fan is on (default false).
- A double data field named **radius** that specifies the radius of the fan (default 5).
- A string data field named **color** that specifies the color of the fan (default blue).
- A **no-arg constructor** that creates a default fan.
- The **accessor** and **mutator** methods for all four data fields.
- Override toString method that returns a string description for the fan.
- If the fan is on, the method returns the fan speed, color, and radius in one combined string.
- If the fan is not on, the method returns fan color and radius along with the string "fan is off" in one combined string.

National Institute of Technology Kurukshetra
Department of Computer Engineering
Object Oriented Programming using Java (CSPC – 205)
End Term Practical Examination, November 2024
SET - 6

Time : 2 hours **Max Marks : 20**

---

1) Given an array **arr** of positive integers sorted in a strictly increasing order, and an integer **k**. Return the **kth positive integer** that is missing from this array.

Example 1:

Input: arr = [2,3,4,7,11], k = 5

Output: 9

Explanation: The missing positive integers are [1,5,6,8,9,10,12,13,...]. The 5th missing positive integer is 9.

Example 2:

Input: arr = [1,2,3,4], k = 2

Output: 6

Explanation: The missing positive integers are [5,6,7,...]. The 2nd missing positive integer is 6.

2) Create an interface **Result** with methods **calculateGrade()** and **displayDetails()**. Create a class **Student** that implements Result with fields **name, rollNumber,** and **marks[]**.

Add two constructors:

- One for initializing name and rollNumber only.
- Another for initializing all fields.

Implement **calculateGrade()** to calculate grades based on average marks.

If marks are not available, throw a custom exception **MarksNotAvailableException**. In **displayDetails()**, print the details of students in an organized format.

National Institute of Technology Kurukshetra
Department of Computer Engineering
Object Oriented Programming using Java (CSPC – 205)
End Term Practical Examination, November 2024
SET - 7

Time : 2 hours **Max Marks : 20**

---

1) Write a Java program that prints all possible outcomes when a coin is tossed n times. Each toss can result in either "H" (Heads) or "T" (Tails). The program should use recursion to generate all possible combinations of outcomes. The order of outcomes in the output does not matter.

Input:

An integer n representing the number of times the coin is tossed.

Output:

All possible combinations of outcomes, each combination on a new line.

2) Create an abstract class **Person** with fields **name** and **age**, and an **abstract method getRole()**. Create subclasses **Doctor** and **Patient** with additional fields **(specialization for Doctor and disease for Patient)**.

Implement an interface **Billable** in Patient with a method **generateBill()**, which calculates charges based on treatment type using predefined rates:

- Rs.500 for consultation
- Rs.15,000 for surgery.
- Rs.3,000 for therapy
- Rs.1,000 for others.

Use constructor chaining to initialize all fields.

National Institute of Technology Kurukshetra
Department of Computer Engineering
Object Oriented Programming using Java (CSPC – 205)
End Term Practical Examination, November 2024
SET - 8

Time : 2 hours **Max Marks : 20**

---

1) You are given an array prices where prices[i] is the price of a given stock on the i^{th} day. You want to maximize your profit by choosing a **single day** to buy one stock and choosing a **different day in the future** to sell that stock.
Return *the maximum profit you can achieve from this transaction*. If you cannot achieve any profit, return 0.
Example 1:
Input: prices = [7,1,5,3,6,4]
Output: 5
Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6 - 1 = 5.
Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.
Example 2:
Input: prices = [7,6,4,3,1]
Output: 0
Explanation: In this case, no transactions are done and the max profit = 0.

2) Create a class **Taxi** with fields **registrationNumber, driverName,** and **farePerKm.**
Add a constructor to initialize all fields.
Write a method **bookTaxi(int distance)** to calculate the fare.
Throw a custom exception **InvalidDistanceException** if the distance is less than or equal to zero.
Use polymorphism to override the **bookTaxi()** method in a subclass **LuxuryTaxi** to include additional charges.
Parse a **String** input like **"Luxury Taxi: 15"** to determine the type of taxi and distance using String methods and the Integer wrapper class.

National Institute of Technology Kurukshetra
Department of Computer Engineering
Object Oriented Programming using Java (CSPC – 205)
End Term Practical Examination, November 2024
SET - 9

Time : 2 hours **Max Marks : 20**

---

1) The **next greater element** of some element x in an array is the **first greater** element that is **to the right** of x in the same array.

You are given two **distinct 0-indexed** integer arrays nums1 and nums2, where nums1 is a subset of nums2.

For each 0 <= i < nums1.length, find the index j such that nums1[i] == nums2[j] and determine the **next greater element** of nums2[j] in nums2. If there is no next greater element, then the answer for this query is -1.

Return *an array* ans *of length* nums1.length *such that* ans[i] *is the* ***next greater element*** *as described above.*

Example 1:
Input: nums1 = [4,1,2], nums2 = [1,3,4,2]
Output: [-1,3,-1]
Explanation: The next greater element for each value of nums1 is as follows:

- 4 is underlined in nums2 = [1,3,4,2]. There is no next greater element, so the answer is -1.
- 1 is underlined in nums2 = [1,3,4,2]. The next greater element is 3.
- 2 is underlined in nums2 = [1,3,4,2]. There is no next greater element, so the answer is -1.

2) Create a class **Employee** with fields:

- **name** (String), **id** (String), **baseSalary** (double).

- Add a constructor to initialize fields.
  - Validate that **baseSalary > 0**; otherwise, throw an **IllegalArgumentException**: **"Base salary must be greater than zero!"**
- Implement methods:
  - **calculateBonus(double percentage)**
    - Return bonus (baseSalary * percentage/100).
    - Throw an **IllegalArgumentException** for invalid percentage (<0 or >100).
  - **getTotalSalary()**
    - Return baseSalary + bonus.
- In the main method:
  - Create an Employee with valid/invalid base salary.
  - Calculate bonus for valid/invalid percentages.