Branching

After working for a year finally Karim got his salary and bonus for Eid 2025, now he decides to do some kind of shopping for his family. His salary is x Tk and he got 75% bonus of his salary. He decides to spend money on shopping precisely so that everyone gets what they want. He planned to distribute 45% of his total money for his wife and 45% for his son. The remaining 10% will be saved for emergency cases. His wife wants a new laptop that costs around 90,000 Tk and his son wants a PS5 that costs about 70,000 Tk with games and an extra controller to play with friends that costs about 7,000 Tk. Now, take x as input from the user and see if Karim can fully fulfill his families wish, if not possible then see whether he can use remaining budget from son and wife to fulfill each other's wish. The saving part should not be touched by any means.

Sample Input	Sample Output
Enter Karim's Salary (x): 110000	Amount with Bonus: 192500.00 Wife Budget: 86625.00 Son Budget: 86625.00 Savings: 19250.00 Shopping Done for Wife with remaining money from Sons budget! Shopping Done for Son!
Enter Karim's Salary (x): 100000	Amount with Bonus: 175000.00 Wife Budget: 78750.00 Son Budget: 78750.00 Savings: 17500.00 Cannot do shopping for wife even with remaining money from his son's budget! Shopping Done for Son!

Linear Loop

A group of astronauts established a space colony on Mars in 1999. Due to the harsh conditions, they must import essential supplies from Earth. The cost of these supplies has increased each month due to inflation and rising transportation expenses. The price of oxygen cylinders has increased by p% each month based on the previous month's price. The price of water reserves has increased by q% each month based on the previous month's price. Write a program that takes as input the price of an oxygen cylinder and a unit of water reserve in 1999, along with the values of p and q. The program should then calculate and print the current price of both items and determine which item's price has increased more in total over these years.

Sample Input	Sample Output
Enter the initial price of an oxygen cylinder in 1999: 120 Enter the initial price of a unit of water reserve in 1999: 20 Enter the monthly inflation rate (percentage) for oxygen cylinders: 1 Enter the monthly inflation rate (percentage) for water reserves: 1.5	Current price of an oxygen cylinder: 2675.78 Taka Current price of a unit of water reserve: 2081.78 Taka The price of oxygen cylinders has increased more.
Enter the initial price of an oxygen cylinder in 1999: 45 Enter the initial price of a unit of water reserve in 1999: 15 Enter the monthly inflation rate (percentage) for oxygen cylinders: 1 Enter the monthly inflation rate (percentage) for water reserves: 2.3	Current price of an oxygen cylinder: 1003.42 Taka Current price of a unit of water reserve: 18083.77 Taka The price of water reserves has increased more.

Nested Loop

Ronaldo was getting bored at university so he decided to create and play a game. The game is going to take a 4 digit number as input and he will have 3 lives to win this game. First, he is going to get the last digit and check if the number is prime or not. If he finds 3 prime numbers before he loses his 3 lives then he wins. Otherwise, game over and he lost.

Sample Input	Sample Output
Enter a 4-digit number: 2486	6 is not prime. Lives left: 2 8 is not prime. Lives left: 1 4 is not prime. Lives left: 0 Game over! You lost.
Enter a 4-digit number: 3475	5 is prime! (1/3 found) 7 is prime! (2/3 found) 4 is not prime. Lives left: 2 3 is prime! (3/3 found) Congratulations! You won the game!

Problem 1:
Suppose, you are given the following equation:

$$sum \ = \ 7 \ + \ 7 \ - \ 7 \ + \ 7 \ + \ 7 \ - \ 7 \ + \ 7 \ + \ 7 \ - \ 7 .... upto \ n^{th} \ term$$

Write a Java program that takes the value of the total number of terms (value of n in the equation above where n > 0) and calculates & displays the value of sum. **Note that, you are NOT allowed to use any loops to solve the problem.**

Sample Input	Sample Output
3	7
7	21
11	35

Problem 2:
Write a Java program that takes two integer numbers: a & b as user inputs and calculates a%b and a/b without using the modulus (%) and division(/) operators respectively.

Problem 3:
Write a Java program that takes two integers: U and L. Here, U & L denotes the upper & lower limit respectively. Your task is to find all the numbers from U to L (both inclusive) where in each number sum of all the digits is prime.

Sample Input	Sample Output
10	11, 12, 14, 16, 20.

20	
36 48	38, 41, 43, 47.

Problem 1:

You have bought a new circle shaped analog clock for your drawing room. One relative of yours who is also a top ranked CS Professor in a prestigious University of Bangladesh came to visit you. By looking at your clock, he first asked you to tell the correct time (in hours and minutes). You as a smart student quickly answered it. But in the next question, he asked you to find the smaller and larger angle of the hour & minute hands of that analog clock. He also gave you two hints to solve this:

- The hour hand moves 360 degrees in 12 hours
- The minute hand moves 360 degrees in 60 minutes

Your task is to write a Java program that takes two integer inputs: current time in hours (12H format) and minutes. Using the above hints you need to calculate the smaller and larger angle of the hour & minute hands of your clock.

Sample Input	Sample Output	Explanation
Hours: 4 Minutes: 35	Smaller angle: 72.5 degrees Larger angle: 287.5 degrees	The hour hand moves (360/12) = 30 degrees per hour and it also slightly moves (30/60) = 0.5 degree per minute. So, if we calculate the angle between the hour hand from 12:00, then we get: (4 x 30) + (35 x 0.5) = 137.5 degrees Also, the minute hand moves (360/60) = 6 degrees per minute. So, if we calculate the angle between the minute hand from 12:00, then we get: (35 x 6) = 210 degrees Taking the absolute difference between the two angles, we get the smaller angle between hour and minute hands: |137.5 - 210| = 72.5 degrees And the larger angle is simply 360 - 72.5 = 287.5 degrees.
Hours: 8 Minutes: 2	Smaller angle: 131.0 degrees Larger angle: 229.0 degrees	Angle between the hour hand from 12:00 = (8 x 30) + (2 x 0.5) = 241 degrees Angle between the minute hand from 12:00 = (2 x 6) = 12 degrees Angle between hour and minute hands = | 241 - 12 | = 229 degrees; as this is more than 180 degrees this is actually the larger angle. Smaller angle = 360 - 229 = 131 degrees.

Problem 2:

Write a Java program that takes two integer numbers as user inputs and finds the Greatest Common Divisor of the numbers using Euclidean approach. The approach is as follows:

1. Store the large number in a and small number in b.
2. Calculate a%b . Replace a with b and b with the remainder you have found.
3. Repeat step 2 until the remainder is 0. The last non-zero remainder (currently in a) is the GCD of original a & b.

Sample Input	Sample Output	Explanation
12 18	6	a = 18 (largest), b = 12 (smallest) Now, gcd(18, 12) = gcd(12, 18%12) = gcd(12, 6) = gcd(6, 12%6) = gcd(6,0) = 6
9 7	1	gcd(9, 7) = gcd(7, 9%7) = gcd(7, 2) = gcd(2, 7%2) = gcd(2, 1) = gcd(1, 2%1) = gcd(1, 0) = 1

Problem 3:

Write a Java program that takes three integers: U, L and D. Here, U & L denotes the upper & lower limit respectively and D is a specific digit from 0 to 9. Your task is to find out which numbers from U to L (both inclusive) contain the digit D.

Sample Input	Sample Output
20 50 3	23, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 43.
15 29 7	17, 27.

Branching:

Cody wants to get fit and is considering two gym memberships options.
Basic Plan - Costs x taka per month and includes gym access and group workouts.
Premium Plan - Costs 30% more than the Basic Plan but includes personal training sessions.
Cody has a total monthly fitness budget of x taka. If he chooses the Basic Plan, he also wants to join a swimming club that costs 15% of the Basic Plan's price.
Write a program that takes x (Basic Plan cost) and y (Cody's budget) as input and determines if he can afford the Premium Plan or if he can afford the Basic Plan with the swimming club. If neither is possible, print how much more money he needs.

Linear Loop:

A librarian is tasked with organizing a large collection of books on shelves. Each shelf (or row) can hold up to k books. The books are added one by one until a shelf is completely filled; then, the librarian starts filling a new shelf. However, to allow for future expansion, every 3rd shelf is intentionally left empty. Write a program that takes two inputs: the total number of books n and the row capacity k. Your program should simulate the placement of books using a linear loop and then print the total number of shelves used, the number of books on the last non-empty shelf.

Nested Loop:

A rectangular garden has R rows and C columns. Each plot requires water equal to the product of its row and column numbers (water needed = row × column). The gardener starts with a watering can that holds a fixed capacity. He waters each plot in sequence; if the water required exceeds what remains in the can, he refills it (resetting it to full capacity) before watering that plot. [Here the row and column starts with 1]
Write a Java program that takes the number of rows R, the number of columns C and the fixed capacity of the watering can in Litres as input and prints the total water used and the number of refills performed.

Branching:

Jane lives in a city where electricity costs are calculated based on usage and choices. She currently spends x units of electricity per month. Per unit costs 9 taka. If her usage exceeds 500 units, she is charged a 20% additional surcharge on her total bill. However, if she uses an energy-saving plan, she can reduce her consumption by 30%. Additionally, she is considering installing solar panels, which will cover 40% of her electricity needs, reducing her bill accordingly.

Write a program that takes as inputs the units of electricity x and two Boolean values specifying if Jane opts for the energy-saving plan and if she installs solar panels. The program should calculate and print her final electricity bill after considering these choices.

Linear Loop:

A city's transportation department wants to simulate the growth of vehicles on the road over a period of months. Initially, there are *V* vehicles. Each month, the number of vehicles increases by a fixed percentage, x%, due to urban expansion. However, every 6 months, the city enacts a temporary traffic regulation to ease congestion that removes y vehicles from the total. Write a program that takes as input the initial number of vehicles *V*, the monthly growth percentage *x*, the reduction amount *y*, and the total number of months to simulate and print the final number of vehicles after the simulation period.

Nested Loop:

A popular theater is preparing for a sold-out performance. The theater has several rows, and each row contains a fixed number of seats. However, to accommodate VIP guests, any seat whose number is a multiple of 5 is reserved and not available for general booking. Write a program that takes as input the total number of rows and the number of seats per row, then uses nested loops to determine and print the total number of seats available for the general public and the total number of VIP reserved seats.

Branching:

A car rental company offers three types of cars: economy, standard, and luxury, each with different daily rental rates. The company provides discounts based on rental duration and weekends. If the rental period exceeds 7 days, a 10% discount is applied to the total cost. Additionally, if the rental is during the weekend, a 5% discount is given. Your task is to write a Java program that calculates the total rental cost after applying the best possible discount, based on the car type selected, the rental duration, and whether the rental falls on a weekend. The program should ask the user for the car type, rental duration in days, and whether it's a weekend rental, and then output the total cost after applying the best discount.

Linear Loop:

A customer is shopping online and adding multiple items to their shopping cart. Each item has a name, price, quantity, and category (electronics, clothing, or others). The system needs to calculate the total price of the cart, applying discounts for bulk purchases (10% discount if the quantity exceeds 10) and category-specific discounts (5% for electronics and 15% for clothing). After calculating the discounts for each item, the system should display the original cost, the discount applied, and the final price after the discount for each item. At the end, it should show the total cost of the cart before discounts, the total amount of discounts applied, and the final

total cost after applying all discounts. Write a Java program that performs these calculations and displays the results to the user.

Nested Loop:

In an airport, there are multiple gates, and each gate handles multiple flights. Each flight has a different number of passengers, and the boarding time per passenger varies for each flight. Your task is to write a program that simulates the boarding process for all flights at the airport. The program should first ask the user for the number of gates at the airport. For each gate, the user should input the number of flights, followed by details for each flight, including the number of passengers and the average boarding time per passenger. The program should then calculate the total boarding time for each gate, the average boarding time per passenger for each gate, and the total and average boarding times for all passengers across all gates. After processing all gates, the program should output the results, including the total boarding time per gate and the average boarding time per passenger across the entire airport.

Branching Question:

You are developing a movie streaming app that offers users the ability to subscribe to either a Standard Plan of 3 months or a Premium Plan of 12 months. The Standard plan consists of facilities like watching 1 screen at a time, 3 device logins and 300 tk monthly payment. A Premium subscription consists of facilities like watching 3 screens at a time, 5 device logins, and a 15% lower monthly payment amount then the Standard subscription. Take the subscription choice (Integer data type: 1 for Standard, 2 for Premium) as user input and calculate the fee required to pay according to the subscription choice of the user and also show the facilities they will get.

Linear loop Question:

A company, operating across multiple regions, is trying to manage their revenue versus expenditure ratio, while keeping the profit margin as at least 25% of the costing (Ex: 100 tk costing & 125 tk revenue = 25 tk profit, which is 25% of the costing). The regions deemed as profitable will continue operations, while the regions incurring loss will stop operations. Take the number and name of regions, the expenditure and revenue in each region as user inputs and find out which regions are producing profit for the company and which are not. The company also aims to find the region with the highest and lowest profit as well as the average profit amount from the profitable regions. If there were no profitable regions, these information should not be shown.

Nested Loop Question:

Multiplication made fun!

A mathematics teacher is trying to think of creative ways to invent a fun multiplication problem of 2 integers. He finally gets an idea! Each digit of an integer will be multiplied with all of the digits of another integer in forward iteration, however there is a twist. Each odd digit of the first integer will only be multiplied by the odd digits of the second integer and similarly, each even digit of the first integer will only be multiplied by the even digits of the second integer. Ex: Integer1: 123456, Integer2: 234567. The first digit of Integer1 is 1. It should be multiplied with all the odd digits of Integer2, i.e. 1*3*5*7 = 105. Then the next digit of Integer1 is 2. It should be multiplied with all the even digits of Integer2, i.e.2*2*4*6 = 96 and so on. Multiplication results will yield 105, 96, 315, 192, 525, 288. Your code should work for integers of any length.

1.

You are working on a digital calendar system that needs to identify **leap years** for accurate date tracking. A year is a leap year if:

It is divisible by **400**, OR
It is divisible by **4** but **not by 100**

Your task is to write a program that determines whether a given year is a leap year or not.

2.

A financial analyst wants to **predict investment growth** using the Fibonacci sequence. The first two months of growth are 0% and 1%, and after that, each month's growth is the sum of the previous two months.

Write a program to print the **first N terms** of the Fibonacci sequence.

3.

You are developing a **banking app** that allows users to input their PIN code for authentication. To improve security, the app records the frequency of each digit used in the user's PIN code over time. The app will then display the frequency of each digit (0-9) in the PIN code entered by the user.

Input	Output
Enter your PIN code: 445899	Frequency of digits: 0: 0 1: 0 2: 0 3: 0 4: 2 5: 0 6: 0 7: 0 8: 1

1.

Write a program that calculates the electricity bill based on the following rates:

● First 100 units: 5 BDT per unit

● Next 200 units: 8 BDT per unit

● Above 300 units: 10 BDT per unit

 Take the number of units consumed as input and print the total bill.

2.

A telecom company wants to analyze mobile numbers to find the largest and smallest digit in a given number. This will help in identifying patterns for better number distribution.

Write a program that extracts the smallest and largest digit in a given number.

3.

A football coach wants to pair players in such a way that their jersey numbers Adds to a prime number.

Write a program that finds all unique pairs of digits (d1, d2) in N where: d1+d2 is a prime number

1.

Take two inputs representing choices of two players: Rock, Paper, or Scissors.

● Rock beats Scissors

● Scissors beats Paper

● Paper beats Roc

Print the winner, or declare a tie if both choose the same option.

N.B: You can represent 1, 2, 3 as Rock, Paper, and Scissor consecutively.

2.

You are developing a lottery system where the sum of even and odd digits in a ticket number determines the prize category.

● The sum of even digits represents the bonus prize amount.

● The sum of odd digits determines the main prize category.

Write a program to calculate and print both sums separately.

3.

A bank assigns unique account numbers to new customers. Each account number consists of a branch code (first N digits) followed by a customer ID (Last M digits). Generate all possible account numbers for that bank

Input	Output
2 3	01001
	01002
	01003
	..
	..
	45001
	45002
	..
	99999

Branching Question: A restaurant offers different discounts based on the time of the day. The restaurant opens at 8 AM and closes at 10 PM. If a customer visits the restaurant during happy hour (10 AM - 12 PM), they get a discount on the total price. The discount amount depends on the time of the happy hour. If the customer visits at time x, they will receive an x% discount, where x is the hour during happy hour. Write a Java program that takes the time of the visit in hours (integer) and the total price as inputs from the user. If the customer visits during happy hour, the program should calculate and print the discount percentage and total price after the discount. The program should handle all invalid inputs, such as entering a time outside the restaurant's operating hours.

Linear Loop Question: You and one of your friends will participate in the upcoming 5 km marathon, which will be held in Hatirjheel. Both of you have been practicing for the last few days. Let x be the total number of days you have been practicing, and y be the total number of days your friend has been practicing. Write a Java program that takes your daily running speeds (in km/h) over the past x days and your friend's daily running speeds (in km/h) over the past y days as inputs from the user. The program should calculate the average running speed for both

of you. The program should also calculate the approximate time (in hours) you and your friend will take to cover the distance of the upcoming marathon based on the average speed.

Nested Loop Question: At XYZ Company, most of the employees tend to arrive late to the office. To encourage better punctuality, the HR department implements a rule that if an employee comes late to the office, then they need to pay $0.5 for each late coming day. At the end of the month, the total amount collected from these late fees will be used to organize a party. Write a Java program that first takes the total number of employees as input from the user. Then, for each employee, the program should prompt the user to enter a boolean value for each of the 30 days of the month, indicating whether the employee was late on that day. The program then calculates and prints the total late fee collected from all employees over the month.

Branching Question: Yakutsk experiences some of the coldest temperatures in the world, with winter temperatures often dropping below -40°C. People living in Yakutsk must balance their monthly budget based on the temperature. The monthly budget of a family that lives in Yakutsk is x taka. The family needs to spend the money for energy consumption based on the temperature of the city. If the temperature is between -10°C and -20°C, they use 15% of their monthly budget. If the temperature is between -20°C and -30°C, they use 30% of their monthly budget.If the temperature is below -30°C, they use 50% of their monthly budget due to the extreme cold. Write a Java program that takes the monthly budget x and the temperature as inputs from the user. Based on the temperature, the program should calculate and print the total money spent on energy consumption for the month.

Linear Loop Question: To live a healthy life and maintain a regular exercise routine, you want to develop a fitness tracker app. The app will store the number of steps (x) the user takes, as well as the total number of minutes (y) the user walks each day. The user's daily goals are 10,000 steps and 100 minutes of walking. If the user meets the goal for the day, there will be no additional adjustments to their progress. However, if they fall short, the app will show the percentage of the goal they achieved. Write a Java program that takes x and y as inputs from the user to calculate the total number of steps taken & total minutes walked during the week. It will check and print if the user met the daily goals of 10,000 steps and 100 minutes on average. The program will also print the percentage of the goal the user achieved for each day they fall short of their goal.

Nested Loop Question: You are working as a project manager in a construction company. The company has a team of five workers, and each worker is assigned to complete a specific number of tasks each week. The company has set a target of completing at least 50 tasks (per worker) and 250 tasks in total by all of the workers by the end of the week. The company wants to evaluate the weekly performance of each worker and track the total number of tasks completed during the week. Write a Java program that takes the number of tasks completed by each worker for each of the 7 days of the week as input. For each of the workers, the program will calculate the total number of tasks completed during the week and print if the worker met his weekly goal of 50 tasks. Also, it will print if the company has achieved the weekly goal in completing at least 250 tasks.

Question 1(Branching)

A **washing machine** operates based on a **Fuzzy System**, where the weight of clothes placed inside is uncertain. However, using sensors, it determines the **required washing time and water level**, which can be adjusted through menu options on the control panel.

Apart from the weight of clothes, the machine also checks for the **detergent level**. The detergent must be **at least 20 ml** for the washing process to begin; otherwise, the machine will display **"INSUFFICIENT DETERGENT"**.

The estimated time for different water levels is determined based on the weight of clothes:

● **Low-level water** → Time Estimated: **25 minutes** for weight between **1 and 2000 grams**.

● **Medium-level water** → Time Estimated: **35 minutes** for weight between **2001 and 4000 grams**.

● **High-level water** → Time Estimated: **45 minutes** for weight between **4001 and 7000 grams**.

● **Zero weight (0 grams)** → Time Estimated: **0 minutes**.

● **Weight exceeding 7000 grams** → Output: **"OVERLOADED"**.

● **Detergent level less than 20 ml** → Output: **"INSUFFICIENT DETERGENT"**.

● **Any invalid input (negative values or non-numeric inputs for weight or detergent)** → Output: **"INVALID INPUT"**.

Write a Java program that takes two integer inputs (weight of clothes and detergent level) and prints the estimated washing time.

Sample Input	Sample Output
2000 50	Time Estimated: 25 minutes
7500 30	OVERLOADED

3000 10	INSUFFICIENT DETERGENT
-5 25	INVALID INPUT

Question 2(Loops)

The Ancient Mystery Series!

Deep in the archives of mathematical history, an **enigmatic sequence** has puzzled scholars for centuries. This series, composed of two intertwined sub-series, has a **hidden structure** that can only be uncovered by carefully following a **step-by-step process**.

The Mysterious Series : 0, 0, 2, 1, 4, 2, 6, 3, 8, 4, 10, 5, 12, 6, 14, 7, 16, 8, …

- All odd-positioned terms (1st, 3rd, 5th, ...) form an increasing sequence of even numbers starting from 0.

- All even-positioned terms (2nd, 4th, 6th, ...) are derived by dividing the previous term by 2.

Your task is to **write a Java program** that takes an integer **n** (where 1 ≤ n ≤ 20,000) as input and finds the **nth term** in the series.

IMPORTANT CONSTRAINT:

- The solution MUST use a loop (i.e., you are NOT allowed to use direct formulas).
- You must start from the beginning of the series and calculate each term iteratively.

Sample Input	Sample Output	Explanation

10	4	9th term = 8 (follows the even-number sequence). 10th term = 8 / 2 = 4.
15	14	The 15th term follows the even-number sequence at odd position 14.

Question 3(Nested Loops)

Long ago, in the mystical world of Java Land, a sacred Butterfly symbol was crafted by the greatest coders of all time. This symbol is known for its perfect balance, symmetry, and elegance. It is said that only the most skilled programmers can unlock its true form by using nested loops and if-else statements. Your task is to take a single integer n (3 ≤ n ≤ 1000) as input and print a butterfly-shaped symbol, using only *, -, /, \, and @.

The **Butterfly** follows strict **mathematical principles**:

1. The total **width** is **2 * n - 1** characters.
2. The total **height** is **2 * (n - 2) + 1** rows.
3. The **left and right wings** are **n - 1** characters wide each.
4. At the **center of the emblem**, there is a **sacred @** symbol, representing balance.
5. The **wings are made with alternating patterns**:

   - If **n > 3**, odd rows use *****\ /*****, even rows use **---\ /---**.

   - If **n = 3**, the first row uses ***\ /*** to maintain symmetry.

   - The bottom half mirrors the top, but **slashes (/ and \) are swapped**.

Sample Input	Sample Output

| 3 | ```
*\ /*
   @
*/ \*
``` |
|---|---|
| 5 | ```
***\ /***
---\ /---
***\ /***
      @
***/ \***
---/ \---
***/ \***
``` |
| 7 | ```
*****\ /*****
-----\ /-----
*****\ /*****
-----\ /-----
*****\ /*****
        @
*****/ \*****
-----/ \-----
*****/ \*****
-----/ \-----
*****/ \*****
``` |

1.

A family spends x taka each month on groceries, with their budget divided into four categories: 40% for vegetables and fruits, 30% for meat and fish, 20% for dairy products, and 10% for snacks and beverages. To save money, they are considering two changes—growing their own vegetables, which would cut vegetable and fruit costs in half, and buying meat and fish wholesale, which would reduce those costs by 25%. Write a Java program that takes x as input, along with two Boolean values (growVegetables & wholesaleMeatFish) indicating whether they have implemented these changes, and calculates the new total grocery cost after applying the discounts.

2.

You are tasked with designing a smart elevator system for a building with 10 floors. The elevator starts at the ground floor (Level 1) and responds to user input, where "U" indicates moving up and "D" indicates moving down. If an invalid request is made, such as attempting to move up from the 10th floor or down from the 1st floor, the system should display an error. The program should allow users to enter multiple requests, like "UUDDUDU", and the elevator must process these requests in order, efficiently optimizing its movements. The system should use loops to handle the requests and ensure that the elevator follows the correct sequence of movements without exceeding the floor limits.

3.

In a school, there are multiple classrooms, each with different subjects taught by various teachers throughout the day. Your task is to write a program that simulates the school schedule for all classrooms. The program should first prompt the user to input the number of classrooms. For each classroom, the user should input the number of subjects scheduled, followed by details for each subject, including the subject name, teacher's name, and the duration of the subject in hours. The program should then calculate and output the total number of hours each classroom is used during the day.

1.

Design a Java program for a multi-stage loan approval system that processes a loan application based on the applicant's credit score, monthly income, existing debt, and loan amount. The program should follow these rules: If the credit score is below 600, the loan is **rejected** immediately. If it falls between 600 and 700 and the applicant's existing debt exceeds 40% of their monthly income, the loan is **rejected**; otherwise, it moves to the next stage. If the credit score is above 700 and the loan amount exceeds five times the monthly income, the loan is **conditionally approved**, requiring additional documentation. If the loan amount is within five times the monthly income, the loan is **approved**. However, if a conditionally approved applicant has existing debt exceeding 50% of their income, the loan is **rejected**. Implement this decision-making process in Java using only if-else statements.

2.

Develop a Java program that calculates the amount owed on a credit card after 6 months, based on an initial balance (entered by the user), an annual interest rate (APR) of 18%, and monthly payments (entered by the user for each month). The program should compute the balance each month, showing the starting balance, interest applied, payment made, and remaining balance. The monthly interest rate is derived from dividing the APR by 12 and converting it to a percentage. If the payment is less than the interest accrued, the balance will increase. The program should display a month-by-month breakdown, and stop early if the balance is paid off before the 6 months.

3.

In a busy train station, multiple platforms are handling various trains throughout the day. Each platform has different types of trains arriving, with varying numbers of passengers waiting for each train. The average wait time for passengers differs between trains, and the goal is to calculate the total wait time for all passengers at each platform and across all platforms. Based on this scenario, write a program that allows the user to input the number of platforms, the number of trains at each platform, the number of passengers for each train, and the average wait time per passenger. The program should calculate the total wait time for all passengers at each platform and compute the average wait time per passenger across all platforms.

1.

A smart traffic system calculates fines based on the speed of a vehicle relative to the speed limit. If the speed is within the limit, there is no fine. If the speed exceeds the limit by 1-10 km/h, the fine is $50, if it exceeds by 11-30 km/h, the fine is $150 and if it exceeds by more than 30 km/h, the fine is $500 along with a license suspension. Additionally, if the driver has had 2 or more past violations, the fine is doubled. Write a Java program that takes the vehicle's speed, speed limit, and past violations count as input and determines the appropriate fine.

2.

An investor places an amount A in a bank account that offers an annual interest rate of r%, compounded yearly. However, there is a 5% tax deduction on the earned interest each year unless the investor has a tax return, in which case the tax is waived. Write a Java program that calculates the value of the investment after nnn years using a loop, applying the annual interest rate and deducting the tax unless the investor is tax-exempt. The program should also compute the total increase in the investment, which is the difference between the final value and the initial amount A. The program should take user input for A, r, n, and whether the investor has a tax return (as a boolean value). It should then output the value of the investment after n years and the total increase in the investment. For example, if the initial investment is $1000, the interest rate is 5%, the investment period is 10 years, and the investor does not have a tax return, the program should output the final investment value as $1538.63 and the total increase as $538.63. If the investor has a tax return, the values should be $1628.89 and $628.89, respectively. Implement the program using a class `Investment`, utilizing a loop to calculate the

investment value each year, applying tax deductions conditionally, and using appropriate data types like `double` for monetary values.

3.

In a smart city traffic management system, an intersection is monitored with a traffic light that alternates between red and green every minute, with the red light lasting 60 seconds and the green light lasting 30 seconds. During rush hour (8 hours), the number of vehicles at the intersection is provided by the user, and the simulation calculates the total wait time for all vehicles, assuming they wait for the entire red light duration. The program then computes the average wait time per vehicle by dividing the total wait time by the number of vehicles. Given this scenario, write a Java program to simulate this traffic management system, taking user input for the number of vehicles, calculating the total wait time, and printing the average wait time for each vehicle based on the provided traffic light durations.

Branching Question:

A shipping company has to deliver products to a location. The truck that will carry the products has a capacity of X ton. Product weight exceeding this capacity cannot be shipped. If weight is within capacity, the shipping charge should now be calculated. The shipping charge is a product of the weight and distance. For 20 KG or less, weight charge will be 5 tk per kg. 3 tk will be added per every additional kilogram of weight. This charge should be multiplied with the distance covered to find the total shipping charge. Take the truck capacity, X (in kg), product weight (in km) and distance (in km) to be covered as double type inputs to solve this problem.

Linear loop Question:

The employees of Z company get their Eid bonuses each year on the basis of their annual salary earned that year. The salary was X tk during their joining year which increments by a fixed amount each year. For the next 3 years after the joining year, their salary incremented by 10% and by 15% for the years after that. Their Eid bonuses are 25% of their salary every year. Show the bonuses earned by the employees starting from their joining year to the current year by taking their initial annual salary (double data), start and current year as user inputs.

Nested Loop Question:

For N number of students and M number of subjects, the CSE department wants to calculate the students' performance across all subjects by calculating the mean and standard deviation of their subject scores. For that, you need to take N, M as user input and also the scores of each student as user input.

Formula of standard deviation:

$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{N}}$$

σ = population standard deviation

N = the size of the population

x_i = each value from the population

μ = the population mean

Branching Question: A company offers health insurance to its employees based on their age, health condition, and the number of years they have worked at the company. Employees between 20-40 years of age are eligible for health insurance only if they do not have any pre-existing health conditions. Employees between 40-60 years of age are eligible only if they have worked for the company for more than 5 years. Employees over 60 years of age are eligible only if they have been employed for more than 10 years. Write a Java program that takes the employee's age, health condition (true/false), and the number of years they have worked at the company, and determines if they are eligible for health insurance based on these criteria.

Linear Loop Question: A city in Dhaka monitors the water level in its main reservoir over a period of days, with daily rainfall increasing the water level in the morning by a fixed percentage and city-wide consumption reducing it each evening. Write a Java program that takes four inputs: the initial reservoir level (liters), the percentage increase due to rainfall, the number of days, and the percentage of water consumed each evening. For each day, the program should display the starting water level, the amount added by rainfall, the amount consumed, and the final water level. At the end of the simulation, the program should report the total water gained, total water used, and a comparison of the final water level with a scenario where no water was consumed.

Nested Loop Question: Write a Java program for a company with N employees. In that company each employee tracks three key metrics (tasks completed, hours worked, and breaks taken) for seven consecutive days (Monday through Sunday). The program should read the number of employees and then use nested loops to collect data for each employee for each day. After collecting the data, calculate and display the average number of tasks completed, average

hours worked, and average breaks taken for each day. Additionally, track the employee who completed the most tasks on each day. Finally, determine and display the employee who worked the most total hours across all seven days. Avoid using arrays and instead use variables to accumulate totals and perform calculations on the fly.

Branching Question: Sumaiya runs a clothing outlet during Ramadan. She is focusing on three main product lines, which are Men's Panjabi (50%), Women's Salwar Kameez (30%), and Children's Eid Sets (20%). She invests a total amount in these products, and a tiered discount scheme applies based on her total expenses: a 10% discount for up to 300,000 Taka, a 20% discount for 300,000 to 700,000 Taka, and a 30% discount for anything beyond 700,000 Taka. After this tiered discount, Sumaiya qualifies for a combined promotional reduction of 25% (from store loyalty programs, online coupons, and a Ramadan special sale). If the remaining expense after all these discounts is less than 200,000 Taka, she gets an additional 10% off the remainder to reach her budget goals. Write a Java program that takes the total expense amount as input and outputs the final expense after applying all the discounts.

Linear Loop Question: Consider a grocery store in Dhaka that tracks its rice stock over several consecutive days. Each morning, the store increases its inventory by a certain percentage of the previous day's remaining stock, and every evening a spoilage rate reduces the stock. Write a Java program that takes four inputs—(1) initial stock (int), (2) daily restock percentage (%), (3) number of days (int), and (4) daily spoilage rate (%)—and, for each day, prints the starting stock, the restocked amount, the spoilage deduction, and the final adjusted stock. At the end, the program should report the total rice added over the entire period, total spoilage, and compare the final stock with what it would have been had there been no spoilage at all.

Nested Loop Question: In Dhaka, there are N restaurants running a dinner buffet for seven consecutive days (Monday through Sunday). Each restaurant tracks three integers for each day: the number of visitors, the number of leftover dishes, and the number of fresh orders placed for the following day. Write a Java program that first reads N (int), then uses nested loops to collect data for each of the seven days (outer loop) and each of the N restaurants (inner loop), calculating the average visitors, leftovers, and new orders per day across all restaurants. After collecting and averaging the data for a day, print the results immediately (or store them in variables for that day, then print), then proceed to the next day.

Branching Question: Rizwan, a keen adventurer, manages his living from three sources such as 50% from trekking excursions, 30% from his travel blog, and 20% from selling presents. His overall income determines his travel expenses in a tiered manner: 30% for revenue under 500,000 taka, 40% for income between 500,000 and 1,200,000 taka, and 50% for income over 1,200,000 taka. After the expense calculation, Rizwan can save a total of 23% (10% from working with a travel agency, 8% from sponsorship, and 5% from booking early). If his remaining balance after expenses and discounts is less than 200,000 taka, he can save an additional 12%

on the remaining expenses (after all discounts) to fulfill his savings goal. Now, write a Java program with the given scenario and where income will be taken from the user and prints total expenses and the final savings..

Linear Loop Question: Write a Java program that tracks an employee's salary growth over time, taking into account annual percentage raises, a performance-based bonus applied every other year (beginning in the second year), and the effect of inflation on real income growth. The program's inputs should be the initial salary, annual percentage increase, number of years, performance bonus %, and inflation rate. It should print the wage progression for each year, including the adjusted real salary after accounting for inflation. To give a complete financial analysis, the program should also calculate the total income earned, the total salary increase, and compare salary growth with and without inflation.

Nested Loop Question: Write a Java program that calculates the average temperature, humidity, and precipitation for every day of the week across multiple locations in a city. There are 7 days in a week, from Monday to Sunday. The city has N locations, and N is user input. Each location has daily records for temperature (in °C), humidity (in %), and precipitation (in mm). The program should take input for N (number of locations). Take input for each day's data (temperature, humidity, and precipitation) for all locations. Use nested loops to calculate the average temperature, humidity, and precipitation per day across all locations. Print the results for each day in a formatted manner.

Branching Question: Mr. X earns 60% of his income from principal employment, 25% from freelancing, and 15% from rental income, with a total annual income T given as input. His income is taxed gradually: the first 300,000 Taka is tax-free, the next 500,000 Taka is taxed at 10%, and any income beyond 800,000 Taka is taxed at 20%. If he pays for health insurance, his taxable income is reduced by 10% before tax calculation. If he donates at least 10% of his income to charity, he gets a 5% reduction on the final tax, and if he invests in government bonds, he gets an 8% reduction on the final tax. If both charity and bond reductions apply, the charity reduction is applied first, followed by the bond reduction. Write a Java program that takes T, along with three boolean inputs (for health insurance, charity donation, and bond investment), and calculates the final tax amount.

Linear Loop Question: Five years ago, the populations of Town X and Town Y were recorded as X0 and Y0 , respectively. Over the past five years, both towns have experienced consistent compounded monthly growth at rates of p% and q%, respectively. Write a Java program that calculates the population of both towns after 3 years (36 months), tracking the monthly population increase for each town, the population difference between them every month, and the total population increase for both towns. The program should take four space-separated inputs: X_0 Y_0 p q, where X0 and Y0 are the initial populations, and p and q are the monthly growth rates (in percentage, compounded monthly). The output should display the monthly population for both towns, the increase in population each month, the final populations after 36

months, the total population increase, and a message indicating which town had the greater absolute population increase over the 3-year period.

Nested Loop Question: Imagine you're playing a treasure hunt game. In that game, the player walks through chambers full of traps, each of which diminishes their health by a set percentage. With each successive room, the quantity of traps grows by a certain proportion. After completing each room, the player regains some health. However, the recuperation process slows as they progress. The game finishes when the player's health reaches zero or falls below a predetermined threshold. Write a Java program to calculate how many rooms the player can survive based on the player's beginning health and the given conditions. (One loop to iterate through rooms, and a nested loop to calculate the health decrease for each trap in the room.)

Q1. A **gaming tournament** is held where players compete in multiple rounds to earn points and win prizes. Each player starts with **s points** and competes in three types of challenges:

1. **Speed Round** – If a player wins this round, their total points increase by **40%**. If they lose, their points decrease by **20%**.
2. **Strategy Round** – If a player wins this round, their total points increase by **25%**. If they lose, their points decrease by **10%**.
3. **Endurance Round** – This round is unique:
   - If the player's total points (after the first two rounds) are **greater than or equal to 200,** they automatically pass and gain an additional **15% bonus points**.
   - Otherwise, they must compete. If they win, they gain **30% points**; if they lose, they lose **25% points**.
4. If the player's final score exceeds **400 points**, they earn a **"Grand Master Bonus"**, which adds an extra **50 points.**

Now, write a Java program that takes as input:

- The initial number of points **s**
- Two Boolean values indicating whether the player won the **Speed Round** and the **Strategy Round**
- A third Boolean value indicating whether the player won the **Endurance Round** (if required)

After all rounds, the program should compute and print the **player's final score.**

Input	Output	explanation
100 true true	Final Score: 227.5	Initial score, s =100 Speedy Round: True So, S = 100 * 1.4=140

true		Strategy Round = True So, S = 140 * 1.25 = 175 S < = 200 , So, the Endurance round will be played. Endurance round = True S = 175 * 1.30 = 227.5 Since S <= 400, the Grand Master Bonus does not apply.
75 false true false	Final Score: 56.25	Initial score, s =75 Speedy Round: false So, S = 75 * 0.80 = 60 Strategy Round = True So, S = 60 * 1.25 = 75 S < = 200 , So, the Endurance round will be played. Endurance round = false S = 75 * 0.75 = 56.25 Since S <= 400, the Grand Master Bonus does not apply.
300 true false	Final Score: 484.7	Initial score, s =300 Speedy Round: True So, S = 300 * 1.4 = 420 Strategy Round = false So, S = 420 * 0.90 = 378 S > = 200 , So, the Endurance round will not be played. S = 378 * 1.15 = 434.7 Since S >= 400, the Grand Master Bonus will be applied. S = 434.7 + 50 = 484.7

Q2. Sheldon, Amy, Penny, and Leonard are playing a game where each of them must name a country. The game follows these rules:

1. Each player earns **points equal to the length of the country's name** they say.
2. If a player **fails to recall a country's name**, they say **"Giveup"**, and they are **eliminated** from the game.
3. The game **continues with the remaining players** until only **one player is left**.
4. The last remaining player is declared the **winner**.

Now, write a **Java program** that takes four strings as input, each representing the name of a country given by Sheldon, Amy, Penny, and Leonard. Eliminates players who say **"Giveup"** and continues until one player remains. And lastly, prints the **winner's name** and the **final scores** of all players.

You may use str.length() and str.equals() methods.

Input	Output	Explanation
Sheldon: Australia **Amy:** Canada **Penny:** Brazil **Leonard:** Giveup **Sheldon:** Argentina **Amy:** Giveup **Penny:** India **Sheldon:** Pakistan **Penny:** Netherlands **Sheldon:** Mexico **Penny:** Giveup	Leonard is eliminated! Amy is eliminated! Penny is Eliminated. Winner: Sheldon Final Scores: Sheldon: 32 Amy: 6 Penny: 22 Leonard: 0	Sheldon: "Australia" → 9 points Amy: "Canada" → 6 points Penny: "Brazil" → 6 points Leonard: "Giveup" → Eliminated **The remaining players:** Sheldon, Amy, and Penny Sheldon: "Argentina" → 9 points (Total: 9 + 9 = 18) Amy: "Giveup" → Eliminated Penny: "India" → 5 points (Total: 6 + 5 = 11) **The remaining players:** Sheldon, Penny Sheldon: "Pakistan" → 8 points (Total: 18 + 8 = 26) Penny: "Netherlands" → 11 points (Total: 11 + 11 = 22) Sheldon: "Mexico" → 6 points (Total: 26 + 6 = 32) Penny: "Giveup" → Eliminated **Sheldon remains, hence, he is the winner.**

Q3. Three brave friends—**Jessy, Sarah, and Oliver**—are on a mission to defeat a **Devil**. The Devil's **life force** is hidden in a **5-digit number**. The friends must guess the digits one by one, starting from the **last digit** and working backward.

Each friend starts with **10 power points**.

Game Rules:

1. Jessy goes first, followed by Sarah, and then Oliver.
2. The player must guess the last digit of the Devil's life number.
   - **If correct** → The last digit is removed, shortening the number.
   - **If incorrect** → The player's power decreases by the absolute difference between their guess and the actual digit.
3. If a player's power reaches **zero**, the **player dies**, and the next friend continues the guessing.
4. If the Devil's **number is completely removed** (all digits guessed correctly), the **friends win**.
5. **If all three friends die before the number reaches zero, the Devil wins.**

Now, write a Java program that takes a 5-digit number (the Devil's life force) and sequential guesses from Jessy, Sarah, and Oliver. The program should:

- Deduct points for incorrect guesses.
- Remove digits from the Devil's number for correct guesses.
- Continue until the number is fully removed (friends win) or all friends lose power (Devil wins).
- Prints the final result and whenever any player dies.

Input	Output
Devil's Number: 54321 Jessy: 1 2 3 4 5	Congratulations! The Devil has been defeated!
Devil's Life: 98765 Jessy:	Output: Jessy has died!

8 5 2 1 Sarah: 6 2 3 6 Oliver: 8 7 8 1 3	Sarah has died! Oliver has died! The Devil wins! Better luck next time!

Explanation:

Test case 1:
Jessy goes first.
Turn 1:Jessy guesses 1 , correct; Power = 10 ; Devil's number: 5432
Turn 2:Jessy guesses 2 , correct; Power = 10 ; Devil's number: 543
Turn 3:Jessy guesses 3 , correct; Power = 10 ; Devil's number: 54
Turn 4:Jessy guesses 4 , correct; Power = 10 ; Devil's number: 5
Turn 5:Jessy guesses 5 , correct; Power = 10 ; Devil's number: 0

Test Case 2 :
Jessy goes first.
Turn 1:Jessy guesses 8 , Incorrect; Power = 7 ; Devil's number: 98765
Turn 2:Jessy guesses 5 , correct; Power = 7 ; Devil's number: 9876
Turn 3:Jessy guesses 2 , Incorrect; Power = 3 ; Devil's number: 9876
Turn 4:Jessy guesses 1 , Incorrect; Power = -2 ; Devil's number: 9876
Since, power less than 0, Jessy Dies.

Now, Sarah's turn.
Turn 1:Sarah guesses 6 , correct; Power = 10 ; Devil's number: 987
Turn 2:Sarah guesses 2 , correct; Power = 5 ; Devil's number: 987
Turn 3:Sarah guesses 3 , Incorrect; Power = 1 ; Devil's number: 987
Turn 4:Sarah guesses 6 , Incorrect; Power = 0 ; Devil's number: 987
Since power is 0, Sarah dies.

Now, Oliver's turn.
Turn 1:Oliver guesses 8 , Incorrect; Power = 9 ; Devil's number: 987

Turn 2:Oliver guesses 7 , correct; Power = 9 ; Devil's number: 98
Turn 3:Oliver guesses 8 , correct; Power = 9 ; Devil's number: 9
Turn 4:Oliver guesses 1 , Incorrect; Power = 1 ; Devil's number: 98
Turn 5:Oliver guesses 3 , Incorrect; Power = -5 ; Devil's number: 98
Since the power is less than 0, Oliver dies.

The Devil wins.

Q1. Harry Potter and Voldemort are engaged in the final battle. Voldemort has hidden his soul in five Horcruxes. Each time a Horcrux is destroyed, Harry and Voldemort's power levels are affected. Harry gains power, and Voldemort loses power. If Harry fails to destroy a Horcrux, he loses power, and Voldemort gains power. The winner of the battle is determined by their final power levels, with some additional conditions.

- Harry starts with **H** power points.
- Voldemort starts with **V** power points.
- **Horcrux Destruction**:
    - For each Horcrux that is destroyed:
        - Harry gains **50 points**.
        - Voldemort loses **40 points**.
- **Horcrux Failure**:
    - For each Horcrux that is **not destroyed**:
        - Harry loses **20 points**, multiplied by the number of Horcruxes Harry has attempted so far (starting from 1 for the first Horcrux, 2 for the second, and so on).
        - Voldemort gains **40 points**.
- If at any point during the battle, either **Harry's or Voldemort's power reaches 0**, the other character wins automatically without proceeding to the final battle.
- **Final Battle**:
    - Harry wins if his power is greater than Voldemort's power at the end of the battle.
    - If Harry's power is less than Voldemort's power, Harry can still win **if he has the Elder Wand** and at least one Horcrux is destroyed.

Now, write a Java program where you take the following inputs:

1. The initial power values for Harry (**H**) and Voldemort (**V**).
2. Five boolean inputs represent whether each Horcrux was destroyed (true/false).

3. A boolean value indicating if Harry has the Elder Wand.

The program should print the winner of the battle.

Input	Output
Harry's Power: 200 Voldemort's Power: 300 Horcrux Destruction: true true false false False Has Elder Wand: false	Winner: Voldemort
Harry's Power: 300 Voldemort's Power: 250 Horcrux Destruction: false false false True true Has Elder Wand: True	Winner: Harry

Explanation:

Test Case 1:

Harry's power = 200; Voldemort's Power: 300
First Horcrux: true ; Harry's power = 250 ; Voldemort's Power: 260
Second Horcrux: true ; Harry's power = 300 ;Voldemort's Power: 220
Third Horcrux: false ; Harry loses (20*3)=60 points ; Harry's power = 240 ;
Voldemort's Power: 260
Fourth Horcrux: false; Harry loses (20*4)=80 points ; Harry's power = 160;
Voldemort's Power: 300
Fifth Horcrux: false; Harry loses (20*5)=100 points; Harry's power = 60
Voldemort's Power: 340
Harry's Final Power: 60
Voldemort's Final Power: 340
Hence, Voldemort wins.

Test Case 2:

Harry's power = 300 ; Voldemort's Power: 250
First Horcrux: false; Harry loses (20*1) = 20 points ;Harry's power = 280;
Voldemort's Power: 290
Second Horcrux: false ; Harry loses (20*2) = 40 points ;Harry's power = 240

Voldemort's Power: 330
Third Horcrux: false ; Harry loses (20*3)=60 points ; Harry's power = 180
Voldemort's Power: 370
Fourth Horcrux: true ; Harry's power = 230 ; Voldemort's Power: 330
Fifth Horcrux: true ; Harry's power = 280; Voldemort's Power: 290
Harry's Final Power: 280
Voldemort's Final Power: 290
Since Harry has the Elder Wand and at least one Horcrux was destroyed, Harry wins.

Q2. Write a Java program that simulates a golf game. In this game, your goal is to hit the ball into the hole using the fewest possible strokes. The hole is located at coordinates (X=5, Y=5) on a 2D plane, and you start at the origin (0, 0). On each turn, you take a **shot** by specifying **how far (x, y) the ball moves** from its current position. Then, the program **calculates the Euclidean distance** between the ball's new position and the hole.

Now, the game **ends in one of three ways**:

1. If the ball **lands exactly** at (X, Y), print:
   "Congratulations! You hit the hole!" and end the game.

2. If the ball **crosses the hole,** calculate how far it **overshot.** Print **"You crossed the hole in the X/Y axis!"** and continue playing.

3. If after **5 turns** the hole is still not reached, print:
   "Game over! The hole is still [distance] units away."

You will take the X, Y coordinates of each shot as input. The program will calculate the new position of the ball and print the distance from the hole along with the proper statements according to the given constraints.

Euclidean Distance Calculation: $\sqrt{(x - x_1)^2 + (y - y_1)^2}$

Input	Output	Explanation
Shot 1: X = 2 Y = 2 **Shot 2:** X = 1 Y = 1 **Shot 3:** X = 2 Y = 2	New position: (2, 2) Distance from hole: 4.24 units New position: (3, 3) Distance from hole: 2.83 units Congratulations! You hit the hole!	Initially, **x_ball = 0; y_ball= 0** (coordinates of the ball) **Shot 1:** X = 2 ; Y = 2 So, x_ball = 2; y_ball= 2 Didn't hit the hole. Distance = 4.24 units **Shot 2:** X = 1 ; Y = 1 So, x_ball = 3; y_ball= 3 Didn't hit the hole. Distance = 2.83 Units **Shot 3:** X = 1 ; Y = 1 So, x_ball = 5; y_ball= 5 **Hit the hole.**
Shot 1: X = 6 Y = 3	New position: (6, 3) You crossed the hole in the X-axis!	Initially, **x_ball = 0; y_ball= 0** (coordinates of the ball)

Shot 2: X = -1 Y = 2	Distance from hole: 1.41 units Congratulations! You hit the hole!	**Shot 1:** X = 6 ; Y = 3 So, x_ball = 6; y_ball= 3 Didn't hit the hole. Crossed the X axis. Distance = 1.41 units **Shot 2**: X = -1 ; Y = 2 So, x_ball = 5; y_ball= 5 **Hit the hole.**
Shot 1: X = 1 Y = 1 **Shot 2:** X = 1 Y = 1 **Shot 3:** X = 1 Y = 1 **Shot 4:** X = 1 Y = 1 **Shot 5:** X = 1 Y = 0	New position: (1, 1) Distance from hole: 5.66 units New position: (2, 2) Distance from hole: 4.24 units New position: (3, 3) Distance from hole: 2.83 units New position: (4, 4) Distance from hole: 1.41 units New position: (5, 4) Distance from hole: 1.00 units Game over! The hole is still 1.00 units away.	Initially, **x_ball = 0; y_ball= 0** (coordinates of the ball) **Shot 1:** X = 1 ; Y = 1 So, x_ball = 1; y_ball= 1 Didn't hit the hole. Distance = 5.66 units **Shot 2**: X = 1 ; Y = 1 So, x_ball = 2; y_ball= 2 Didn't hit the hole. Distance = 4.24 Units **Shot 3**: X = 1 ; Y = 1 So, x_ball = 3 ; y_ball= 3 Didn't hit the hole. Distance = 2.83 Units **Shot 4:** X = 1 ; Y = 1 So, x_ball = 4 ; y_ball= 4 Didn't hit the hole. Distance = 1.41 Units **Shot 5:** X = 1 ; Y = 0 So, x_ball = 5 ; y_ball= 4 Didn't hit the hole. Distance = 1.00 Units **All five shots over, hence, the game is over.**

Q3. You are organizing an Archery Tournament where players shoot arrows at a target with N rings. Each player shoots M arrows, and the scores for each shot correspond to the ring number that the arrow lands on. The challenge lies in a dynamic bullseye system that changes in a circular manner, and you need to calculate the total score for each player and finally determine the winner.

Bullseye calculation :

In this game, the bullseye changes in a circular fashion after every shot. The bullseye starts at ring 1; after the first shot, it moves to ring 2, then to ring 3, and so on until it reaches ring N. After that, it loops back to ring 1.

Score Calculation:

- If the ring number (the shot the player hits) is **less than** the current bullseye, the score is calculated as follows:
  score = (ring number/bullseye number) * 100
- If the ring number (the shot) is **greater than or equal to** the bullseye, the score is calculated as follows:
  score = (bullseye number/ring number) * 100
- The score is then rounded down (i.e., use the floor value).
- The total score for each player is the sum of the scores of all their shots.

Additionally, you need to count how many times a player hits the bullseye during the game.

Final Winner Determination:

- First, you compare the total scores of all players.
- If there is a tie in the highest score, we check the number of bullseyes. The player with the most bullseyes wins.
- If there is still a tie in both the total score and the number of bullseyes, pick any one as the winner.

Now, write a Java program for this game, where you will take **N** (number of rings), **M** (number of shots per player), and **P** (number of players) as input. Then, in the next lines, you will take an integer X as input that indicates where the shot landed. Finally, you will print the total score and bullseye count for each player. Print the **winner's name** based on the ranking criteria.

Input	Output
Enter number of players (N): 3 Enter number of rings (M): 5 Enter number of arrows per player (P): 4 Player 1, enter your shots: X = 2 X = 4 X = 3 X = 3 Player 2, enter your shots: X = 1 X = 1 X = 1 X = 1 Player 3, enter your shots: X = 5 X = 4 X = 3 X = 3	Player 1: Total score: 275, Bullseyes: 1 Player 2: Total score: 203, Bullseyes: 1 Player 3: Total score: 260, Bullseyes: 0 Winner: Player 1

Explanation:

- **Player 1:**

  - Shot 1: Ring 2, Bullseye 1 → score = 1 / 2 = 0.50 * 100 = 50
  - Shot 2: Ring 4, Bullseye 2 → score = 2 / 4 = 0.50 * 100 = 50
  - Shot 3: Ring 3, Bullseye 3 → score = 3 / 3 = 1.00 * 100 = 100 (bullseye!)
  - Shot 4: Ring 3, Bullseye 4 → score = 3 / 4 = 0.75 * 100 = 75
  - **Total score: 50 + 50 + 100 + 75 = 275, Bullseyes: 1**
- **Player 2:**

  - Shot 1: Ring 1, Bullseye 5 → score = 1 / 5 = 0.2 * 100 = 20
  - Shot 2: Ring 1, Bullseye 1 → score = 1 / 1 = 1 * 100 = 100 (bullseye!)
  - Shot 3: Ring 1, Bullseye 2 → score = 1 / 2 = 0.5 * 100 = 50
  - Shot 4: Ring 1, Bullseye 3 → score = 1 / 3 = 0.33 * 100 = 33
  - **Total score: 20 + 100 + 50 + 33 = 203, Bullseyes: 1**
- **Player 3:**

  - Shot 1: Ring 5, Bullseye 4 → score = 4 / 5 = 0.8 * 100 = 80
  - Shot 2: Ring 4, Bullseye 5 → score = 4 / 5 = 0.8 * 100 = 80
  - Shot 3: Ring 3, Bullseye 1 → score = 1 / 3 = 0.33 * 100 = 33
  - Shot 4: Ring 3, Bullseye 2 → score = 2 / 3 = 0.67 * 100 = 67
  - **Total score: 80 + 80 + 33 + 67 = 260, Bullseyes: 0**

Winner:

- Player 1 has the highest total score of 275, so they win.