LEVEL - 1

1. ACME Corporation wants to implement a bonus system for its employees. The bonus calculation is based on the employee's salary and gender.

   Let's denote the following variables:

   salary: Salary of the employee.
   gender: Gender of the employee
   bonus: bonus of the employee

   The bonus policy is as follows:

   - If salary < Rs. 25000, bonus is capped at Rs. 5000 for both male (M) and female (F) employees.
   - If salary is between Rs. 25000 and Rs. 50000, bonus is capped at Rs. 7500 for both genders. [Inclusion of boundary values]
   - If salary > Rs. 50000, calculates the bonus as 10% of salary for male (M) and 15% for female (F).

Write a program that takes an employee's salary and gender as input and calculates the bonus amount.

Constraints:

1<= Salary <= 2^31 - 1
Gender should consist of uppercase English letter ‘M’ or ‘F’
Gender.length = 1

Test Case 1:
Salary < Rs. 25000, Gender = ‘M’
Expected Output: Bonus = Rs. 5000
Test Case 2:
Salary = Rs. 25000, Gender = ‘M’
Expected Output: Bonus = Rs. 7500
Test Case 3:
Salary = Rs. 50000, Gender = ‘M’
Expected Output: Bonus = Rs. 7500
Test Case 4:
Salary = Rs. 75000, Gender = ‘F’
Expected Output: Bonus = Rs. 11250
Test Case 5:
Salary = Rs. 100000, Gender = ‘M’
Expected Output: Bonus = Rs. 10000

Test Case 6:
Salary = -10000, Gender = ‘M’
Expected Output: Bonus = Rs. 0 *(Invalid salary, no bonus)*
Test Case 7:
Salary = 0, Gender = ‘F’
Expected Output: Bonus = Rs. 0 *(No bonus for zero salary)*
Test Case 8:
Salary = Rs. 24999, Gender = ‘M’
Expected Output: Bonus = Rs. 5000
Test Case 9:
Salary = Rs. 25001, Gender = ‘F’
Expected Output: Bonus = Rs. 7500
Test Case 10:
Salary = Rs. 50001, Gender = ‘M’
Expected Output: Bonus = Rs. 5000.10
Test Case 11:
Salary = Rs.15000, Gender = “m”
Expected Output: Gender Should be ‘M’ for Male and ‘F’ for Female
Test Case 12:
Salary = Rs.15000, Gender = “H”
Expected Output: Gender Should be ‘M’ for Male and ‘F’ for Female

2. ABCDE Bank offers a savings account with a compound interest rate that can be set by the user. The interest is compounded annually. The bank allows customers to deposit money for a specified time period (T), which can be in years or months. Write a program that calculates the Compound Interest (A) for a given principal amount (P), interest rate (R), and time period.

$$A = P\left(1 + \frac{R}{100}\right)^T$$

Constraints:

1< = P < = 2^31 - 1

1.0 <= R < = 8.5

T → T_{years} (In terms of Years)

Test Case 1: Minimum Values

Input: P = 1, R = 1.0%, T = 1 year

Output: A = 1.01

Test Case 2: Boundary Condition - Maximum Values

Input: P = 2147483647, R = 8.5%, T = 1 year

Output: A = 2330019756.99

Test Case 3: Whole Year - Simple Case

Input: P = 1000, R = 5.0%, T = 2 years

Output: A =1102.50

Test Case 4: Fractional Year - Converted to Months

Input: P = 5000, R = 6.5%, T = 3.5 years

Output: A = 6039.75

Test Case 5: Low Interest Rate

Input: P = 3000, R = 1.5%, T = 4 years

Output: A = 3184.09

Test Case 6: High Interest Rate

Input: P = 1500, R = 8.0%, T = 2.5 years

Output: A = 1749.60

Test Case 7: Large Principal Amount

Input: P = 1000000, R = 4.2%, T = 3 years

Output: A = 1131366.09

Test Case 8: Long Time Period

Input: P = 2000, R = 3.75%, T = 10 years

```
Output: A = 2890.09
```

Test Case 9: Short Time Period

Input: P = 5000, R = 6.0%, T = 0.5 years

```
Output: A = 5000.00
```

Test Case 10: Fractional Interest Rate

Input: P = 8000, R = 2.25%, T = 2 years

```
Output: A = 8364.05
```

Test Case 11: Invalid Interest Rate

Input: P = 4000, R = 9.25%, T = 2 years

```
Output: Error: Interest rate must be between 1.0 and 8.5.
```

Test Case 12: Invalid Principal amount

Input: P = 0, R = 7%, T = 2 years

```
Output: Error: Principal amount must be between 1 and 2^31 - 1.
```

3. Hello Corporation wants to implement a financial tracking system for its employees. The system should be able to calculate the total salary, expenditure, savings, and savings percentage for each employee based on the following criteria:

- S_{base}: Base salary of the employee.
- S_{bonus}: Bonus received by the employee.
- $S_{allowance}$: Allowances earned by the employee.
- E: Total expenditure by the employee on various items (rent, utilities, groceries, entertainment, etc.).
- S_{total}: Total salary of the employee, which includes base salary, bonuses, and allowances. [$S_{total}=S_{base}+S_{bonus}+ S_{allowance}$]
- S: Savings of the employee, calculated as the difference between total salary and expenditure.[$S=S_{total}-E$]
- P: Savings percentage, representing the portion of total salary saved.

$$P = \left(\frac{S}{S_{total}}\right) * 100$$

Write a program that takes an employee's base salary, bonuses, allowances, and expenditure as input, and calculates the total salary, expenditure, savings, and savings percentage.

Constraints:

$0 <= S_{base} <= 2^{31} - 1$
$0 <= S_{bonus} <= 2^{31} - 1$
$0 <= S_{allowance} <= 2^{31} - 1$
$0 <= E <= 2^{31} - 1$

Test Case 1: Minimum values
Input: $S_{base} = 0$, $S_{bonus} = 0$, $S_{allowance} = 0$, $E = 0$
Expected Output:
$S_{total} = 0$ $(0 + 0 + 0)$
$S = 0$ $(0 - 0)$
P = undefined

Test Case 2: Only base salary
Input: $\mathbf{S_{base}} = 15000$, $S_{bonus} = 0$, $S_{allowance} = 0$, $E = 5000$
Expected Output:
$S_{total} = 15000$
$S = 10000$
$P = 66.67\%$

Test Case 3: Only bonuses and allowances
Input: $S_{base} = 0$, $S_{bonus} = 5000$, $S_{allowance} = 3000$, $E = 2000$
Expected Output:
$S_{total} = 8000$
$S = 6000$
$P = 75\%$

Test Case 4: All components nonzero
Input: $S_{base} = 25000$, $S_{bonus} = 7500$, $S_{allowance} = 2000$, $E = 15000$
Expected Output:
$S_{total} = 34500$
$S = 19500$
$P = 56.52\%$ (19500 / 34500 * 100)

Test Case 5: Maximum values
Input: $S_{base} = 2147483647$ (2^31 - 1), $S_{bonus} = 2147483647$ (2^31 - 1),
$S_{allowance} = 2147483647$ (2^31 - 1), $E = 2147483647$ (2^31 - 1)
Expected Output:
$S_{total} = 6442450941$
$S = 4294967294$
$P = 66.67\%$

Test Case 6: Base salary at maximum, others zero
Input: $S_{base} = 2147483647$, $S_{bonus} = 0$, $S_{allowance} = 0$, $E = 0$
Expected Output:
$S_{total} = 2147483647$
$S = 2147483647$
$P = 100\%$

Test Case 7: Random values
Input: $S_{base} = 50000$, $S_{bonus} = 10000$, $S_{allowance} = 3000$, $E = 20000$
Expected Output:
$S_{total} = 63000$
$S = 43000$
$P = 68.25\%$

Test Case 8: Base salary and bonuses only
Input: $S_{base} = 30000$, $S_{bonus} = 15000$, $S_{allowance} = 0$, $E = 10000$
Expected Output:
$S_{total} = 45000$
$S = 35000$
$P = 77.78\%$

Test Case 9: No negative bonuses or allowances
Input: $S_{base} = 40000$, $S_{bonus} = -1$, $S_{allowance} = -10$, $E = 15000$
Expected Output:
$S_{total} = 40000$
$S = 25000$
$P = 62.5\%$

Test Case 10: Equal expenditure to total salary
Input: $S_{base} = 20000$, $S_{bonus} = 5000$, $S_{allowance} = 2000$, $E = 27000$
Expected Output:

$S_{total} = 27000$
$S = 0$
$P = 0\%$

Test Case 10: Equal expenditure to total salary
Input: $S_{base} = 20000$, $S_{bonus} = 5000$, $S_{allowance} = 2000$, $E = 27000$
Expected Output:

$S_{total} = 27000$
$S = 0$
$P = 0\%$

4. Imagine you are running a small business and you need to keep track of your transactions to determine if you are making a profit or a loss on each item you sell. To automate this process, you decide to write a program that will help you quickly identify whether you made a profit, incurred a loss, or broke even on a sale.

Let's denote the following variables:

- C: Cost price of the item (what you paid to acquire the item).
- S: Selling price of the item (what you sold the item for).

The profit (P) or loss (L) on a sale can be calculated using the formula:

$$P/L=S-C$$

The nature of the transaction is as follows:

- **Profit (P>0)**: You made a profit on the sale.
- **Loss (P<0)**: You incurred a loss on the sale.
- **Break-even (P=0)**: You neither made a profit nor incurred a loss; you broke even.

Constraints:

C>=0 , non-negative value (since you cannot have a negative cost price)
S<=0, non-negative, (since you cannot sell something for less than zero money).

Test Case 1: Minimum values - Break-even case
Input: C = 0 (minimum cost price), S = 0 (minimum selling price)
Expected Output:

- P = 0
- Result: Break-even

Test Case 2: Cost price only, no selling price
C = 100 (typical cost price), S = 0 (no selling price)
Expected Output:

- P = -100
- Result: Loss

Test Case 3: Selling price slightly higher than cost price
Input: C = 200, S = 220
Expected Output:

- P = 20
- Result: Profit

Test Case 4: Equal cost and selling prices
Input: C = 150, S = 150
Expected Output:

- P = 0
- Result: Break-even

Test Case 5: Selling price exactly zero
Input: C = 180, S = 0
Expected Output:

- P = -180
- Result: Loss

Test Case 6: Cost price exactly zero
Input: C = 0, S = 50
Expected Output:

- P = 50
- Result: Profit

Test Case 7: Maximum values - Large profit
Input: C = 99999, S = 199999
Expected Output:

- P = 100000
- Result: Profit

Test Case 8: Large cost price, zero selling price
Input: C = 1000000, S = 0
Expected Output:

- P = -1000000
- Result: Loss

Test Case 9: Cost price close to maximum, minimal selling price
Input: C = 2147483647 (maximum value for a 32-bit signed integer), S = 1
Expected Output:

- P = -2147483646
- Result: Loss

Test Case 10: Selling price close to maximum, minimal cost price
Input: C = 1, S = 2147483647
Expected Output:

- P = 2147483646
- Result: Profit

Test Case 11: Selling price close to maximum, negative cost price
Input: C = -1, S = 2147483647
Expected Output:

- Error: Cost price (C) cannot be negative.

5. You are developing a voter eligibility verification system for a government agency. The system should determine if a person is eligible to vote based on their age and display their eligibility status with a proper salutation for the name based on their gender.
Let's denote the following variables:

- **name:** Name of the person
- **age:** Age of the person.
- **gender:** Gender of the person

The eligibility criteria can be defined as follows:

- Minimum Age (age_{min}): The minimum age required to vote. (Minimum age to vote: 18 years)

The salutation based on gender can be determined using the gender variable G:

- If gender=1: Salutation is "Mr."
- If gender=2: Salutation is "Ms."

Constraint:

- age > 0
- gender=1 for male, gender=2 for female

Test Case 1: Minimum age, male
Input:
```
name = "John"
age = 18
gender = 1
```
Expected Output:
```
Eligibility Status: Eligible
Salutation: "Mr. John"
```

Test Case 2: Minimum age, female
Input:
```
name = "Emma"
age = 18
gender = 2
```
Expected Output:
```
Eligibility Status: Eligible
Salutation: "Ms. Emma"
```

Test Case 3: Just eligible age, male
Input:
```
name = "Michael"
age = 19
gender = 1
```
Expected Output:
```
Eligibility Status: Eligible
Salutation: "Mr. Michael"
```

Test Case 4: Just eligible age, female
Input:
name = "Sophia"
age = 19
gender = 2
Expected Output:
Eligibility Status: Eligible
Salutation: "Ms. Sophia"

Test Case 5: Age, Wrong gender
Input:
name = "Robert"
age = 60
gender = 3
Expected Output:
Please type a valid gender

Test Case 6: Age, Negative gender
Input:
name = "Alice"
age = 150
gender = -2
Expected Output:
Please type a valid gender

Test Case 7: Below minimum age, male
Input:
name = "Tom"
age = 17
gender = 1
Expected Output:
Eligibility Status: Not Eligible
Salutation: None (or handle as per system design)

Test Case 8: Below minimum age, female
Input:
name = "Lily"
age = 17
gender = 2
Expected Output:
Eligibility Status: Not Eligible
Salutation: None (or handle as per system design)

Test Case 9: Zero age (Invalid age), male
Input:

name = "James"
age = 0
gender = 1

Expected Output:

Eligibility Status: Not Eligible
Salutation: None (or handle as per system design)

Test Case 10: Negative age (Invalid age) , female

Input:

name = "Sarah"
age = -25
gender = 2

Expected Output:

Eligibility Status: Not Eligible
Salutation: None (or handle as per system design)

6. The school administration wants to calculate the attendance percentage for each student. They need a program that can take the following information as input:
   1. Student's name
   2. Student's roll number
   3. Total number of classes
   4. Number of classes attended

The program should then calculate the attendance percentage for the student and display the result.

Let's denote the following variables:

- **name**: Student's name.
- **roll_no**: Student's roll number.
- **total_class**: Total number of classes.
- **no_of_attended_class**: Number of classes attended.

The attendance percentage (P) for the student can be calculated using the formula:

$$P = \left(\frac{A}{T}\right) * 100$$

Constraints:

- name ≠ empty string (name : a non-empty string)
- roll_no > 0 (a positive integer)
- total_class>0 (a positive integer.)
- 0<= no_of_attended_class <= total_class (Number of classes attended should be a non-negative integer and should not exceed T)
- $0 \leq P \leq 100$

Test Case 1:
Input:
name: "John Doe"
roll_no: 101
total_class: 20
no_of_attended_class: 15
Expected Output:
Attendance Percentage: 75%
Test Case 2:
Input:

name: "Jane Smith"
roll_no: 202
total_class: 25
no_of_attended_class: 20

Expected Output:

Attendance Percentage: 80%

Test Case 3:

Input:

name: "Michael Johnson"
roll_no: 303
total_class: 30
no_of_attended_class: 0

Expected Output:

Attendance Percentage: 0%

Test Case 4:

Input:

name: "Emily Brown"
roll_no: 404
total_class: 10
no_of_attended_class: 10

Expected Output:

Attendance Percentage: 100%

Test Case 5:

Input:

name: "David Lee"
roll_no: 505
total_class: 15
no_of_attended_class: 7

Expected Output:

Attendance Percentage: 46.67%

Test Case 6:

Input:

name:

Expected Output:

Name cannot be empty. (Program terminated)

Test Case 7:

Input:

name: "Oliver Taylor"

roll_no: 707

total_class: 12

no_of_attended_class: 6

Expected Output:

Attendance Percentage: 50%

Test Case 8:

Input:

name: "Ava Martinez"

roll_no: 808

total_class: 22

no_of_attended_class: 25

Expected Output:

Number of attended class cannot be greater than total number of class

Test Case 9:

Input:

name: "Noah Hernandez"

roll_no: 909

total_class: -24

no_of_attended_class: 12

Expected Output:

Invalid total number of class

Test Case 10:

Input:

name: "Isabella Gonzalez"

roll_no: 1010

total_class: 16

no_of_attended_class: -8

Expected Output:

Invalid number of class attended

7. You're planning a trip and found a train schedule with departure times listed in railway time (HHMM) format. You're more familiar with the 12-hour clock (HH:MM AM/PM).Develop a program that can take a railway time as input and convert it to normal time (including AM or PM). This will help you easily understand the departure times on the train schedule.The program can also handle situations like noon (12:00) being displayed as either 1200 or 0000 in railway time.

   Let's denote the following variables:

   - $T_{railway}$: Time in railway format (HHMM).
   - $T_{12\text{-}hour}$: Time in 12-hour format (HH AM/PM).

   The conversion process involves:

   1. Extracting hours (HH) and minutes (MM) from $T_{railway}$.
   2. Converting the 24-hour format to 12-hour format with appropriate AM/PM designation.

   Constraints:

   - $0<=T_{railway}<=2359$ [$T_{railway}$ should be a four-digit number representing time in the range from 0000 to 2359]
   - 0<=HH<=23 [Hours (HH) should be between 00 and 23 in railway time]
   - 0<=MM<=59 [Minutes (MM) should be between 00 and 59]

   These constraints ensure that the input railway time is valid and within the expected range for a typical 24-hour clock representation. The program should handle special cases like:
   - Midnight (00:00 or 2400 in railway time).
   - Noon (12:00).

   Test Case 1:
   Input:
   Railway Time: 0
   Expected Output:
   12-Hour Time: 12:00 AM
   Test Case 2:
   Input:
   Railway Time: 1200

Expected Output:

  12-Hour Time: 12:00 PM

Test Case 3:

Input:

  Railway Time: 2400

Expected Output:

  Invalid railway time input.

Test Case 4:

Input:

  Railway Time: 2359

Expected Output:

  12-Hour Time: 11:59 PM

Test Case 5:

Input:

  Railway Time: 100

Expected Output:

  12-Hour Time: 1:00 AM

Test Case 6:

Input:

  Railway Time: 815

Expected Output:

  12-Hour Time: 8:15 AM

Test Case 7:

Input:

  Railway Time: 1215

Expected Output:

  12-Hour Time: 12:15 PM

Test Case 8:

Input:

  Railway Time: 1300

Expected Output:

  12-Hour Time: 1:00 PM

Test Case 9:

Input:

  Railway Time: 1545

Expected Output:

  12-Hour Time: 3:45 PM

Test Case 10:
Input:
    Railway Time: 2350
Expected Output:
    12-Hour Time: 11:50 PM
Test Case 11:
Input:
    Railway Time: 1263
Expected Output:
    Invalid railway time input.

8. You're running an online store with limited stock on a particular item. To prevent overselling, you want to automate order confirmations.Develop a program that acts as an auto-reply system for incoming orders. It takes the order quantity (x) as an integer input from the user.If the order quantity (x) is less than 70 (available stock), print "Order Confirmed".Otherwise, print "Order Limit Reached".In both cases, the program should thank the customer with a separate line that says "Thank YOU!".

Constraints:

$x > 0$.
$0 < x < 70 \leftarrow$ Order Confirmation
$x >= 70 \leftarrow$ Order Limit Reached
In either case, the program should print a thank you message.

Test Case 1:
Input: x = -1
Expected Output: Order has no proper quantity specified. Thank YOU!
Test Case 2:
Input: x = 0
Expected Output: Order has no proper quantity specified. Thank YOU!
Test Case 3:
Input: x = 1
Expected Output: Order Confirmed. Thank YOU!
Test Case 4:
Input: x = 69
Expected Output: Order Confirmed. Thank YOU!
Test Case 5:
Input: x = 70
Expected Output: Order Limit Reached. Thank YOU!
Test Case 6:
Input: x = 71
Expected Output: Order Limit Reached. Thank YOU!
Test Case 7:
Input: x = 100
Expected Output: Order Limit Reached. Thank YOU!
Test Case 8:
Input: x = 24
Expected Output: Order Confirmed. Thank YOU!

Test Case 9:
Input: x = 2
Expected Output: Order Confirmed. Thank YOU!
Test Case 10:
Input: x = 150
Expected Output: Order Limit Reached. Thank YOU!

9. Imagine you're a developer working on a new mobile banking app that allows users to withdraw cash easily. To optimize ATM dispensing and minimize the number of notes dispensed, you want to implement a feature that calculates the minimum number of notes needed for a specific withdrawal amount.

Let's denote the following variables:

A → withdrawal amount (in the currency unit).

n_i → the quantity of each denomination i of notes used.

D_i → the denomination value of each note i.

Constraints:

$$\sum n_i \cdot D_i = A$$

where n_i are non-negative integers representing the number of notes of denomination D_i used, and $\Sigma n_i \cdot D_i$ equals A, the specified withdrawal amount.

Denomination includes: 500, 100,50,20,10,5,2,1

Use the largest denomination first

Test Case 1:

Input: A = -50

Expected Output: Invalid amount

Test Case 2:

Input: A = 0

Expected Output: No cash needed

Test Case 3:

Input: A = 1

Expected Output: {1: 1}

Test Case 4:

Input: A = 50

Expected Output: {50: 1}

Test Case 5:

Input: A = 75

Expected Output: {50: 1, 20: 1, 5: 1}

Test Case 6:

Input: A = 100

Expected Output: {100: 1}

Test Case 7:

Input: A = 380

Expected Output: {100: 3, 50: 1, 20: 1, 10: 1}

Test Case 8:

Input: A = 999

Expected Output: {500:1, 100: 4, 50: 1, 20: 2, 5: 1, 2: 2}

Test Case 9:

Input: A = 1234

Expected Output: {500: 2, 100: 2, 20: 1, 10: 1, 2: 2}

Test Case 10:

Input: A = 5000

Expected Output: {500: 10}

10. Develop a program for a payroll system that calculates the gross salary of an employee based on their basic salary and allowances. The allowances (HRA and DA) vary depending on the basic salary range. Here are the rules for calculating the allowances:

- If the basic salary is up to Rs. 10,000, the allowances are: HRA = 20% of basic salary, DA = 80% of basic salary.
- If the basic salary is between Rs. 10,001 and Rs. 20,000, the allowances are: HRA = 25% of basic salary, DA = 90% of basic salary.
- If the basic salary is Rs. 20,001 or more, the allowances are: HRA = 30% of basic salary, DA = 95% of basic salary.

Constraints:

0< basic_salary <=2^31 - 1

Test Case 1:
Input: basic_salary = -1000
Expected Output: Invalid salary
Test Case 2:
Input: basic_salary = 0
Expected Output: Invalid salary
Test Case 3:
Input: basic_salary = 1
Expected Output: Gross Salary = 2.00
Test Case 4:
Input: basic_salary = 10,000
Expected Output: Gross Salary = 20000.00
Test Case 5:
Input: basic_salary = 10,001
Expected Output: Gross Salary = 21502.15
Test Case 6:
Input: basic_salary = 15,000
Expected Output: Gross Salary = 32250.00
Test Case 7:
Input: basic_salary = 20,000
Expected Output: Gross Salary = 43000.00
Test Case 8:
Input: basic_salary = 20,001

Expected Output: Gross Salary = 45002.25

Test Case 9:

Input: basic_salary = 50,000

Expected Output: Gross Salary = 112500.00

Test Case 10:

Input: basic_salary = 2,147,483,647 (maximum integer value)

Expected Output: Gross Salary = 4831838208.00

11. Develop a program for a utility company that calculates the electricity bill for customers based on their consumption of electricity units. The bill calculation should consider different rates per unit and apply a 20% surcharge to the final amount.

Let's denote the following variables:

- U → Number of units consumed.
- B → Bill amount before surcharge.
- S → Surcharge amount.
- T → Total bill amount after applying the surcharge.

The rules for calculating the bill can be expressed as:

1. For the first 50 units: Rs. 0.50 per unit
2. For the next 100 units (51 to 150): Rs. 0.75 per unit
3. For the next 100 units (151 to 250): Rs. 1.20 per unit
4. For units above 250: Rs. 1.50 per unit

After calculating the bill amount B, a 20% surcharge is applied: $S = 0.20 \times B$
$T = B + S$

Constraints
$0 <= U <= 2^{31} - 1$

Test Case 1:

Input: U = -10
Expected Output:
Bill amount before surcharge: Rs. 0.00
Surcharge amount: Rs. 0.00
Total bill amount after surcharge: Rs. 0.00
Reason: Negative units are invalid; hence, no charge is calculated.

Test Case 2:
Input: U = 0
Expected Output:

Bill amount before surcharge: Rs. 0.00
Surcharge amount: Rs. 0.00
Total bill amount after surcharge: Rs. 0.00
Reason: No units consumed; hence, no charge is calculated.

Test Case 3:
Input: U = 25
Expected Output:

Bill amount before surcharge: Rs. 12.50
Surcharge amount: Rs. 2.50
Total bill amount after surcharge: Rs. 15.00

Test Case 4:
Input: U = 50
Expected Output:

Bill amount before surcharge: Rs. 25.00
Surcharge amount: Rs. 5.00
Total bill amount after surcharge: Rs. 30.00

Test Case 5:
Input: U = 75
Expected Output:

Bill amount before surcharge: Rs. 43.75
Surcharge amount: Rs. 8.75
Total bill amount after surcharge: Rs. 52.50

Test Case 6:
Input: U = 150
Expected Output:

Bill amount before surcharge: Rs. 100.00
Surcharge amount: Rs. 20.00
Total bill amount after surcharge: Rs. 120.00

Test Case 7:
Input: U = 175
Expected Output:

Bill amount before surcharge: Rs. 130.00
Surcharge amount: Rs. 26.00
Total bill amount after surcharge: Rs. 156.00

Test Case 8:
Input: U = 250
Expected Output:

Bill amount before surcharge: Rs. 220.00
Surcharge amount: Rs. 44.00
Total bill amount after surcharge: Rs. 264.00

Test Case 9:
Input: U = 300
Expected Output:

Bill amount before surcharge: Rs. 295.00
Surcharge amount: Rs. 59.00
Total bill amount after surcharge: Rs. 354.00

Test Case 10:
Input: U = 400
Expected Output:

Bill amount before surcharge: Rs. 445.00
Surcharge amount: Rs. 89.00
Total bill amount after surcharge: Rs. 534.00

12. Imagine you're developing a mobile app for a geometry tutoring service that helps students learn about triangles. One feature of the app needs to validate whether a given set of three angles can form a valid triangle.
Let's denote the following variables:

A - First Angle of the triangle
B - Second Angle of the triangle
C - Third Angle of the triangle

Constraints:
$A>0$, $B>0$, $C>0$
$A + B + C = 180$ degree

Test Case 1:
Input: A = 60, B = 60, C = 60
Expected Output: Valid triangle
Test Case 2:
Input: A = 90, B = 45, C = 45
Expected Output: Valid triangle
Test Case 3:
Input: A = 0, B = 0, C = 180
Expected Output: Invalid triangle
Test Case 4:
Input: A = 89, B = 91, C = 0
Expected Output: Invalid triangle
Test Case 5:
Input: A = -10, B = 90, C = 100
Expected Output: Invalid triangle
Test Case 6:
Input: A = 120, B = 30, C = 30
Expected Output: Valid triangle
Test Case 7:
Input: A = 0, B = 90, C = 90
Expected Output: Invalid triangle
Test Case 8:
Input: A = 59, B = 60, C = 61
Expected Output: Valid triangle
Test Case 9:

Input: A = 180, B = 0, C = 0
Expected Output: Invalid triangle
Test Case 10:
Input: A = 45, B = 45, C = 90
Expected Output: Valid triangle

13. You are developing a software tool for architects to quickly determine the type of shape (square or rectangle) based on the dimensions provided for various building components. The tool also calculates and displays the area of each component.

Let's denote the following variables:

L - Length of the shape

W - Width of the shape

If L==W, then the shape is a square.

If L≠W, then the shape is a rectangle.

Constraints:

L>0

W>0

Test Case 1:

Input:

L = 1, W = 1

Expected Output:

Shape: Square

Area: 1

Test Case 2:

Input:

L = 2, W = 2

Expected Output:

Shape: Square

Area: 4

Test Case 3:

Input:

L = 3, W = 4

Expected Output:

Shape: Rectangle

Area: 12

Test Case 4:

Input:

L = 5, W = 3

Expected Output:

Shape: Rectangle

Area: 15

Test Case 5:
Input:
L = 0, W = 0
Expected Output:
Shape: Invalid
Area: Invalid
Test Case 6:
Input:
L = -1, W = 2
Expected Output:
Shape: Invalid (Length cannot be negative)
Area: Invalid
Test Case 7:
Input:
L = 4, W = -5
Expected Output:
Shape: Invalid (Width cannot be negative)
Area: Invalid
Test Case 8:
Input:
L = -3, W = -3
Expected Output:
Shape: Invalid (Length and Width cannot be negative)
Area: Invalid
Test Case 9:
Input:
L = 0, W = -1
Expected Output:
Shape: Invalid (Width cannot be negative)
Area: Invalid
Test Case 10:
Input:
L = -5, W = 0
Expected Output:
Shape: Invalid (Length cannot be negative)
Area: Invalid

14. You are developing a game where a player rolls a virtual dice (with numbers ranging from 1 to 6).

Let's denote the following variables:

N → an integer representing the number rolled on the dice.

D → Current position of a dice (a positive integer)

Based on the rolling the dice:

- If the dice rolled 'N' is odd, you want to multiply 'D' by 3 to determine how far the player can move forward.
- If the dice rolled 'N' is even, you want to divide 'D' by 3 to determine how far the player moves backward.

Print the Position of the player in a move.

Constraints:

1 <= N <= 6

D > 0

Decimal result to be rounded off to floor value.

Test Case 1:
Input: N = 1, D = 5
Expected Output: Player moves forward 15 units
Test Case 2:
Input: N = 2, D = 9
Expected Output: Player moves backward 3 units
Test Case 3:
Input: N = 3, D = 7
Expected Output: Player moves forward 21 units
Test Case 4:
Input: N = 4, D = 18
Expected Output: Player moves backward 6 units
Test Case 5:
Input: N = 5, D = 6
Expected Output: Player moves forward 18 units
Test Case 6:
Input: N = 6, D = 12
Expected Output: Player moves backward 4 units
Test Case 7:
Input: N = 0, D = 5
Expected Output: Invalid Dice Roll

Test Case 8:
Input: N = 8, D = 7
Expected Output: Invalid Dice Roll
Test Case 9:
Input: N = -4, D = 4
Expected Output: Invalid Dice Roll
Test Case 10:
Input: N = 4, D = -15
Expected Output: Invalid Position of the Player

15. Each credit card has a credit limit. A transaction is attempted on the credit card. The transaction amount is checked against the available credit limit. If the transaction amount is less than or equal to the available credit limit, the transaction is approved. If the transaction amount exceeds the available credit limit, the transaction is declined. Hence, develop a system to check if a transaction can be approved based on the available credit limit of a user.

Let's denote the following variables:

$C \rightarrow$ Credit Limit

$T \rightarrow$ Transaction Amount

Conditions:

Transaction Approval: $T <= C$

Transaction Decline: $T > C$

Constraints:

- $C>0$
- $T>0$

Test Case 1:
Input: $C=1000$, $T = -100$
Expected Output: Invalid transaction amount
Test Case 2:
Input: $C = -500$, $T=200$
Expected Output: Invalid credit limit
Test Case 3:
Input: $C=1000$, $T=0$
Expected Output: Invalid transaction amount
Test Case 4:
Input: $C=0$, $T=100$
Expected Output: Invalid credit limit
Test Case 5:
Input: $C=1000$, $T=1000$
Expected Output: Transaction Approved
Test Case 6:
Input: $C=1000$, $T=1001$
Expected Output: Transaction Declined
Test Case 7:
Input: $C=500$, $T=500$
Expected Output: Transaction Approved

Test Case 8:
Input: C =500, T =501
Expected Output: Transaction Declined
Test Case 9:
Input: C=10000, T=5000
Expected Output: Transaction Approved
Test Case 10:
Input: C=10000, T=15000
Expected Output: Transaction Declined

16. You are developing a program to calculate the total cost of items purchased by a customer based on their unit prices and quantities. Depending on whether the total cost (product of unit price and quantity) is even or odd, apply the corresponding discount for the product and display the final amount after applying the discount.

Let's denote the following variables:

P → Price of the product
Q → Quantity of the product
T → Total cost of the product before applying discount
D → Discounted amount
A → Total cost of the product after applying discount

If T mod 2 == 0, then provide 10 % discount from total cost
If T mod 2 ≠ 0, then provide 15 % discount from total cost

Constraints:

$P > 0$
$Q >= 1$

Test Case 1:
Input: P=10, Q=1
Expected Output: A=9.00
Test Case 2:
Input: P=15, Q=1
Expected Output: A=12.75
Test Case 3:
Input: P=20, Q=3
Expected Output: A=54.00
Test Case 4:
Input: P=25, Q=2
Expected Output: A=45.00
Test Case 5:
Input: P = -30, Q = -3
Expected Output: Invalid price and quantity
Test Case 6:
Input: P=9, Q = -1
Expected Output: Invalid quantity

Test Case 7:
Input: P = -17, Q=2
Expected Output: Invalid price
Test Case 8:
Input: P=0, Q=0
Expected Output: Invalid price and quantity
Test Case 9:
Input: P=0, Q=2
Expected Output: Invalid price
Test Case 10:
Input: P=5, Q=0
Expected Output: Invalid quantity