LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING
(AUTONOMOUS)
Accredited by NAAC with 'A' Grade,
ISO 21001:2018, 50001:2018, 14001:2015 Certified Institution
Approved by AICTE, New Delhi and Affiliated to JNTUK, Kakinada.
L.B.REDDY NAGAR, MYLAVARAM. NTR District, AP, India. 521230.
hodads@lbrce.ac.in , ads@lbrce.ac.in , Phone: 08659-222933, Fax: 08659-222931
DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

Scenario Based Problems on Operators in Java

Problem 1: Smart Energy Consumption Verification

A city has recently deployed smart electricity meters to monitor household power consumption. Every meter records two values:

- The reading recorded during the previous billing cycle.
- The reading recorded during the current billing cycle.

Due to communication errors or faulty meters, the current reading may occasionally be smaller than the previous reading. Such readings are considered invalid and should not be processed further.

If the readings are valid, the electricity board needs to determine the number of units consumed during the billing period.

As a software developer, your task is to help the electricity board identify whether the received readings are valid and, if they are, determine the units consumed.

Input Format

A single line containing two space-separated integers:

previousReading currentReading

Output Format

If the current reading is smaller than the previous reading, print

Invalid Reading

Otherwise, print

Units Consumed = X

where **X** is the number of units consumed.

Constraints

$0 \leq previousReading \leq 1,000,000$

$0 \leq currentReading \leq 1,000,000$

Notes

- Meter readings are always non-negative.
- The program must first verify the validity of the readings.
- A current reading equal to the previous reading is valid and indicates zero consumption.

Sample Test Case 1

Input

1200 1450

Output

Units Consumed = 250

Explanation

Since the current reading is greater than the previous reading, the readings are valid. The consumption is calculated from the difference between the two readings.

Sample Test Case 2

Input

5400 5100

Output

Invalid Reading

Explanation

The current reading is smaller than the previous reading, which indicates an invalid meter record.

Hidden Test Cases

Input	Expected Output
0 0	Units Consumed = 0

Input	Expected Output
0 1	Units Consumed = 1
1 0	Invalid Reading
999999 1000000	Units Consumed = 1
1000000 1000000	Units Consumed = 0
1000000 999999	Invalid Reading
345678 876543	Units Consumed = 530865
543210 543209	Invalid Reading

Problem 2: Warehouse Container Optimization

A logistics company receives thousands of products every day that must be packed into identical shipping containers before dispatch. Each container can accommodate only a fixed number of products.

The warehouse management system must determine how many containers can be completely filled using the available products. If some products remain after filling all possible containers, they are temporarily stored for the next shipment.

As a software developer, your task is to help the warehouse determine the number of completely filled containers and the number of products left unpacked.

Input Format

A single line containing two space-separated integers:

totalProducts containerCapacity

where:

- totalProducts represents the total number of products available.
- containerCapacity represents the maximum number of products that can be packed into one container.

Output Format

Print two lines in the following format:

Filled Containers = X

Remaining Products = Y

where:

- **X** is the number of completely filled containers.
- **Y** is the number of products left unpacked.

Constraints

$1 \leq$ totalProducts $\leq 1{,}000{,}000$

$1 \leq$ containerCapacity $\leq 100{,}000$

containerCapacity $\leq$ totalProducts

Notes

- Every container must be completely filled.
- Partially filled containers are not allowed.
- Remaining products, if any, are stored for the next shipment.
- The program should work efficiently for the largest valid inputs.

Sample Test Case 1

Input

58 7

Output

Filled Containers = 8

Remaining Products = 2

Explanation

Eight containers can each hold seven products, leaving two products unpacked.

Sample Test Case 2

Input

80 10

Output

Filled Containers = 8

Remaining Products = 0

Explanation

All products can be packed into completely filled containers with no products remaining.

Hidden Test Cases

Input	Expected Output
1 1	Filled Containers = 1 Remaining Products = 0
2 1	Filled Containers = 2 Remaining Products = 0

Input	Expected Output
999999 100000	Filled Containers = 9 Remaining Products = 99999
1000000 100000	Filled Containers = 10 Remaining Products = 0
1000000 99999	Filled Containers = 10 Remaining Products = 10
999999 999999	Filled Containers = 1 Remaining Products = 0
100001 100000	Filled Containers = 1 Remaining Products = 1
765432 12345	Filled Containers = 61 Remaining Products = 12387

Problem 3: Elite Sports Camp Selection

A national sports academy is conducting selections for its Elite Training Camp. To maintain high standards, a participant must satisfy **both** performance requirements established by the selection committee.

Each participant is evaluated based on:

- **Fitness Score** (out of 100)
- **Skill Assessment Score** (out of 100)

Only candidates meeting the academy's minimum requirement in **both evaluations** are shortlisted for the camp. If either requirement is not satisfied, the candidate is not selected.

As the software developer for the academy, your task is to automate the shortlisting process.

Input Format

A single line containing two space-separated integers:

fitnessScore skillScore

where:

- fitnessScore is the participant's fitness evaluation score.
- skillScore is the participant's technical skill assessment score.

Output Format

Print exactly one of the following:

Selected

or

Not Selected

Constraints

$0 \leq \text{fitnessScore} \leq 100$

$0 \leq \text{skillScore} \leq 100$

Selection Criteria

A participant is selected **only if**:

- Fitness Score is **at least 75**
- Skill Assessment Score is **at least 80**

Otherwise, the participant is **Not Selected**.

Sample Test Case 1

Input

82 91

Output

Selected

Explanation

The participant satisfies both minimum requirements and is therefore selected.

Sample Test Case 2

Input

78 75

Output

Not Selected

Explanation

Although the fitness score satisfies the requirement, the skill assessment score does not.

Hidden Test Cases

Input	Expected Output
75 80	Selected
74 80	Not Selected
75 79	Not Selected
100 100	Selected
0 0	Not Selected
100 79	Not Selected
74 100	Not Selected
88 85	Selected

Problem 4: Secure Vault Authentication

A research laboratory stores confidential project data inside a secure digital vault. To protect sensitive information, every authorized researcher is assigned two confidential credentials:

- **Researcher ID**
- **Security PIN**

Whenever a researcher attempts to access the vault, the system receives the registered credentials and the credentials entered during login.

Access is granted **only when the entered credentials perfectly match the registered credentials**. Any mismatch, even in one of the credentials, results in the access request being rejected.

As the software developer for the laboratory, your task is to automate the authentication process.

Input Format

The first line contains two space-separated integers representing the registered credentials.

registeredID registeredPIN

The second line contains two space-separated integers representing the credentials entered by the user.

enteredID enteredPIN

Output Format

Print exactly one of the following:

Access Granted

or

Access Denied

Constraints

$1 \leq \text{registeredID} \leq 999999$

$1000 \leq \text{registeredPIN} \leq 999999$

$1 \leq \text{enteredID} \leq 999999$

$1000 \leq \text{enteredPIN} \leq 999999$

Notes

- Every credential consists only of positive integers.
- Both credentials must match exactly for successful authentication.
- A mismatch in either the ID or the PIN results in denied access.

Sample Test Case 1

Input

245681 784512

245681 784512

Output

Access Granted

Explanation

Both the Researcher ID and Security PIN match the registered credentials. Hence, access is granted.

Sample Test Case 2

Input

245681 784512

245681 784500

Output

Access Denied

Explanation

Although the Researcher ID matches, the Security PIN is different. Therefore, access is denied.

Hidden Test Cases

Input	Expected Output
1 1000 1 1000	Access Granted
1 1000 2 1000	Access Denied
1 1000 1 1001	Access Denied
999999 999999 999999 999999	Access Granted
999999 999999 999998 999999	Access Denied
999999 999999 999999 999998	Access Denied
456789 123456 456789 123456	Access Granted
456789 123456 123456 456789	Access Denied

Problem 5: Digital Wallet Recharge

A digital payment company allows users to recharge their wallets. Whenever a recharge is successful, the recharge amount is immediately added to the user's existing wallet balance, and the updated balance is displayed.

The system does not maintain multiple transactions separately. Instead, it updates the existing wallet balance after every successful recharge.

As a software developer, your task is to update the customer's wallet balance after the recharge.

Input Format

A single line containing two space-separated integers.

currentBalance rechargeAmount

where:

- currentBalance is the amount currently available in the wallet.
- rechargeAmount is the amount added by the customer.

Output Format

Print the updated wallet balance in the following format.

Updated Balance = X

where X is the new wallet balance.

Constraints

$0 \leq$ currentBalance $\leq 1{,}000{,}000$

$1 \leq$ rechargeAmount $\leq 100{,}000$

Sample Test Case 1

Input

1500 500

Output

Updated Balance = 2000

Explanation

The recharge amount is added to the existing wallet balance.

Sample Test Case 2

Input

0 250

Output

Updated Balance = 250

Explanation

Since the wallet was empty, the updated balance equals the recharge amount.

Hidden Test Cases

Input	Expected Output
0 1	Updated Balance = 1
1000000 1	Updated Balance = 1000001
999999 1	Updated Balance = 1000000
500000 50000	Updated Balance = 550000
250000 100000	Updated Balance = 350000
1 99999	Updated Balance = 100000
100 100	Updated Balance = 200
765432 23456	Updated Balance = 788888

Problem 6: Premium Membership Verification

An online streaming platform offers **Premium Membership** to customers who maintain a minimum account balance. During the monthly verification process, the system checks the customer's wallet balance and displays the membership category.

Customers maintaining a balance of **₹5000 or more** continue as **Premium Members**. All other customers are categorized as **Regular Members**.

As a software developer, your task is to determine the customer's membership category.

Note: The verification system displays only one membership category for each customer.

Input Format

A single integer representing the customer's wallet balance.

walletBalance

Output Format

Print exactly one of the following:

Premium Member

or

Regular Member

Constraints

$0 \leq$ walletBalance $\leq 1,000,000$

Sample Test Case 1

Input

7200

Output

Premium Member

Explanation

The wallet balance satisfies the minimum requirement for Premium Membership.

Sample Test Case 2

Input

3500

Output

Regular Member

Explanation

The wallet balance is below the required minimum.

Hidden Test Cases

Input	Expected Output
0	Regular Member
4999	Regular Member
5000	Premium Member
5001	Premium Member
1000000	Premium Member
999999	Premium Member
2500	Regular Member
75000	Premium Member

Problem 7: Visitor Counter Update

A science museum uses an automated visitor counting system at its entrance. Every time a visitor enters, the system immediately updates the total visitor count before displaying it on the information screen.

The system stores only the latest count and updates it whenever a new visitor is detected.

As the software developer, your task is to update the visitor count after a new visitor enters the museum.

Note: The displayed value should represent the updated visitor count after processing the new entry.

Input Format

A single integer representing the current visitor count.

currentVisitorCount

Output Format

Print the updated visitor count in the following format:

Updated Visitor Count = X

where **X** is the visitor count after processing the new visitor.

Constraints

$0 \leq$ currentVisitorCount $\leq 1{,}000{,}000$

Sample Test Case 1

Input

256

Output

Updated Visitor Count = 257

Explanation

A new visitor enters the museum, so the visitor count is updated before being displayed.

Sample Test Case 2

Input

0

Output

Updated Visitor Count = 1

Explanation

Since this is the first visitor, the count becomes 1.

Hidden Test Cases

Input	Expected Output
0	Updated Visitor Count = 1
1	Updated Visitor Count = 2
999999	Updated Visitor Count = 1000000
1000000	Updated Visitor Count = 1000001
500000	Updated Visitor Count = 500001
123456	Updated Visitor Count = 123457
999998	Updated Visitor Count = 999999
765432	Updated Visitor Count = 765433

Problem 8: Smart Factory Equipment Status

A manufacturing plant uses an automated monitoring system to track the operational status of its machines. Each machine reports its condition using a binary status code, where every bit represents the state of a specific hardware component.

For maintenance purposes, engineers need to identify only those components that are active in **both** status reports collected from two independent monitoring units. The monitoring software generates a new status code representing these common active components.

As the software developer, your task is to generate the resulting status code.

Note: The problem statement intentionally does not describe how the common active components are determined.

Input Format

A single line containing two space-separated integers.

statusCode1 statusCode2

where:

- statusCode1 represents the first monitoring unit's status.
- statusCode2 represents the second monitoring unit's status.

Output Format

Print the resulting status code in the following format:

Common Status = X

where **X** is the generated status code.

Constraints

$0 \leq$ statusCode1 ≤ 1023

$0 \leq$ statusCode2 ≤ 1023

Both status codes are 10-bit unsigned integers.

Sample Test Case 1

Input

12 10

Output

Common Status = 8

Explanation

Only the hardware components that are active in both monitoring reports are retained in the generated status.

Sample Test Case 2

Input

15 7

Output

Common Status = 7

Explanation

The generated status contains only the common active components present in both reports.

Hidden Test Cases

Input	Expected Output
0 0	Common Status = 0
1023 1023	Common Status = 1023
1023 0	Common Status = 0
0 1023	Common Status = 0
511 255	Common Status = 255
512 511	Common Status = 0
341 682	Common Status = 0
1000 1000	Common Status = 1000

Problem 9: Smart Home Lighting Controller

A smart home uses two independent control systems to operate the living room lights:

- A **Wall Switch**
- A **Mobile Application**

The lighting system is designed so that the lights turn **ON** whenever **at least one** of these control systems sends an activation signal. If neither control system requests activation, the lights remain **OFF**.

Each control system sends its status as:

- **1** → Activation Requested
- **0** → No Activation

As the software developer, your task is to determine whether the lights should be turned ON or remain OFF.

Note: The problem statement intentionally does not describe how the activation signals should be combined.

Input Format

A single line containing two space-separated integers.

wallSwitch mobileApp

where:

- wallSwitch represents the status of the wall switch.
- mobileApp represents the status of the mobile application.

Output Format

Print exactly one of the following:

Lights ON

or

Lights OFF

Constraints

wallSwitch ∈ {0,1}

mobileApp ∈ {0,1}

Sample Test Case 1

Input

1 0

Output

Lights ON

Explanation

The wall switch requests activation, so the lights are turned ON.

Sample Test Case 2

Input

0 0

Output

Lights OFF

Explanation

Neither control system requests activation.

Hidden Test Cases

Input	Expected Output
0 0	Lights OFF
0 1	Lights ON
1 0	Lights ON
1 1	Lights ON
0 0	Lights OFF
1 1	Lights ON
1 0	Lights ON
0 1	Lights ON

Problem 10: Secure Message Integrity Verification

A defense communication network transmits encrypted messages between command centers. To verify that a message has not been altered during transmission, the security system compares two encrypted message signatures:

- Original Signature generated before transmission.
- Received Signature generated after reception.

The verification system processes both signatures and generates a Difference Code, which is used by security analysts during integrity verification. The internal mechanism used to generate this code is part of the organization's secure communication protocol.

As the software developer, your task is to implement the module that generates the Difference Code.

Note: The communication protocol intentionally does not disclose how the Difference Code is computed. Participants are expected to analyze the problem and determine the required computation.

Input Format

A single line containing two space-separated integers.

originalSignature receivedSignature

where:

- originalSignature represents the signature generated before transmission.
- receivedSignature represents the signature generated after reception.

Output Format

Print the generated Difference Code in the following format:

Difference Code = X

where X is the generated Difference Code.

Constraints

$0 \leq$ originalSignature ≤ 1023

$0 \leq$ receivedSignature ≤ 1023

Both signatures are represented as 10-bit unsigned integers.

Sample Test Case 1

Input

12 10

Output

Difference Code = 6

Explanation

The communication system processes both signatures according to its predefined verification protocol and produces the corresponding Difference Code. For the given input, the generated verification value is 6.

Sample Test Case 2

Input

15 15

Output

Difference Code = 0

Explanation

After processing the two signatures using the protocol, the generated verification value is 0.

Hidden Test Cases

Input	Expected Output
0 0	Difference Code = 0
1023 1023	Difference Code = 0
1023 0	Difference Code = 1023
0 1023	Difference Code = 1023
512 511	Difference Code = 1023
256 128	Difference Code = 384
341 682	Difference Code = 1023
1000 1000	Difference Code = 0