Q1. Movie Ticket Pricing

A cinema charges ticket prices based on age:
• if < 13 → 80 units
• if 13–59 → 150 units
• else ≥ 60 → 100 units

Input Format:
Single integer: age

Output Format:
Single integer: ticket_price

Q2. Student Pass/Fail Evaluation

A student passes if marks ≥ 40; otherwise fails.

Input Format:
Single integer: marks

Output Format:
"Pass" or "Fail"

Q3. Restaurant Meal Suggestion by Temperature

A restaurant recommends food based on temperature:
• < 10°C → "Hot Soup"
• 10–25°C → "Warm Meal"
• > 25°C → "Cold Drink"

Input Format:
Single integer: temperature

Output Format:
String: recommended_item

Q4. Library Late Fee Calculation

Late fee per day:
• 1–5 days → 5 units/day
• 6–10 days → 10 units/day
• >10 → 20 units/day

Input Format:
Single integer: days_late

Output Format:
Single integer: fee

Q5. Parking Fee Based on Vehicle Type

Parking charges:
• Two-wheeler → 20 units
• Four-wheeler → 50 units
• Other → 100 units

Input Format:
String: vehicle_type

Output Format:
Single integer: parking_fee

Q6. Water Purifier TDS Status

TDS Level:
• < 300 → “Safe”
• 300–600 → “Moderate”
• > 600 → “Unsafe”

Input Format:
Single integer: tds

Output Format:
String: water_status

Q7. Bank Withdrawal Check

Amount can be withdrawn only if balance ≥ amount.

Input Format:
Two integers: balance, amount

Output Format:
“Approved” or “Insufficient Funds”

Q8. Canteen Eligibility With ID

Service is allowed only if student has ID.

Input Format:
String: has_id (“yes/no”)

Output Format:
“Allowed” or “Denied”

Q9. Number Check

Identify if number is positive, negative, or zero.

Input Format:
Single integer: n

Output Format:
“Positive” / “Negative” / “Zero”

Q10. Weekend Discount

• Weekend → 15% discount of original price(user own input only two values -> Weekday/ weekend)
• Weekday → 5% discount

Input Format:
String: day (“weekday”/“weekend”)

Output Format:
Integer: discount_percentage

Q11. Pass to Exam Based on Attendance

If Minimum 75% attendance required -> Allowed.

Else -> Not Allowed

Input Format:
Integer: attendance_percentage

Output Format:
“Allowed” or “Not Allowed”

Q12. Store Delivery Charge

Free delivery if amount ≥ 500; else add 50.

Input Format:
Integer: amount

Output Format:
Integer: total_amount (free delivery or not)

Q13. Battery Level Mode

• <20% → “Battery Saver ON”
• Else → “Normal Mode”

Input Format:
Integer: battery

Output Format:
String: mode

Q14. Clothing Suggestion by Season

• if Summer → “Cotton Wear”
• if Winter → “Woolen Wear”
• if Rainy → “Raincoat”

Input Format:
String: season

Output Format:
String: clothing

Q15. Traffic Speed Message

• if message <40 → “Slow”
• if message 40–80 → “Normal”
• else >80 → “Overspeeding”

Input Format:
Integer: speed

Output Format:
String: speed_status

Q16. Free Bag Eligibility

Free shopping bag only if bill > 1000.

Input Format:
Integer: amount

Output Format:
“Free Bag” or “Charge 10”

Q17. Recharge Extra Data

• >300 → “1GB extra”
• Else → “No extra data”

Input Format:
Integer: plan_amount

Output Format:
String: data_offer

Q18. Age-Based Streaming Category

• <13 → “Kids”
• Else → “General”

Input Format:
Integer: age

Output Format:
String: category

Q19. Scholarship (Easy Criteria)

Marks ≥ 90 → “Scholarship Eligible”

Input Format:
Integer: marks

Output Format:
String: eligibility

Q20. Metro Ticket Price

• Child (<10) → 10
• Adult (10–59) → 30
• Senior (60+) → 15

Input Format:
Integer: age

Output Format:
Integer: ticket_price

Q21. Coffee Machine Options

1 → “Espresso”
2 → “Latte”
3 → “Cappuccino”
Other → “Invalid Choice”

Input Format:
Integer: choice

Output Format:
String: drink

Q22. Water Level in Tank

• <30% → “Fill Water”
• 30–70% → “Normal”
• >70% → “Overflow Risk”

Input Format:
Integer: level

Output Format:
String: status

Q23. Exam Hall Entry

Student must show hall-ticket (“yes/no”).

Input Format:
String: hall_ticket

Output Format:
“Entry Allowed” or “Denied”

Q24. Job Application Age Check

Eligible only if age ≥ 18.

Input Format:
Integer: age

Output Format:
“Eligible” / “Not Eligible”

Q25. Locker Access

A locker system grants access based on password verification.

Access Rules:

- If the **entered password matches** the **actual password**, access is **granted**.
- If the passwords **do not match**, access is **denied**.

Input Format:

- String: input_password
- String: actual_password

Output Format:

- String:
    - "Unlocked" or "Wrong Password"

Constraints:

- Password length ≤ 50 characters

Q26. Electricity Usage Category

• <100 units → “Low Usage”
• 100–300 → “Moderate”
• >300 → “High Usage”

Input Format:
Integer: units

Output Format:
String: usage_category

Q27. Fruit Ripeness Check

• <50 → “Unripe”
• 50–80 → “Ready”
• >80 → “Overripe”

Input Format:
Integer: ripeness_score

Output Format:
String: result

Q28. Restaurant Dining Type Charge

A restaurant calculates the **final bill amount** based on the dining type selected by the customer.

Charging Rules:

1. If the dining type is **"Dine-in"**, add a **5% service charge** to the bill amount.
2. If the dining type is **"Takeaway"**, no service charge is applied.

Input Format:

- Integer: amount
- String: type ("Dine-in" or "Takeaway")

Output Format:

- Integer: final_amount

Constraints:

- 0 ≤ amount ≤ 1,000,000

- type ∈ {"Dine-in", "Takeaway"}

Q29. Simple Coupon Check

A system checks whether a coupon can be applied based on its validity status.

Rules:

- If the coupon status is **"valid"**, the coupon can be applied.
- If the coupon status is **"expired"**, the coupon cannot be applied.

Input Format:

- String: coupon_status ("valid" or "expired")

Output Format:

- String: result
    - "Coupon Applied" or "Coupon Expired"

Constraints:

- coupon_status ∈ {"valid", "expired"}

Q30. Online Exam Time Check

If time > 3 hours → “Auto-submit”.

Input Format:
Integer: time_hours

Output Format:
String: status

Q31. School Bus Fee

• Distance <5km → 100
• 5–15km → 200
• >15km → 300

Input Format:
Integer: distance

Output Format:
Integer: fee

Q32. Temperature Alert System

• >40 → “Heat Alert”
• <5 → “Cold Alert”
• Else → “Normal”

Input Format:
Integer: temp

Output Format:
String: alert

Q33. Simple Return Window

Return allowed only if days ≤ 7.

Input Format:
Integer: days

Output Format:
“Return Accepted” / “Rejected”

Q34. Product Availability

If quantity < 1 → “Out of Stock”.

Else -> In Stock

Input Format:
Integer: quantity

Output Format:
String: status

Q35. Simple GST Calculation

A system calculates the **GST tax amount** based on the product category and purchase amount.

GST Rules:

- **Food** items are taxed at **5%**.
- **Electronics** items are taxed at **18%**.

Input Format:

- String: category ("Food" or "Electronics")
- Integer: amount

Output Format:

- Integer: tax_amount

Constraints:

- 0 ≤ amount ≤ 1,000,000
- category ∈ {"Food", "Electronics"}

Q36. Employee Shift Validation

Shift must be “day” or “night”.

Input Format:
String: shift

Output Format:
“Valid” / “Invalid”

Q37. Marks Grading

• ≥90 A
• ≥75 B
• ≥60 C
• ≥40 D
• Else F

Input Format:
Integer: marks

Output Format:
String: grade

Q38. Door Lock with Keycard

• Keycard = “active” → “Door Open”
• Else → “Access Denied”

Input Format:
String: keycard_status

Output Format:
String: access

Q39. Daily Steps Target

Target met if steps ≥ 10,000.

Input Format:
Integer: steps

Output Format:
“Target Achieved” / “Keep Going”

Q40. AC Mode Selection

• Temp < 18 → “Heating”
• Temp 18–25 → “Normal”
• Temp > 25 → “Cooling”

Input Format:
Integer: temp

Output Format:
String: ac_mode

Q41. Credit Card Type Assignment

A bank assigns credit card type based on income and credit score:
• If income < 30,000 → “Silver”
• 30,000–70,000 → “Gold”
• > 70,000 → “Platinum”
BUT if credit score < 600 → “Basic Only” regardless of income.

Input Format:
Two integers: income, credit_score

Output Format:
String: card_type

Q42. Company Shift Allocation by Age

A company assigns work shifts:
• Age < 18 → “Not Allowed to Work”
• 18–40 → “Day or Night Shift”
• >40 → “Day Shift Only”

Input Format:
Integer: age

Output Format:
String: assigned_shift

Q43. Insurance Premium Category

Health insurance premium changes based on smoker status and age:
• Non-smoker & age < 40 → “Low Premium”
• Non-smoker & age ≥ 40 → “Medium Premium”

• Smoker & age < 40 → “High Premium”
• Smoker & age ≥ 40 → “Very High Premium”

Input Format:
Integer: age
String: smoker (“yes/no”)

Output Format:
String: premium_category

Q44. Hotel Room Pricing by Season and Membership

A hotel calculates the **final room price** based on the season and customer membership status.

Pricing Rules:

1. If the season is **"peak"**, increase the base price by **30%**.
2. If the season is **"normal"**, the base price remains unchanged.
3. If the customer is a **member**, apply an additional **10% discount** on the price after the seasonal adjustment.
4. The final price should be displayed as an **integer** (rounded to the nearest whole number).

Input Format:

- Integer: base_price
- String: season ("peak" or "normal")
- String: membership ("yes" or "no")

Output Format:

- Integer: final_price

Constraints:

- 1,000 ≤ base_price ≤ 100,000
- season ∈ {"peak", "normal"}
- membership ∈ {"yes", "no"}

Q45. Delivery Charge with Peak Hours

• Distance < 3 km → delivery free
• 3–10 km → 20
• >10 km → 40
• If peak hours (“yes”), add extra 10 units.

Input Format:
Integer: distance
String: peak (“yes/no”)

Output Format:
Integer: delivery_charge

Q46. Voting Eligibility with Department Check

A student can vote only if:
• Age ≥ 18
• Department = “CSE” or “ECE”

Input Format:
Integer: age
String: department

Output Format:
“Eligible” / “Not Eligible”

Q47. Festival Discount Conditions

• Bill > 1000 AND membership = yes → 20% off
• Bill > 1000 AND membership = no → 10% off
• Bill ≤ 1000 → no discount

Input Format:
Integer: amount
String: membership (“yes/no”)

Output Format:
Integer: discount_percentage

Q48. Travel Reimbursement Rules

• Mode = bus/train AND distance > 20 km → “Reimbursed”
• Else → “Not Reimbursed”

Input Format:
String: mode
Integer: distance

Output Format:
String: status

Q49. Phone Unlock with Lockout Check

A smartphone determines whether it can be unlocked based on PIN verification and lockout status.

Unlock Rules:

1. If the phone is in **lockout mode** (lockout = "yes"), display **"Locked Out"**, regardless of the PIN entered.
2. If the phone is **not in lockout mode** and the **entered PIN matches** the correct PIN, display **"Unlocked"**.
3. If the phone is **not in lockout mode** and the **entered PIN does not match**, display **"Wrong PIN"**.

Input Format:

- String: entered_pin
- String: correct_pin
- String: lockout ("yes" or "no")

Output Format:

- String: Result
    - "Unlocked" / "Locked Out" / "Wrong PIN"

Constraints:

- PIN length ≤ 10 characters
- lockout ∈ {"yes", "no"}

Q50. Student Class Promotion

Student is promoted only if:
• All 3 subject marks ≥ 35
• Total marks ≥ 120

Input Format:
Three integers: m1, m2, m3

Output Format:
“Promoted” / “Not Promoted”

Medium

Q56. Hotel Rating System

A hotel is classified into a category based on its price range, user rating, and location.

Classification Rules:

1. If the **price is less than 2000**, the **user rating is 4 or above**, and the **location is prime**, classify the hotel as **"Budget Plus"**.
2. If the **price is 2000 or above**, the **user rating is 4 or above**, and the **location is prime**, classify the hotel as **"Premium"**.
3. If the **user rating is between 3 and 3.9**, classify the hotel as **"Standard"**.
4. In all other cases, classify the hotel as **"Economy"**.

Input Format:

- Integer: price
- Integer: rating
- String: location ("prime" or "normal")

Output Format:

- String: hotel_category

Constraints:

- 0 ≤ price ≤ 100,000
- 1 ≤ rating ≤ 5
- location ∈ {"prime", "normal"}

Q52. Storage Category based on Temperature & Humidity

• Temp < 10 and humidity < 30 → “Cold-Dry Storage”
• Temp < 10 and humidity ≥ 30 → “Cold-Humid Storage”
• Temp ≥ 10 and humidity < 30 → “Dry Storage”
• Else → “General Storage”

Input Format:
Two integers: temp, humidity

Output Format:
String: category

Q53. Food Delivery Offer by Order Type

- Vegetarian & amount > 200 → 10% off
- Non-veg & amount > 300 → 15% off
- No offer otherwise

Input Format:
String: type
Integer: amount

Output Format:
Integer: discount_percentage

Q54. Hiring Eligibility

A candidate is hired only if:
- Degree = “B.Tech”
- Experience ≥ 2
- Score ≥ 60

Input Format:
String: degree
Integer: experience
Integer: test_score

Output Format:
“Hired” / “Rejected”

Q55. Gym Membership Level

- BMI < 18 → “Underweight Plan”
- BMI 18–25 → “Standard Plan”
- BMI > 25 → “Weight Loss Plan”
- If workout_days < 3 → override to “Beginner Plan”

Input Format:
Float: bmi
Integer: workout_days

Output Format:
String: plan

Q56. Hotel Rating System

A hotel is classified into a category based on its price range, user rating, and location.

Classification Rules:

1. If the **price is less than 2000**, the **user rating is 4 or above**, and the **location is prime**, classify the hotel as **"Budget Plus"**.
2. If the **price is 2000 or above**, the **user rating is 4 or above**, and the **location is prime**, classify the hotel as **"Premium"**.
3. If the **user rating is between 3 and 3.9**, classify the hotel as **"Standard"**.
4. In all other cases, classify the hotel as **"Economy"**.

Input Format:

- Integer: price
- Integer: rating
- String: location ("prime" or "normal")

Output Format:

- String: hotel_category

Constraints:

- $0 \leq$ price $\leq 100{,}000$
- $1 \leq$ rating ≤ 5
- location $\in$ {"prime", "normal"}

Q57. Restaurant Combo Offer

A restaurant provides a **combo discount** based on the items ordered by a customer.

Discount Rules:

- If the customer orders **both a starter and a main course**, apply a **15% discount**.
- Otherwise, **no discount** is applied.

Input Format:

- String: starter ("yes" or "no")
- String: main ("yes" or "no")

Output Format:

- String: discount_status

- "Discount Applied" or "No Discount"

Constraints:

- starter ∈ {"yes", "no"}
- main ∈ {"yes", "no"}

Q58. Metro Ticket Fare Computation

A metro system calculates the **ticket fare** based on passenger age, travel distance, and peak-hour timing.

Fare Rules:

1. **Base fare** is calculated as:
   fare = distance × 2
2. **Age-based discount**:
   - If age is **below 12**, apply a **50% discount**. (fare *0.5)
   - If age is **60 or above**, apply a **30% discount**. (fare *0.7)
3. **Peak-hour surcharge**:
   - If peak hour is **yes**, add **20%** to the fare.
4. The final fare should be rounded to the **nearest integer**.

Input Format:

- Integer: age
- Integer: distance
- String: peak ("yes" or "no")

Output Format:

- Integer: fare

Constraints:

- 0 ≤ age ≤ 120
- 1 ≤ distance ≤ 100
- peak ∈ {"yes", "no"}

Q59. Content Filtering in Streaming App

• Age < 13 → “Kids Only”
• Age 13–17 → If subscription = premium → “Teen + Premium” else “Teen Standard”
• Age ≥ 18 → “All Content”

Input Format:
Integer: age
String: subscription

Output Format:
String: category

Q60. EMI Eligibility

EMI allowed only if:
• Amount > 5000
• Mode = “credit”

Input Format:
Integer: amount
String: mode

Output Format:
“EMI Available” / “Not Available”

Q61. Order Return Priority

An e-commerce system assigns a **return processing priority** based on customer type and the number of days since purchase.

Priority Rules:

1. If the customer type is **VIP**, assign **"High Priority"**.
2. If the customer is **not VIP** and the number of days since purchase is **7 or fewer**, assign **"Medium Priority"**.
3. In all other cases, assign **"Low Priority"**.

Input Format:

- String: customer_type ("VIP" or "Regular")
- Integer: days

Output Format:

- String: priority

Constraints:

- customer_type ∈ {"VIP", "Regular"}

- 0 ≤ days ≤ 365

Q62. Hostel Room Eligibility

Eligible only if:
• Distance > 50 km
• Marks > 70

Input Format:
Integer: distance
Integer: marks

Output Format:
“Eligible” / “Not Eligible”

Q63. Hospital Emergency Level

A hospital determines the **emergency level** of a patient based on severity score and age.

Emergency Level Rules:

1. Assign the initial emergency level based on **severity score**:
    - Severity score **greater than 80** → **"Critical"**
    - Severity score **between 50 and 80 (inclusive)** → **"High"**
    - Severity score **less than 50** → **"Moderate"**
2. If the patient’s **age is greater than 70**, increase the emergency level by **one step**, unless it is already **Critical**.

Severity level order:
Moderate → High → Critical

Input Format:

- Integer: severity
- Integer: age

Output Format:

- String: level

Constraints:

- 0 ≤ severity ≤ 100
- 0 ≤ age ≤ 120

Q64. Packaging Charge Calculation

Packaging charges vary:
• Fragile = yes AND size = “large” →cost = 50
• Fragile = yes AND size = “small” →cost = 30
• Not fragile →cost = 10

Input Format:
String: fragile
String: size

Output Format:
Integer: charge

Q65. Courier Service Charge

A courier service calculates the **delivery fee** based on parcel weight and delivery distance.

Charging Rules:

1. If the parcel weight is **less than 1 kg**, the base charge is **20**.
2. If the parcel weight is **between 1 and 5 kg (inclusive)**, the base charge is **50**.
3. If the parcel weight is **greater than 5 kg**, the base charge is **100**.
4. If the delivery distance is **greater than 50 km**, add an additional charge of **20**.

Input Format:

- Integer: weight *(in kilograms)*
- Integer: distance *(in kilometers)*

Output Format:

- Integer: fee

Constraints:

- 0 ≤ weight ≤ 100
- 0 ≤ distance ≤ 1000

Q66. ATM Withdrawal Rules

ATM allows withdrawal only if:
• Amount is divisible by 100 and Balance ≥ amount -> Success

If Not Failed

Input Format:
Integer: balance
Integer: amount

Output Format:
“Success” / “Failed”

Q67. Game Badge Assignment

• Win rate ≥ 80% & matches ≥ 50 → “Pro Badge”
• Win rate ≥ 50% → “Intermediate Badge”
• Else → “Beginner Badge”

Input Format:
Integer: win_rate
Integer: matches

Output Format:
String: badge

Q68. Parking Floor Assignment

A parking management system assigns a parking floor based on the type of vehicle and the time of day.

Assignment Rules:

1. If the time is **night**, **all vehicles** are assigned to the **"Ground Floor"**.
2. If the time is **day**, parking is assigned as follows:
    - **EV cars → "Ground Floor"**
    - **SUVs → "First Floor"**
    - **Bikes → "Second Floor"**

Input Format:

- String: vehicle ("EV", "SUV", "Bike")
- String: time ("day" or "night")

Output Format:

- String: floor

Constraints:

- vehicle ∈ {"EV", "SUV", "Bike"}
- time ∈ {"day", "night"}

Q69. Fee Concession Rules

Concession allowed only if:
• Marks > 85
• Income < 30000

Input Format:
Integer: marks
Integer: income

Output Format:
“Concession Granted” / “Rejected”

Q70. Cinema Seat Category

A cinema assigns seat categories to customers based on ticket type and age.

Seat Assignment Rules:

1. If the ticket type is **VIP**, assign **"Premium Seat"**.
2. If the ticket type is **Regular** and the customer’s age is **60 or above**, assign **"Priority Seat"**.
3. In all other cases, assign **"Standard Seat"**.

Input Format:

- String: ticket_type ("VIP" or "Regular")
- Integer: age

Output Format:

- String: seat

Constraints:

- ticket_type ∈ {"VIP", "Regular"}
- 0 ≤ age ≤ 120

Q71. EMI Holiday Request

Approved only if:
• Payment history > 12 months
• Defaults = 0

Input Format:
Integer: months
Integer: defaults

Output Format:
“Approved” / “Denied”

Q72. GST Slab Assignment

A taxation system assigns the **GST percentage** to a product based on its category.

GST Rules:

- **Essential** products are taxed at **5%**.(Just Print)
- **Standard** products are taxed at **12%**.(Just Print)
- **Luxury** products are taxed at **28%**.(Just Print)

Input Format:

- String: category ("Essential", "Standard", "Luxury")

Output Format:

- Integer: gst_percentage

Constraints:

- category ∈ {"Essential", "Standard", "Luxury"}

Q73. Water Billing System

A water utility company calculates the **water bill** based on consumption and customer type.

Billing Rules:

1. If water usage is **less than 100 liters**, the cost is **0.5 per unit**.
2. If water usage is **between 100 and 300 liters (inclusive)**, the cost is **1.0 per unit**.
3. If water usage is **greater than 300 liters**, the cost is **1.5 per unit**.
4. If the customer type is **commercial**, the total bill is multiplied by **1.2**.

Input Format:

- Integer: usage *(water usage in liters)*
- String: type ("residential" or "commercial")

Output Format:

- Float: bill

Constraints:

- 0 ≤ usage ≤ 10,000
- type ∈ {"residential", "commercial"}

Q74. Grade Calculation with Minimum Subject Requirement

A student's **grade** is calculated based on the **average marks** of three subjects.
However, if the student scores **less than 30 in any subject**, the result is **Fail**, regardless of the average.

Grading Rules:

1. If **any subject mark is less than 30**, output **"Fail"**.
2. Otherwise, calculate the **average** of the three marks and assign the grade:
    - Average ≥ 75 → **"A"**
    - Average ≥ 60 and < 75 → **"B"**
    - Average ≥ 50 and < 60 → **"C"**
    - Average < 50 → **"D"**

Input Format:

- Three Integers: m1, m2, m3

Output Format:

- String: **Grade** ("A", "B", "C", "D") or "Fail"

Constraints:

- 0 ≤ m1, m2, m3 ≤ 100

Q75. Cashback Eligibility

Cashback only if:
• Payment = "wallet"
• Amount > 200

Input Format:
String: payment
Integer: amount

Output Format:
"Cashback Applied" / "No Cashback"

Q76. Resume Filtering

Shortlist if:
• Experience ≥ 3
• Portfolio score ≥ 70
• Competition rank in top 100

Input Format:
Integer: exp
Integer: portfolio
Integer: rank

Output Format:
“Shortlisted” / “Rejected”

Q77. Medicine Purchase Rule

A pharmacy system determines whether a user is allowed to purchase a medicine based on prescription requirements.

The purchase decision depends on:

- **Whether a prescription is required**
- **Whether the user has a valid prescription**

Rules:

1. If a **prescription is required** and the user **has a prescription**, the purchase is **"Allowed"**.
2. If a **prescription is required** and the user **does not have a prescription**, the purchase is **"Not Allowed"**.
3. If a **prescription is not required**, the purchase is **"Allowed"**.

Input Format:

- String: required ("yes" or "no")
- String: has_prescription ("yes" or "no")

Output Format:

- String: status
    - "Allowed" or "Not Allowed"

Constraints:

- required ∈ {"yes", "no"}
- has_prescription ∈ {"yes", "no"}

Q78. Transaction Fraud Check

Block if:
• amount > 5000 AND suspicious = yes

Input Format:
Integer: amount
String: suspicious

Output Format:
“Blocked” / “Allowed”

Q79. Cloud Billing Tier

A cloud service provider calculates the **bill amount** based on system usage.

The billing depends on:

- **CPU Usage Percentage**
- **Active Usage Hours**

Billing Rules:

1. The **base cost** is calculated as:
   bill = cpu × hours
2. If the **CPU usage is greater than 80%**, add a **20% surcharge** to the bill.
3. The final bill amount should be displayed as a **floating-point value**.

Input Format:

- Integer: cpu *(CPU usage percentage)*
- Integer: hours *(number of active hours)*

Output Format:

- Float: bill_amount

Constraints:

- 0 ≤ cpu ≤ 100
- 1 ≤ hours ≤ 10,000

Q80. Restaurant Waiting Time

A restaurant estimates the **waiting time** for customers based on daily demand and staff availability.

The waiting time depends on:

- **Day Type** (weekday or weekend)
- **Queue Length** (number of customers waiting)
- **Staff Availability** (number of staff on duty)

Waiting Time Rules:

1. On **weekdays**, the base waiting time is **5 minutes** per customer in the queue.
2. On **weekends**, the base waiting time is **10 minutes** per customer in the queue.
3. Each available staff member reduces the total waiting time by **5 minutes**.
4. The waiting time cannot be **less than 0 minutes**.

Input Format:

- String: day ("weekday" or "weekend")
- Integer: queue
- Integer: staff

Output Format:

- Integer: Estimated waiting_time (in minutes)

Constraints:

- day ∈ {"weekday", "weekend"}
- 0 ≤ queue ≤ 100
- 0 ≤ staff ≤ 50

Hard:

Q81. Employee Appraisal Calculation

A company calculates annual appraisal based on:
• Performance Tier: A, B, C
• Attendance Percentage
• Number of projects completed
• Behaviour rating (1–5)

Rules:
• Tier A: If attendance ≥ 90% AND projects ≥ 5 AND behaviour ≥ 4 → “High Appraisal”
• Tier B: If attendance ≥ 85% AND projects ≥ 3 → “Medium Appraisal”
• Tier C: If attendance ≥ 80% → “Low Appraisal”
Otherwise → “No Appraisal”

Input Format:
String: performance_tier
Integer: attendance
Integer: projects
Integer: behaviour

Output Format:
String: appraisal_result

Q82. Home Loan Advanced Eligibility System

A bank evaluates a customer's eligibility for a **home loan** based on financial stability and risk factors.

The loan approval depends on the following criteria:

- **Monthly Income**
- **Credit Score**
- **Age**
- **Number of Existing Loans**
- **Job Stability**

Eligibility Rules:

A home loan is **approved** only if **all** of the following conditions are satisfied:

1. Monthly income is at least **40,000**.
2. Credit score is **700 or above**.
3. Applicant age is between **21 and 60** (inclusive).
4. Number of existing loans is **2 or fewer**.
5. Job status must be **stable**.

If any of the above conditions are not met, the loan is **rejected**.

Input Format:

- Integer: income
- Integer: credit_score
- Integer: age
- Integer: existing_loans
- String: job_status ("stable" or "unstable")

Output Format:

- String: status
    - "Approved" or "Rejected"

Constraints:

- 0 ≤ income ≤ 1,000,000
- 300 ≤ credit_score ≤ 900
- 18 ≤ age ≤ 70
- 0 ≤ existing_loans ≤ 10
- job_status ∈ {"stable", "unstable"}

Q83. Ride-Sharing Dynamic Pricing

A ride-sharing platform calculates a **fare multiplier** based on trip conditions and rider profile.

The fare multiplier depends on:

- **Trip Distance**
- **Time of Day** (peak or non-peak)
- **Demand Level** (low, medium, high)
- **Rider Rating**

Rules:

1. If the time is **peak** and demand is **high**, set the multiplier to **2.0**.
2. If the time is **peak** and demand is **medium**, set the multiplier to **1.5**.
3. If the time is **non-peak** and demand is **high**, set the multiplier to **1.3**.
4. In all other cases, set the multiplier to **1.0**.
5. If the rider rating is **less than 3.0**, add a **penalty of 0.2** to the multiplier.

Input Format:

- Integer: distance
- String: time
- String: demand
- Float: rating

Output Format:

- Float: fare_multiplier

Constraints:

- 1 ≤ distance ≤ 1000
- time ∈ {"peak", "non-peak"}
- demand ∈ {"low", "medium", "high"}
- 1.0 ≤ rating ≤ 5.0

Q84. University Scholarship Multi-Level Selection

A student qualifies ONLY IF:
• Academic score ≥ 85
• Attendance ≥ 90
• Income ≤ 200000
• If they have sports quota = “yes”, minimum score reduces to 80
• If income > 200000 → automatically rejected

Input Format:
Integer: score
Integer: attendance
Integer: income
String: sports (“yes/no”)

Output Format:
“Selected” / “Rejected”

Q85. Cybersecurity Login Protection

A cybersecurity system evaluates login attempts and determines the appropriate result based on risk indicators and authentication status.

The decision depends on:

- **Location Match** (true or false)
- **Device Trust Status** (true or false)
- **Number of Failed Login Attempts**
- **Risk Score**
- **Entered Password**
- **Correct Password**

Rules (evaluated in order):

1. If the **location does not match** or the **device is not trusted**, the system considers the login attempt as **high risk**.
2. If the **risk score is greater than 70 OR** the **number of failed attempts is greater than 3**, the system returns **"Account Locked"**.
3. If the **entered password matches** the correct password, the system returns **"Login Successful"**.
4. Otherwise, the system returns **"Invalid Credentials"**.

Input Format:

- String: location_match ("true" or "false")
- String: device_trusted ("true" or "false")
- Integer: failed_attempts
- Integer: risk_score
- String: password
- String: correct_password

Output Format:

- String: result

Constraints:

- location_match ∈ {"true", "false"}
- device_trusted ∈ {"true", "false"}
- 0 ≤ failed_attempts ≤ 10
- 0 ≤ risk_score ≤ 100
- Password length ≤ 50 characters

Q86. Warehouse Autonomous Robot Navigation

An autonomous warehouse robot selects a navigation path based on environmental and load conditions to ensure safety and efficiency.

The navigation decision depends on:

- **Aisle Traffic Level** (low, medium, high)
- **Load Weight** (in kilograms)
- **Temperature Zone** (cold, normal, hot)

Rules:

1. If the aisle traffic is **high**, select **"Alternate Path"**.
2. If the aisle traffic is **medium** and the load weight is **greater than 20 kg**, select **"Reinforced Path"**.
3. If the temperature zone is **cold** and the load weight is **less than 10 kg**, select **"Fast Path"**.
4. If the temperature zone is **hot** and the load weight is **greater than 30 kg**, select **"Cooling Required"**.
5. If none of the above conditions are satisfied, select **"Standard Path"**.

Input Format:

- String: traffic
- Integer: weight
- String: temp_zone

Output Format:

- String: Selected navigation path

Constraints:

- traffic ∈ {"low", "medium", "high"}
- 0 ≤ weight ≤ 100
- temp_zone ∈ {"cold", "normal", "hot"}

Q87. Airline Boarding Priority System

An airline assigns boarding priority to passengers based on personal and ticket-related factors.

The boarding priority is determined using the following rules **in order**:

1. If the passenger has a **disability** (yes), assign **Priority 1**.
2. If the passenger's **age is 60 or above**, assign **Priority 2**.
3. If the passenger has **Gold loyalty status**, assign **Priority 3**.
4. If the passenger is traveling in **Business class**, assign **Priority 4**.
5. All other passengers are assigned **Priority 5**.

Input Format:

- String: loyalty ("Gold", "Silver", "None")
- String: ticket_class ("Business" or "Economy")
- String: disability ("yes" or "no")
- Integer: age

Output Format:

- String: priority_number
  (Example: "Priority 1", "Priority 2", etc.)

Constraints:

- loyalty ∈ {"Gold", "Silver", "None"}
- ticket_class ∈ {"Business", "Economy"}
- disability ∈ {"yes", "no"}
- 0 ≤ age ≤ 120

Q88. ICU Ventilator Allocation

A hospital system allocates ventilators to patients based on medical urgency and risk factors.

The priority for ventilator allocation depends on:

- **Severity Score**
- **Age**
- **Comorbidity Level** (low, medium, high)

Rules:

1. If the **severity score is greater than 85**, assign **"Highest Priority"**.
2. If the **severity score is between 60 and 85 (inclusive)** and the **comorbidity level is high**, assign **"High Priority"**.
3. If the **patient's age is greater than 70**, increase the priority to **"High Priority"** (unless already highest).
4. If the **severity score is less than 40**, assign **"Lowest Priority"**.
5. If none of the above conditions apply, assign **"Normal Priority"**.

Input Format:

- Integer: severity
- Integer: age

- String: comorbidity

Output Format:

- String: priority

Constraints:

- 0 ≤ severity ≤ 100
- 0 ≤ age ≤ 120
- comorbidity ∈ {"low", "medium", "high"}

Q89. Corporate Salary Computation

A company calculates an employee's **final salary** based on performance, location, and overtime work.

The final salary depends on:

- **Base Salary**
- **Performance Multiplier**
    - Grade **A** → 1.2
    - Grade **B** → 1.1
    - Grade **C** → 1.0
- **Region Tax**
    - **Urban** → 10%
    - **Rural** → 5%
- **Overtime Hours** (each hour adds a fixed amount of 100)

Salary Calculation Rules:

1. Apply the **performance multiplier** to the base salary.
2. Add **overtime pay** (overtime × 100).
3. Deduct **region tax** from the total salary.
4. The final salary should be displayed as an **integer**.

Input Format:

- Integer: base
- String: performance ("A", "B", "C")

- String: region ("urban" or "rural")
- Integer: overtime

Output Format:

- Integer: final_salary

Constraints:

- 10,000 ≤ base ≤ 1,000,000
- performance ∈ {"A", "B", "C"}
- region ∈ {"urban", "rural"}
- 0 ≤ overtime ≤ 500

Q90. Smart Home Temperature Intelligence

A smart home system automatically selects the temperature control mode based on environmental conditions and user settings.

The mode selection depends on:

- **Outside Weather**
- **User Preference** (hot, cold, normal)
- **Room Occupancy** (yes or no)
- **Energy Saving Mode** (yes or no)

Rules:

1. If **energy saving mode** is enabled, the system selects **"Eco Mode"**.
2. If there is **no room occupancy**, the system selects **"Power Saving Mode"**.
3. If the user preference is **hot** and the outside weather is **cold**, the system selects **"Heating"**.
4. If the user preference is **cold** and the outside weather is **hot**, the system selects **"Cooling"**.
5. If none of the above conditions are satisfied, the system selects **"Normal Mode"**.

Input Format:

- String: outside ("hot" or "cold")
- String: preference ("hot", "cold", "normal")
- String: occupancy ("yes" or "no")

- String: energy_save ("yes" or "no")

Output Format:

- String: Selected temperature control mode

Constraints:

- outside ∈ {"hot", "cold"}
- preference ∈ {"hot", "cold", "normal"}
- occupancy ∈ {"yes", "no"}
- energy_save ∈ {"yes", "no"}

Q91. Stock Trading Bot Decision System

A stock trading bot decides an action based on market conditions and available resources.

The decision depends on the following factors:

- **Price Trend** (up, down, stable)
- **Trading Volume**
- **Risk Level** (low, medium, high)
- **Available Capital**

Rules:

1. If the **price trend is up**, **volume is high**, and **risk level is low**, the bot performs **"Buy"**.
2. If the **price trend is down** and **risk level is low**, the bot performs **"Short Sell"**.
3. If the **risk level is high**, the bot performs **"Hold"**.
4. If the **available capital is less than 1000**, the bot performs **"Insufficient Capital"**.
5. If none of the above conditions are met, the bot performs **"No Action"**.

Input Format:

- String: trend
- Integer: volume
- String: risk
- Integer: capital

Output Format:

- String: action

Constraints:

- trend ∈ {"up", "down", "stable"}
- 0 ≤ volume ≤ 1,000,000
- risk ∈ {"low", "medium", "high"}
- 0 ≤ capital ≤ 10,000,000

Q92. City Traffic Intelligent Signal Timing

A smart traffic control system determines the signal timing based on real-time traffic conditions.

The signal timing depends on:

- **Vehicle Density** (low, medium, high)
- **Time of Day**
- **Emergency Vehicle Detection** (yes or no)

Rules:

1. If an **emergency vehicle** is detected, the signal should turn **"Immediate Green"**.
2. If vehicle density is **high**, set the signal to **"Long Green"**.
3. If vehicle density is **medium**, set the signal to **"Normal Green"**.
4. If vehicle density is **low**, set the signal to **"Short Green"**.

Input Format:

- String: density
- String: time
- String: emergency

Output Format:

- String: signal_type

Constraints:

- density ∈ {"low", "medium", "high"}
- time ∈ {"morning", "afternoon", "evening", "night"}
- emergency ∈ {"yes", "no"}

Q93. Banking Fraud Detection Engine

A banking system needs to determine whether a transaction should be flagged as fraudulent based on multiple risk factors.

A transaction is flagged as **fraudulent** if **two or more** of the following conditions are met:

- Transaction amount is **greater than twice the average transaction amount**
- **Location mismatch** is detected
- Transaction time is **unusual** (for example, between **2 AM and 4 AM**)
- The account is in a **risky state**

Input Format:

- Integer: amount
- Integer: average
- String: location_match ("yes" or "no")
- String: unusual_time ("yes" or "no")
- String: risky_account ("yes" or "no")

Output Format:

- String:
    - "Fraud Detected" if two or more conditions are met
    - "Transaction Safe" otherwise

Constraints:

- $0 \leq$ amount $\leq 1{,}000{,}000$
- $0 \leq$ average $\leq 500{,}000$
- location_match $\in$ {"yes", "no"}
- unusual_time $\in$ {"yes", "no"}
- risky_account $\in$ {"yes", "no"}

Q94. Software License Verification System

A software license is considered **valid** only if **all** the following conditions are satisfied:

- The **Device ID matches**
- The **User ID matches**
- The **License key is not expired**

- The **Subscription status is active**

Input Format:

- String: device
- String: user
- String: expiry *(e.g., "valid" or "expired")*
- String: status *(e.g., "active" or "inactive")*

Output Format:

- String:
    - "Valid License" if all conditions are met
    - "Invalid License" otherwise

Constraints:

- device ∈ {"matched", "not_matched"}
- user ∈ {"matched", "not_matched"}
- expiry ∈ {"valid", "expired"}
- status ∈ {"active", "inactive"}

Q95. Logistics Delivery Route Optimization

The delivery route is determined based on the following factors:

- **Package Fragility**
- **Driver Rating**
- **Distance**

Rules:

1. If the package is **fragile** and the delivery distance is **long**, select **"Cushioned Route"**.
2. If the driver has a **low rating**, select **"Safe Route Only"**.
3. If the delivery distance is **less than 5 km**, select **"Express Route"**.
4. If none of the above conditions are satisfied, select **"Standard Route"**.

Input Format:

- String: fragile ("yes" or "no")
- Integer: rating

- Integer: distance

Output Format:

- String: Selected delivery route

Constraints:

- fragile ∈ {"yes", "no"}
- 1 ≤ rating ≤ 5
- 1 ≤ distance ≤ 1000

Q96. Government Subsidy Allotment

Eligibility depends on:
• Income
• Landholding
• Tax filing history
• Category (general/sc/st/obc)

Rules:
• Income ≤ 150000
• Landholding < 2 acres
• If all true → “Eligible”

Input Format:
Integer: income
Float: land
String: category

Output Format:
String: eligibility

Q97. Fitness Tracker Wellness Score

Score depends on:
• Sleep hours
• Steps
• Heart rate
• Stress score

Rules (example):
• Good sleep & high steps → +30
• High stress OR high heart rate → reduce 20
• Base score starts at 50

Input Format:
Integer: sleep
Integer: steps
Integer: heart_rate
Integer: stress

Output Format:
Integer: wellness_score

Constraints:

- 0 ≤ sleep ≤ 24 *(hours per day)*
- 0 ≤ steps ≤ 100000
- 30 ≤ heart_rate ≤ 200 *(beats per minute)*
- 0 ≤ stress ≤ 100 *(stress score)*

Q98. Drone Autonomous Landing Decision

Drone decides landing spot using:
• Wind speed
• Battery
• GPS signal
• Obstacle detection

Rules:
• Wind > 40 → "Abort Landing"
• Battery < 20 → "Emergency Landing"
• GPS weak → "Hold Position"
• Obstacles → "Move to Safe Zone"

Input Format:
Integer: wind
Integer: battery
String: gps
String: obstacle

Output Format:
String: decision

Q99. Investment Plan Recommendation System

Recommendation depends on:
• Age

• Income
• Risk appetite
• Savings
• Future goals (short/long)

Rules:
• Young + high income + high risk → “Equity Plan” (Based on the constraints given)
• Low risk + long-term → “Debt Plan”
• Low savings → “Basic Savings Plan”

Input Format:
Integer: age
Integer: income
String: risk
Integer: savings
String: goals

Output Format:
String: plan

Constraints:

- 18 ≤ age ≤ 65
- 0 ≤ income ≤ 1,000,000
- risk ∈ {"high", "low"}
- 0 ≤ savings ≤ 500,000
- goals ∈ {"short", "long"}

Q100. Factory Machine Mode Switch

Machine mode depends on:
• Temperature
• Load
• Maintenance status
• Shift type

Rules:
• Temp > 90 OR load > 80 → “Shutdown Mode”
• Maintenance = due → “Service Mode”
• Night shift → “Low Power Mode”
• Else → “Operational Mode”

Input Format:
Integer: temp

Integer: load
String: maintenance
String: shift

Output Format:
String: mode