Conditional Statements - Exercises

1. Marks 1
2. Larger number 2
3. Number from 0 to 9 with words 2
4. Face of figures 2
5. Day of the week 3
6. Even or odd 4
7. Speed 4
8. Alarm after 15 minutes 5
9. Address by age and gender 5
10. Grocery 6
11. Number in the range 7
12. Simple Calculator 7
13. Vegetable Market 8
14. Holiday 9
15. Makeup Shop 10
16. At sea 11
17. Grade Calculator 13
18. Leap Year Checker 13
19. Movie Ticket Price 14
20. Days in a Month 14
21. University Admissions 15
22. Discount Calculator 15
23. Movie Classification 15
24. Airline Luggage Charges 16
25. Adventure Game: Path Decision 16
26. Potion Brewing Decision 17
27. Survival in the Wilderness 18
28. Climate Zone Identifier 19
29. Architectural Era Identifier 19

# Marks

Write a console program that reads a mark (real number) entered by the user and prints "Excellent!" if the mark is **5.50 or higher.**

|  |  |
| --- | --- |
| **Input** | **Output** |
| **6** | Excellent! |
| **5** | no output |
| **5.51** | Excellent! |

# Larger number

Write a function that reads accepts two integers and prints the larger of the two.

|  |  |
| --- | --- |
| **Input** | **Output** |
| **2**  **4** | 4 |
| **7**  **12** | 12 |
| **-1**  **-5** | -1 |

# Number from 0 to 9 with words

Write a function that gets **an integer in the range [0... 9]** and spells it out in words in English. If the number is out of range, it spells out **"too big".**

|  |  |
| --- | --- |
| **Input** | **Output** |
| **7** | seven |
| **5** | five |
| **10** | too big |

Guidelines:

1. Check for the size of the number. Print with text any of the words ["zero", "one", "two", "three"... "nine"] depending on the value.

2. Print "**too big**" if the number is greater than 9:

# Face of figures

Write a function that takes the dimension of a geometric figure and calculates its face.

The figures are of four types: **square, rectangle, circle, and triangle.**

* If the figure is **a square**, the next parameter is a number - the length of its side.
* If the figure is **a rectangle**, the next two parameters - the lengths of its sides.
* If the figure is a circle, the next parameter is a number - the radius of the circle.
* If the figure is a **triangle**, the next two parameters - the length of its side and the length of the height to it.

Round the result to **2 decimal places**.

Sample input:

|  |  |
| --- | --- |
| **Input** | **Output** |
| **square**  **5** | 25.00 |
| **rectangle**  **10**  **3.5** | 35.00 |
| **triangle**  **4.5**  **20** | 45.00 |
| **circle**  **10** | 314.16 |

You can search for the face formulas of the different shapes on the Internet. Use **Math.PI** to use the number pi when calculating the face of a circle.

# Day of the week

Write a function that receives an integer and prints a day of the week (in English), within the limits [1...7] or prints "Error" in case the entered number is invalid.

Sample input:

|  |  |
| --- | --- |
| **Input** | **Output** |
| **1** | Monday |
| **2** | Tuesday |
| **3** | Wednesday |
| **4** | Thursday |
| **5** | Friday |
| **6** | Saturday |
| **7** | Sunday |
| **8** | Error |

# Even or odd

Write a function that receives an integer and prints on the console, whether even or odd.

Sample input:

|  |  |
| --- | --- |
| **Input** | **Output** |
| **2** | even |
| **3** | odd |
| **25** | odd |
| **1024** | even |

Guidelines:

1. Check what is % (operator modulo) <https://www.google.com/search?q=modulo>

2. Check whether the number is even by dividing it by 2 and check whether there is a remainder of the division. Print the output by condition – the text "even":

if (num % 2 == 0) {

sc.println("even")

}

3. Otherwise, print "odd"

# Speed

Write a program that receives speed (real number) and prints speed information.

* At **speeds up to 10 (inclusive)** print "**slow**".
* At speeds above **10** and up to **60** print "**average**".
* At speeds above **60** and up to **120**, print "**fast**".
* At speeds above **120** and up to **160**, print "**super-fast**".
* At higher speeds, print "**turbo-fast**".

Sample input:

|  |  |
| --- | --- |
| **Input** | **Output** |
| **10** | slow |
| **59** | average |
| **120** | fast |
| **121** | super-fast |
| **183** | turbo-fast |

# Alarm after 15 minutes

Write a function that takes an hour and minutes of a 24-hour day and calculates what time it will be after 15 minutes. Print the result in hours: minutes format.

Hours are always between 0 and 23, and minutes are always between 0 and 59. Hours are written in one or two digits.

Minutes are always written with two digits, with a leading zero when necessary.

Try to find out how to add 0s in front of a number here: <https://www.google.com/search?q=java+print+number+with+leading+zeros>

Sample input:

|  |  |
| --- | --- |
| **Input** | **Output** |
| **1**  **47** | 2:02 |
| **0**  **2** | 0:17 |
| **23**  **59** | 0:14 |
| **11**  **07** | 11:22 |
| **12**  **48** | 13:03 |

# Address by age and gender

Write a console program that assumes age (real number) and gender ('m' or 'f'), and prints a reference among the following:

* "Mr." — male (sex 'm') 16 years of age or older
* "Master" – boy (gender 'm') under 16 years
* "Ms." — woman (sex 'f') 16 years of age or older
* "Miss" – girl (gender 'f') under 16

Sample input:

|  |  |
| --- | --- |
| **Input** | **Output** |
| **14**  **f** | Miss |
| **17**  **m** | Mr. |
| **10**  **m** | Master |
| **32**  **f** | Ms. |

if (gender == "f") {

if (age <= 16) {

sc.println("Miss");

} else {

sc.println("Ms.");

}

} else {

if (age <= 16) {

TODO

} else {

TODO

}

}

# Grocery

A chain of stores opens groceries in several cities and sells at different prices according to the city:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **City / Product** | coffee | **water** | **juice** | **sweets** | **chips** |
| **Sofia** | 0.50 | 0.80 | 1.20 | 1.45 | 1.60 |
| **Plovdiv** | 0.40 | 0.70 | 1.15 | 1.30 | 1.50 |
| **Varna** | 0.45 | 0.70 | 1.10 | 1.35 | 1.55 |

Write a program that accepts a product (string), city (string) and quantity (decimal number) and calculates and prints how much the corresponding quantity of the selected product costs in the specified city.

Sample input:

|  |  |
| --- | --- |
| **Input** | **Output** |
| **coffee**  **Varna**  **2** | 0.90 |
| **chips**  **Plovdiv**  **1** | 1.50 |
| **juice**  **Sofia**  **6** | 7.20 |

# Number in the range

Write a program that checks whether the received number is in the range [-100, 100] and is different from 0 and outputs "Yes" if it meets the conditions, or "No" if it is outside them.

Sample input:

|  |  |
| --- | --- |
| **Input** | **Output** |
| **-25** | Yes |
| **0** | No |
| **25** | Yes |

# Simple Calculator

Write a function that receives two numbers and an operation and prints the result of it.

* add +
* subtract -
* divide /
* multiply \*

Format the result to two decimal places.

Sample input:

|  |  |
| --- | --- |
| **Input** | **Output** |
| **5**  **5**  **add** | 10 |
| **10**  **12**  **subtract** | -2 |
| **9**  **3**  **divide** | 3 |
| **5**  **2**  **divide** | 2.5 |
| **3.1**  **0.1**  **multiply** | 0.31 |

# Vegetable Market

The vegetable market works on working days at the following prices:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **vegetable** | **tomato** | **onion** | **lettuce** | **cucumber** | **pepper** |
| **price** | 2.50 | 1.20 | 0.85 | 1.45 | 5.50 |

Saturday and Sunday the stock exchange operates at higher prices:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **vegetable** | **tomato** | **onion** | **lettuce** | **cucumber** | **pepper** |
| **price** | 2.80 | 1.30 | 0.85 | 1.75 | 3.50 |

Write a program that accepts vegetable (**tomato / onion / lettuce / cucumber / pepper**), day of the week (**Monday / Tuesday / Wednesday / Thursday / Friday / Saturday / Sunday**) and quantity (real number) and calculates the price according to the prices in the tables above.

* Print the result **rounded by 2 decimal places**.
* In case of an invalid day of the week or invalid vegetable name, print "error".

Sample input:

|  |  |
| --- | --- |
| **Input** | **Output** |
| **tomato**  **Tuesday**  **2** | 5.00 |
| **onion**  **Sunday**  **3** | 3.90 |
| **pepper**  **Monday**  **10** | 55.00 |
| **banana**  **Friday**  **5** | error |

# Holiday

A young programmer has a certain budget and free time in a given season. Write a program that accepts the budget and the season and calculates where the programmer will go on vacation and how much he will spend from his budget.

The budget determines the destination, and the season determines how much of the budget he will spend. If it's summer, he'll be camping, in winter he is going to a hotel.

If he is in Asia, regardless of the season he will rest in a hotel. Each campsite or hotel, according to the destination, has its own price which corresponds to a given percentage of the budget:

* At 100lv. or less – somewhere in Bulgaria
  + Summer – 30% of the budget
  + Winter – 70% of the budget
* At 1000lv. Somewhere in Europe, somewhere in Europe.
  + Summer – 40% of the budget
  + Winter – 80% of the budget
* With more than 1000lv. Somewhere in Asia
  + On vacation in Asia, regardless of the season will spend 90% of the budget.

Input:

The function takes 2 parameters:

* First parameter – Budget, real number in the range [10.00...5000.00].
* Second parameter – One of two possible seasons: "summer" or "winter"

Output:

Two lines must be printed on the console.

* First line – "Somewhere in [destination]" among "Bulgaria", "Europe" and "Asia"
* Second line – "{Type of holiday} – {Amount spent}"

The holiday can be on "Camp" or "Hotel". The sum must be rounded to the nearest second character after the comma.

Sample input:

|  |  |
| --- | --- |
| **Input** | **Output** |
| **50**  **summer** | Somewhere in Bulgaria  Camp - 15.00 |
| **75**  **winter** | Somewhere in Bulgaria  Hotel - 52.50 |
| **312**  **summer** | Somewhere in Europe  Camp - 124.80 |
| **678.53**  **winter** | Somewhere in Bulgaria  Hotel - 542.82 |
| **1500**  **summer** | Somewhere in Asia  Hotel - 1350.00 |

# Makeup Shop

Write a program that calculates the profit from the order in a makeup shop.

Makeup prices:

* powder - 2.60 lv.
* lipstick - 3 lv.
* spiral - 4.10 lv.
* shadows - 8.20 lv.
* concealer - 2 lv.

If the ordered makeup is 50 or more, the store makes a discount of 25% of the total price. Of the money earned, the shop must give 10% for the rent of the store. Find out if the money will be enough to do the renovation.

Input

You get 6 parameters:

* Price of the renovation - a real number in the range [1.00 ... 10000.00]
* Number of powders - integer in the range [0... 1000]
* Number of lipsticks - an integer in the range [0 ... 1000]
* Number of spirals - an integer in the range [0 ... 1000]
* Number of shadows - an integer in the range [0 ... 1000]
* Number of correctors - integer in the range [0 ... 1000]

Output

Print on the Console:

* If the money is enough, print:
  + "Yes! {remaining money} lv left."
* If the money is NOT enough, print:
  + "Not enough money! {the lack of money} lv needed."

The result must be formatted to two decimal places.

Sample input:

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **explanation** |
| **40.8**  **20**  **25**  **30**  **50**  **10** | Yes! 418.20 lv left. | Amount: 20 \* 2.60 + 25 \* 3 + 30 \* 4.10 + 50 \* 8.20 + 10 \* 2 = 680 BGN  Number of makeup: 20 + 25 + 30 + 50 + 10 = 135  135 > 50 => 25% discount; 25% of 680 = 170 BGN discount  End price: 680 – 170 = 510 BGN.  Rent: 10% from 510 BGN = 51  Profit: 510 – 51 = 459 BGN  459 > 40.8 => 459 – 40.8 = 418.20 BGN Remain |
| **320**  **8**  **2**  **5**  **5**  **1** | Not enough money! 238.73 lv needed. | Total: 90.3 BGN  Makeup count: 21  21 < 50 => no discount  Rent: 10% from 90.3 = 9.03 BGN  Profit: 90.3 – 9.03 = 81.27 BGN  81.27 < 320 => 320 – 81.27 = 238.73 BGN They do not reach |

# At sea

Calculate how much will cost a vacation. There are the following types of accommodation, with the following prices for staying:

* **"single room"** – 25.00 BGN per night
* **"apartment"** – 50.00 BGN per night
* **"presidential"** – 100.00 BGN per night

Regarding the number of days of the vacation (example: 11 days = 10 nights) and the type of room chosen, there is a different discount. The discounts are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **room** | **under 10 days** | **between 10 and 15** | **over 15 days** |
| **single room** | No reduction | No reduction | No reduction |
| **apartment** | 30% of the final price | 35% of the final price | 50% of the final price |
| **presidential** | 10% of the final price | 15% of the final price | 20% of the final price |

After the stay, the assessment of the hotel's services may be positive or negative.

* If the assessment is **positive**, **add 25%** to the **price with the already deducted reduction**.
* If the assessment is **negative, deduct 10% from the price.**

You take 3 parameters:

* First - days to stay - integer in the range [0...365]
* Second - type of room - "single room", "apartment" or "president apartment"
* Third - assessment - "positive" or "negative"

One line should be printed on the console:

* The price for the stay at the hotel**, formatted to two** decimal places.

Sample input:

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **explanation** |
| **11**  **apartment**  **positive** | 264.06 | 11 days => 10 nights => 10 \* 50.00 = 500 BGN  discount for days => 500 - 175 = 325  Rating positive -> 325 + 81.25  Total: 406.25 |
| **30**  **presidential**  **negative** | 2088.00 |  |
| **12**  **single room**  **positive** | 343.75 |  |
| **2**  **apartment**  **positive** | 43.75 |  |

case "apartment":

totalPrice = (days - 1) \* 50.00;

if (days < 10)

{

totalPrice = totalPrice - totalPrice \* 30 / 100;

}

else if (days <= 15)

{

totalPrice = totalPrice - totalPrice \* 35 / 100;

}

else

{

totalPrice = totalPrice - totalPrice \* 50 / 100;

}

break;

# Grade Calculator

Write a console program that takes a percentage score (0 to 100) as input and outputs the corresponding grade:

* A: 90-100%
* B: 80-89%
* C: 70-79%
* D: 60-69%
* F: 0-59%

|  |  |
| --- | --- |
| **Input** | **Output** |
| **95** | A |
| **82** | B |
| **76** | C |
| **65** | D |
| **45** | F |

# Leap Year Checker

Write a console program that takes a year as input and tells the user if it's a leap year or not.

Note: A leap year is divisible by 4. However, years divisible by 100 are not leap years, unless they are also divisible by 400.

|  |  |
| --- | --- |
| **Input** | **Output** |
| **2020** | It‘s a leap year! |
| **1900** | It‘s not a leap year. |
| **2000** | It‘s a leap year! |
| **2023** | It‘s not a leap year. |
| **1600** | It‘s a leap year! |

# Movie Ticket Price

Write a console program that determines the price of a movie ticket based on age:

* Child (0-12 years): $5
* Teen (13-19 years): $8
* Adult (20+ years): $10

|  |  |
| --- | --- |
| **Input** | **Output** |
| **10** | $5 |
| **16** | $8 |
| **25** | $10 |
| **13** | $8 |

# Days in a Month

Write a console program that takes a month number (1 to 12) as input and outputs the number of days in that month. Assume it's not a leap year.

|  |  |
| --- | --- |
| **Input** | **Output** |
| **1** | 31 |
| **2** | 28 |
| **4** | 30 |
| **7** | 31 |

# University Admissions

Write a console program to determine if a student is admitted to the university based on their score and extracurricular activities:

* Score >= 90: Admitted regardless of extracurriculars.
* Score 80-89: Admitted if they have >= 2 extracurriculars.
* Score < 80: Not admitted.

|  |  |
| --- | --- |
| **Input** | **Output** |
| **85**  **3** | Admitted |
| **88**  **1** | Not admitted |
| **91**  **0** | Admitted |

# Discount Calculator

Write a console program that calculates the discount a customer receives based on their age and if they have a membership card:

* Age < 18: 10% discount.
* Age 18-64:
  + With membership: 20% discount.
  + Without membership: 10% discount.
* Age 65+: 30% discount.

|  |  |
| --- | --- |
| **Input** | **Output** |
| **20**  **Yes** | 20% discount |
| **15**  **No** | 10% discount |
| **70**  **No** | 30% discount |

# Movie Classification

Determine the movie category a person can watch based on their age:

* Age < 13: Only U-rated movies.
* Age 13-17: U and PG-13 rated movies.
* Age 18+: All movies.

|  |  |
| --- | --- |
| **Input** | **Output** |
| **10** | U-rated movies |
| **16** | U and PG-13 rated movies |
| **21** | All movies |

# Airline Luggage Charges

Write a console program that calculates luggage charges based on weight and dimensions:

* If weight > 50kg: $100 overweight fee.
* If the sum of all dimensions (length + width + height) > 158cm:
  + If the sum exceeds by 1-20cm: $50 oversize fee.
  + If the sum exceeds by 21-50cm: $100 oversize fee.
  + If the sum exceeds by more than 50cm: $200 oversize fee.
* If both overweight and oversize: additional $50 handling fee.

|  |  |
| --- | --- |
| **Input** | **Output** |
| **52**  **160** | $150 (Overweight + Slightly oversize) |
| **48**  **180** | $100 (Oversize) |
| **55**  **190** | $250 (Overweight + Oversize + Handling) |

# Adventure Game: Path Decision

You're designing a text-based adventure game. At a certain point, players have to choose a path based on the tools they have in right and left hand:

* If they have a 'sword':
  + If they also have a 'shield': Take the path to the castle.
  + Else: Take the path to the forest.
* If they have a 'map':
  + If they also have 'coins': Go to the town.
  + Else: Camp at the current spot and prepare for the next day.
* If they don't have any of these tools: Wander aimlessly.

|  |  |
| --- | --- |
| **Input** | **Output** |
| **sword**  **shield** | Path to the castle |
| **map**  **coins** | Go to the town |
| **torch**  **flower** | Wander aimlessly |

# Potion Brewing Decision

Players have 2 ingredients to brew potions. Decide which potion they can brew:

* If they have 'herbs':
  + If they also have 'water': Brew a health potion.
  + Else if they have 'oil': Brew a stealth potion.
  + Else: Brew a minor stamina potion.
* If they have 'berries':
  + If they also have 'sugar': Brew a speed potion.
  + Else: Brew a minor energy potion.
* Else: Can't brew any potion.

|  |  |
| --- | --- |
| **Input** | **Output** |
| **herbs**  water | Health potion |
| **herbs**  oil | Stealth potion |
| **herbs**  **banana** | Minor stamina potion |
| **berries**  **sugar** | Speed potion |
| **berries**  **banana** | Minor energy potion |
| **herbs**  **sugar** | Minor stamina potion |

# Survival in the Wilderness

Players need to decide on their course of action based on time of day, environment, and items:

* If it's 'day':
  + If they're in a 'forest':
    - If they have 'knife': Hunt for food.
    - Else if they have 'container': Collect berries.
    - Else: Explore.
  + If they're in a 'desert':
    - If they have 'hat': Search for water.
    - Else: Find shade.
* If it's 'night':
  + If they're in a 'forest':
    - If they have 'firestarter': Make a campfire.
    - Else: Climb a tree for safety.
  + If they're in a 'desert':
    - If they have 'blanket': Sleep.
    - Else: Keep moving to stay warm.

|  |  |
| --- | --- |
| **Input** | **Output** |
| **day**  **forest**  **knife** | Hunt for food |
| **day**  forest  container | Collect berries |
| **night**  **forest**  **firestarter** | Make a campfire |
| **night**  **desert**  **blanket** | Sleep |
| **day**  **desert**  **hat** | Search for water |
| **night**  **desert**  **sword** | Keep moving to stay warm |

# Climate Zone Identifier

Different areas on Earth have specific climate zones based on latitude. The first line contains a latitude value (between -90 and 90). The second line contains either "N" for Northern Hemisphere or "S" for Southern Hemisphere. The application should identify the climate zone.

* Arctic Zone: greater than 66.5° (N/S)
* Temperate Zone: between 23.5° and 66.5° (N/S)
* Tropic Zone: between 0° and 23.5° (N/S)
* Equator: exactly 0°

|  |  |
| --- | --- |
| **Input** | **Output** |
| **70**  **N** | Arctic Zone |
| **45**  **S** | Temperate Zone |
| **10**  **N** | Tropic Zone |
| **0**  **N** | Equator |
| **-85**  **S** | Arctic Zone |

# Architectural Era Identifier

Different eras in history had specific architectural styles. The first line contains the year a building was constructed. The second line contains the primary material used in the building ("wood", "stone", or "steel"). The application should attempt to guess the architectural era.

* Ancient: year < 500 and material = "stone"
* Medieval: year between 500 and 1500 and material = "stone"
* Colonial: year between 1500 and 1800 and material = "wood"
* Industrial: year between 1800 and 1900 and material = "steel"
* Modern: year > 1900 and material = "steel"
* Uncertain: Any other combinations

|  |  |
| --- | --- |
| **Input** | **Output** |
| **300**  **stone** | Ancient |
| **1500**  **wood** | Colonial |
| **1500**  **stone** | Medieval |
| **2000**  **steel** | Modern |
| **1100**  **wood** | Uncertain |