Exercise 6 - Retirement Calculator

Your computer knows what the current year is, which means you can incorporate that into your programs. You just have to figure out how your programming language can provide you with that information.

Create a program that determines how many years you have left until retirement and the year you can retire. It should prompt for your current age and the age you want to retire and display the output as shown in the example that follows.

Example Output

```
What is your current age? 25
At what age would you like to retire? 65
You have 40 years left until you can retire.
It's 2015, so you can retire in 2055.
```

Constraints

- Again, be sure to convert the input to numerical data before doing any math.
- Don’t hard-code the current year into your program. Get it from the system time via your programming language.

Challenge

- Handle situations where the program returns a negative number by stating that the user can already retire.

Exercise 7 - Area of a Rectangular Room

When working in a global environment, you’ll have to present information in both metric and Imperial units. And you’ll need to know when to do the conversion to ensure the most accurate results.

Create a program that calculates the area of a room. Prompt the user for the length and width of the room in feet. Then display the area in both square feet and square meters.

Example Output

```
What is the length of the room in feet? 15
What is the width of the room in feet? 20
You entered dimensions of 15 feet by 20 feet.
The area is
300 square feet
27.871 square meters
```

The formula for this conversion is m2 = f2 × 0.09290304

Constraints

- Keep the calculations separate from the output.
- Use a constant to hold the conversion factor.

Challenges

- Revise the program to ensure that inputs are entered as numeric values. Don’t allow the user to proceed if the value entered is not numeric.
- Create a new version of the program that allows you to choose feet or meters for your inputs.
- Implement this program as a GUI program that automatically updates the values when any value changes.

Exercise 8 - Pizza Party

Division isn’t always exact, and sometimes you’ll write programs that will need to deal with the leftovers as a whole number instead of a decimal.

Write a program to evenly divide pizzas. Prompt for the number of people, the number of pizzas, and the number of slices per pizza. Ensure that the number of pieces comes out even. Display the number of pieces of pizza each person should get. If there are leftovers, show the number of leftover pieces.

Example Output

```
How many people? 8
How many pizzas do you have? 2
How many slices per pizza? 8
8 people with 2 pizzas (16 slices)
Each person gets 2 pieces of pizza.
There are 0 leftover pieces.
```

Challenges

- Revise the program to ensure that inputs are entered as numeric values. Don’t allow the user to proceed if the value entered is not numeric.
- Alter the output so it handles pluralization properly, for example: "Each person gets 2 pieces of pizza." or "Each person gets 1 piece of pizza." Handle the output for leftover pieces appropriately as well.
- Create a variant of the program that prompts for the number of people and the number of pieces each person wants, and calculate how many full pizzas you need to purchase.

Exercise 9 - Paint Calculator

Sometimes you have to round up to the next number rather than follow standard rounding rules.

Calculate gallons of paint needed to paint the ceiling of a room. Prompt for the length and width, and assume one gallon covers 350 square feet. Display the number of gallons needed to paint the ceiling as a whole number.

Example Output

```
You will need to purchase 2 gallons of paint to cover 360 square feet.
```

Remember, you can't buy a partial gallon of paint. You must round up to the next whole gallon.

Constraints

- Use a constant to hold the conversion rate.
- Ensure that you round up to the next whole number.

Challenges

- Revise the program to ensure that inputs are entered as numeric values. Don't allow the user to proceed if the value entered is not numeric.
- Implement support for a round room.
- Implement support for an L-shaped room.
- Implement a mobile version of this app so it can be used at the hardware store.

Exercise 10 - Self-Checkout

Working with multiple inputs and currency can introduce some tricky precision issues.

Create a simple self-checkout system. Prompt for the prices and quantities of three items. Calculate the subtotal of the items. Then calculate the tax using a tax rate of 5.5%. Print out the line items with the quantity and total, and then print out the subtotal, tax amount, and total.

Example Output

```
Enter the price of item 1: 25
Enter the quantity of item 1: 2
Enter the price of item 2: 10
Enter the quantity of item 2: 1
Enter the price of item 3: 4
Enter the quantity of item 3: 1
Subtotal: $64.00
Tax: $3.52
Total: $67.52
```

Constraints

- Keep the input, processing, and output parts of your program separate. Collect the input, then do the math operations and string building, and then print out the output.
- Be sure you explicitly convert input to numerical data before doing any calculations.

Challenges

- Revise the program to ensure that prices and quantities are entered as numeric values. Don't allow the user to proceed if the value entered is not numeric.
- Alter the program so that an indeterminate number of items can be entered. The tax and total are computed when there are no more items to be entered.

Exercise 11 - Currency Conversion

At some point, you might have to deal with currency exchange rates, and you'll need to ensure your calculations are as precise as possible.

Write a program that converts currency. Specifically, convert euros to U.S. dollars. Prompt for the amount of money in euros you have, and prompt for the current exchange rate of the euro. Print out the new amount in U.S. dollars.

The formula for currency conversion is derived from

```
c_to / c_frame = rate
```

where

- c_to is the amount in U.S. dollars.
- c_from is the amount in euros.
- rate is the per-unit current exchange rate from c_from to c_to.

Example Output

```
How many euros are you exchanging? 81
What is the exchange rate? 1.3751
81 euros at an exchange rate of 1.3751 is
111.38 U.S. dollars.
```

Constraints

- Ensure that fractions of a cent are rounded up to the next penny.
- Use a single output statement.

Challenges

- Build a dictionary of conversion rates and prompt for the countries instead of the rates.
- Wire up your application to an external API that provides the current exchange rates.

Exercise 12 - Computing Simple Interest

Computing simple interest is a great way to quickly figure out whether an investment has value. It's also a good way to get comfortable with explicitly coding the order of operations in your programs.

Create a program that computes simple interest. Prompt for the principal amount, the rate as a percentage, and the time, and display the amount accrued (principal + interest).

The formula for simple interest is A = P(1 + rt), where P is the principal amount, r is the annual rate of interest, t is the number of years the amount is invested, and A is the amount at the end of the investment.

Example Output

```
Enter the principal: 1500
Enter the rate of interest: 4.3
Enter the number of years: 4
After 4 years at 4.3%, the investment will be worth $1758.
```

Constraints

- Prompt for the rate as a percentage (like 15, not .15). Divide the input by 100 in your program.
- Ensure that fractions of a cent are rounded up to the next penny.
- Ensure that the output is formatted as money.

Challenges

- Ensure that the values entered for principal, rate, and number of years are numeric and that the program will not let the user proceed without valid inputs.
- Alter the program to use a function called calculateSimpleInterest that takes in the rate, principal, and number of years and returns the amount at the end of the investment.
- In addition to printing out the final amount, print out the amount at the end of each year.

Exercise 13 - Determining Compound Interest

Simple interest is something you use only when making a quick guess. Most investments use a compound interest formula, which will be much more accurate. And this formula requires you to incorporate exponents into your programs.

Write a program to compute the value of an investment compounded over time. The program should ask for the starting amount, the number of years to invest, the interest rate, and the number of periods per year to compound.

The formula you'll use for this is A = P(1 + r/n)^(n*t) where

- P is the principal amount.
- r is the annual rate of interest.
- t is the number of years the amount is invested.
- n is the number of times the interest is compounded per year.
- A is the amount at the end of the investment.

Example Output

```
What is the principal amount? 1500
What is the rate? 4.3
What is the number of years? 6
What is the number of times the interest is compounded per year? 4
$1500 invested at 4.3% for 6 years compounded 4 times per year is $1938.84.
```

Constraints

- Prompt for the rate as a percentage (like 15, not .15). Divide the input by 100 in your program.
- Ensure that fractions of a cent are rounded up to the next penny.

- Ensure that the output is formatted as money.

Challenges

- Ensure that all of the inputs are numeric and that the program will not let the user proceed without valid inputs.
- Create a version of the program that works in reverse, so you can determine the initial amount you'd need to invest to reach a specific goal.
- Implement this program as a GUI app that automatically updates the values when any value changes.

Exercise 14 - Tax Calculator

You don't always need a complex control structure to solve simple problems. Sometimes a program requires an extra step in one case, but in all other cases there's nothing to do.

Write a simple program to compute the tax on an order amount. The program should prompt for the order amount and the state. If the state is "WI," then the order must be charged 5.5% tax. The program should display the subtotal, tax, and total for Wisconsin residents but display just the total for non-residents.

Example Output

```
What is the order amount? 10
What is the state? WI
The subtotal is $10.00.
The tax is $0.55.
The total is $10.55.
```

Or

What is the order amount? 10
What is the state? MN
The total is $10.00

Constraints

- Implement this program using only a simple if statement—don't use an else clause.
- Ensure that all money is rounded up to the nearest cent.
- Use a single output statement at the end of the program to display the program results.

Challenges

- Allow the user to enter a state abbreviation in upper, lower, or mixed case.
- Also allow the user to enter the state's full name in upper, lower, or mixed case.

Exercise 15 - Password Validation

Passwords are validated by comparing a user-provided value with a known value that's stored. Either it's correct

or it's not.

Create a simple program that validates user login credentials. The program must prompt the user for a username and password. The program should compare the password given by the user to a known password. If the password matches, the program should display "Welcome!" If it doesn't match, the program should display "I don't know you."

Example Output

```
What is the password? 12345
I don't know you.
```

Or

```
What is the password? abc$123
Welcome!
```

Constraints

- Use an if/else statement for this problem.
- Make sure the program is case sensitive.

Challenges

- Investigate how you can prevent the password from being displayed on the screen in clear text when typed.
- Create a map of usernames and passwords and ensure the username and password combinations match.
- Encode the passwords using Bcrypt and store the hashes in the map instead of the clear-text passwords. Then, when you prompt for the password, encrypt the password using Bcrypt and compare it with the value in your map.

Program 16 - Legal Driving Age

You can test for equality, but you may need to test to see how a number compares to a known value and display a message if the number is too low or too high.

Write a program that asks the user for their age and compare it to the legal driving age of sixteen. If the user is sixteen or older, then the program should display "You are old enough to legally drive." If the user is under sixteen, the program should display "You are not old enough to legally drive."

Example Output

```
What is your age? 15
You are not old enough to legally drive.
```

Or

```
What is your age? 35
You are old enough to legally drive.
```

Constraints

- Use a single output statement.
- Use a ternary operator to write this program. If your language doesn't support a ternary operator, use a regular if/else statement, and still use a single output statement to display the message.

Challenges

- If the user enters a number that's less than zero or enters non-numeric data, display an error message that asks the user to enter a valid age.
- Instead of hard-coding the driving age in your program logic, research driving ages for various countries and create a lookup table for the driving ages and countries. Prompt for the age, and display which countries the user can legally drive in.

Exercise 17 - Blood Alcohol Calculator

Sometimes you have to perform a more complex calculation based on some provided inputs and then use that result to make a determination.

Create a program that prompts for your weight, gender, total alcohol consumed (in ounces), and the amount of time since your last drink. Calculate your blood alcohol content (BAC) using this formula

```
BAC = (A × 5.14 / W × r) − .015 × H
```

where

- A is total alcohol consumed, in ounces (oz).
- W is body weight in pounds.
- r is the alcohol distribution ratio:
  - 0.73 for men
  - 0.66 for women
- H is number of hours since the last drink.

Display whether or not it's legal to drive by comparing the blood alcohol content to 0.08.

Example Output

```
Enter a 1 is you are male or a 2 if you are female: 1
How many ounces of alcohol did you have? 3
What is your weight, in pounds? 175
How many hours has it been since your last drink? 1

Your BAC is 0.049323
It is legal for you to drive.
```

```
Enter a 1 is you are male or a 2 if you are female: 1
How many ounces of alcohol did you have? 5
What is your weight, in pounds? 175
```

```
How many hours has it been since your last drink? 1

Your BAC is 0.092206
It is not legal for you to drive.
```

Constraint

- Prevent the user from entering non-numeric values.

Challenges

- Handle metric units.
- Look up the legal BAC limit by state and prompt for the state. Display a message that states whether or not it's legal to drive based on the computed BAC.
- Develop this as a mobile application that makes it easy to record each drink, updating the BAC each time a drink is entered.

Exercise 18 - Temperature Converter

You'll often need to determine which part of a program is run based on user input or other events.

Create a program that converts temperatures from Fahrenheit to Celsius or from Celsius to Fahrenheit. Prompt for the starting temperature. The program should prompt for the type of conversion and then perform the conversion.

The formulas are

```
C = (F − 32) × 5 / 9
```

and

```
F = (C × 9 / 5) + 32
```

Example Output

Press C to convert from Fahrenheit to Celsius.
Press F to convert from Celsius to Fahrenheit.
Your choice: C
Please enter the temperature in Fahrenheit: 32
The temperature in Celsius is 0.

Constraints

- Ensure that you allow upper or lowercase values for C and F.
- Use as few output statements as possible and avoid repeating output strings.

Challenges

- Revise the program to ensure that inputs are entered as numeric values. Don't allow the user to proceed if

the value entered is not numeric.
- Break the program into functions that perform the computations.
- Implement this program as a GUI program that automatically updates the values when any value changes.
- Modify the program so it also supports the Kelvin scale.

Exercise 19 - BMI Calculator

You'll often need to see if one value is within a certain range and alter the flow of a program as a result.

Create a program to calculate the body mass index (BMI) for a person using the person's height in inches and weight in pounds. The program should prompt the user for weight and height.

Calculate the BMI by using the following formula:

```
bmi = (weight / (height × height)) * 703
```

If the BMI is between 18.5 and 25, display that the person is at a normal weight. If they are out of that range, tell them if they are underweight or overweight and tell them to consult their doctor.

Example Output

```
Your BMI is 19.5.
You are within the ideal weight range.
```

or

```
Your BMI is 32.5.
You are overweight. You should see your doctor.
```

Constraint

- Ensure your program takes only numeric data. Don't let the user continue unless the data is valid.

Challenges

- Make the user interface accept height and weight in Imperial or metric units. You'll need a slightly different formula for metric units.
- For Imperial measurements, prompt for feet and inches and convert feet to inches so the user doesn't have to.
- Use a GUI interface with sliders for height and weight. Update the user interface on the fly. Use colors as well as text to indicate health.

Exercise 20 - Multistate Sales Tax Calculator

More complex programs may have decisions nested in other decisions, so that when one decision is made, additional decisions must be made.

Create a tax calculator that handles multiple states and multiple counties within each state. The program prompts the user for the order amount and the state where the order will be shipped.

- Wisconsin residents must be changed 5% sales tax with an additional county-level charge. For Wisconsin residents, prompt for the county of residence.
  - For Eau Claire county residents, add an additional 0.005 tax.
  - For Dunn county residents, add an additional 0.004 tax.
- Illinois residents must be charged 8% sales tax with no additional county-level charge.
- All other states are not charged tax.

The program then displays the tax and the total for Wisconsin and Illinois residents but just the total for everyone else.

Example Output

```
What is the order amount? 10
What state do you live in? Wisconsin
What county do you live in? Dane
The tax is $0.50.
The total is $10.50.
```

Constraints

- Ensure that all money is rounded up to the nearest cent.
- Use a single output statement at the end of the program to display the program results.

Challenges

- Add support for your state and county.
- Allow the user to enter a state abbreviation and county name in upper, lower, or mixed case.
- Allow the user to also enter the state's full name in upper, lower, or mixed case.
- Implement the program using data structures to avoid nested if statements.

Exercise 21 - Numbers to Names

Many programs display information to the end user in one form but use a different form inside the program. For example, you may show the word Blue on the screen, but behind the scenes you'll have a numerical value for that color or an internal value because you may need to represent the textual description in another language for Spanish-speaking visitors.

Write a program that converts a number from 1 to 12 to the corresponding month. Prompt for a number and display the corresponding calendar month, with 1 being January and 12 being December. For any value outside that range, display an appropriate error message.

Example Output

```
Please enter the number of the month: 3
The name of the month is March.
```

Constraints

- Use a switch or case statement for this program.
- Use a single output statement for this program.

Challenges

- Use a map or dictionary to remove the switch statement from the program.
- Support multiple languages. Prompt for the language at the beginning of the program.

Exercise 22 - Comparing Numbers

Comparing one input to a known value is common enough, but you'll often need to process a collection of inputs.

Write a program that asks for three numbers. Check first to see that all numbers are different. If they're not different, then exit the program. Otherwise, display the largest number of the three.

Example Output

```
Enter the first number: 1
Enter the second number: 51
Enter the third number: 2
The largest number is 51.
```

Constraint

- Write the algorithm manually. Don't use a built-in function for finding the largest number in a list.

Challenges

- Modify the program so that all entered values are tracked and the user is prevented from entering a number that's already been entered.
- Modify the program so that it asks for ten numbers instead of three.
- Modify the program so that it asks for an unlimited number of numbers.

Exercise 23 - Troubleshooting Car Issues

An expert system is a type of artificial intelligence program that uses a knowledge base and a set of rules to perform a task that a human expert might do. Many websites are available that will help you self-diagnose a medical issue by answering a series of questions. And many hardware and software companies offer online troubleshooting tools to help people solve simple technical issues before calling a human.

Create a program that walks the user through troubleshooting issues with a car. Use the following decision tree to build the system:

```
@startuml
```

```
if (Is the car silent when you turn the key?) then (yes)
    if (Are the battery terminals corroded?) then (yes)
        :Clean terminals and try starting again.;
        stop
    else (no)
        :Replace cables and try again.;
        stop
    endif
else (no)
    if (Does the car make a slicking noise?) then (yes)
        :Replace the battery.;
        stop
    else (no)
        if (Does the car crank up but fail to start?) then (yes)
            :Check spark plug connections.;
            stop
        else (no)
            if (Does the engine start and then die?) then (yes)
                if (Does you car have fuel injection?) then (yes)
                    :Get it in for service.;
                    stop
                else (no)
                    :Check to ensure the choke is opening and closing.;
                    stop
                endif
            else (no)
                :This should not be possible.;
                stop
            endif
        endif
    endif
endif
@enduml
```

Example Output

```
Is the car silent when you turn the key? y
Are the battery terminals corroded? n
Replace cables and try again.
```

Constraint

Ask only questions that are relevant to the situation and to the previous answers. Don’t ask for all inputs at once.

Challenge

Investigate rules engines and inference engines. These are powerful ways to solve complex problems that are based on rules and facts. Identify a rules engine for your programming language and use it to solve this problem