Introduction to C++ Programming
Problem Collection

Ali Kashefi
kashefi@stanford.edu

Each problem is accompanied by a gray box that illustrates the exact format for the sample input and output. It is crucial for you to understand that when you submit your homework, only the input and output values should appear in the console. Do not include additional prompts or messages like "Please enter the x coordinate" or anything else.

You are **only allowed** to use the following C++ standard libraries:

```
#include <iostream>
#include <string>
#include <cmath>
#include <vector>
```

Problem 1. Write a C++ program that does the following:

- Takes a name as input from the user.
- Outputs "Hello, [User's Name]!" to the console.

This is an example of how the program should work:

Sample Input:

```
Alice
```

Sample Output:

```
Hello, Alice!
```

Sample Input:

```
Erin
```

Sample Output:

```
Hello, Erin!
```

Problem 2. Write a C++ program that takes a positive integer as input and checks if the number is odd or even. If the number is odd, the program should print `odd`. If the number is even, the program should print `even`. As a hint, an even number, when divided by 2, has a remainder of 0, while an odd number, when divided by 2, has a remainder of 1.

Sample Input:

```
101
```

Sample Output:

```
odd
```

Sample Input:

```
24
```

Sample Output:

```
even
```

Problem 3. The distance between a given point (x, y) and a fixed point $(4, 8)$ on a plane can be calculated using the distance formula: $d = \sqrt{(4-x)^2 + (8-y)^2}$. Write a C++ program that takes the coordinates of a point as input, and then outputs the distance between this point and the fixed point $(4, 8)$. Your program should follow the following format. The first number is x and the second number is y. Do not use any `endl` in your code.

Sample input:

```
2.0

3.0
```

Sample output:

```
5.38516
```

Sample input:

```
4.0

8.0
```

Sample output:

```
0
```

Sample input:

```
1

4
```

Sample output:

```
5
```

Problem 4. Write a C++ program that takes four distinct (in terms of their absolute values) integers as input and outputs the one with the maximum absolute value. Do not use any `endl` in your code.

Sample Input:

```
10

88

4

2024
```

Sample Output:

```
2024
```

Sample Input:

```
20

-8

14

-21
```

Sample Output:

```
-21
```

Sample Input:

```
-202

-101

0

-11
```

Sample Output:

```
-202
```

Problem 5. Write a C++ program that accepts three positive real numbers as input and outputs `yes` if these numbers can form the sides of a triangle, or `no` otherwise. Do not use any `endl` in your code. For three numbers to be the lengths of the sides of a triangle, they must satisfy the Triangle Inequality Theorem. This theorem states that the sum of the lengths of any two sides of a triangle must be greater than the length of the third side. This condition must be true for all three combinations of the sides. In mathematical terms, if the lengths of the sides are a, b, and c, then the following three conditions must be met for these lengths to form a triangle:

- $a + b > c$
- $a + c > b$
- $b + c > a$

If all three conditions are satisfied, the numbers a, b, and c can be the lengths of the sides of a triangle. If even one of these conditions is not met, a triangle cannot be formed with these side lengths. For example, consider three numbers: 3, 4, and 5.

- The sum of 3 and 4 is 7, which is greater than 5.
- The sum of 3 and 5 is 8, which is greater than 4.
- The sum of 4 and 5 is 9, which is greater than 3.

Sample Input:

```
1.5

1.5

1.5
```

Sample Output:

```
yes
```

Sample Input:

```
2

1

1
```

Sample Output:

```
no
```

Problem 6. Write a C++ program that accepts three positive real numbers as input and outputs `yes` if these numbers can form the sides of a right triangle, or `no` otherwise. Do not use any `endl` in your code. To determine whether three real numbers can form the sides of a right triangle, we use the Pythagorean Theorem. This theorem is a fundamental principle in geometry and states that in a right triangle, the square of the length of the hypotenuse (the side opposite the right angle) is equal to the sum of the squares of the lengths of the other two sides. In mathematical terms, if the lengths of the sides of a triangle are a, b, and c, with c being the length of the hypotenuse, then the triangle is a right triangle if and only if the following equation holds:

$$a^2 + b^2 = c^2$$

It is important to note that for any set of three positive real numbers to form a right triangle, two conditions must be met:

- The Pythagorean Relationship: As mentioned above, the square of the longest side must equal the sum of the squares of the other two sides.
- Triangle Inequality: The numbers must satisfy the Triangle Inequality Theorem, which stipulates that the sum of any two sides of a triangle must be greater than the third side. This ensures that they can indeed form a triangle. It can be shown that the Pythagorean relationship satisfies the triangle inequality condition for real positive numbers. Therefore, your code should focus primarily on the Pythagorean relationship.

As advice, use the strategy for comparing irrational numbers in `C++`.

Sample Input:

```
3.0

5.0

4.0
```

Sample Output:

```
yes
```

Sample Input:

```
1.0

2.0

2.2360679775
```

Sample Output:

```
yes
```

Sample Input:

```
2.0

2.44948974278

1.41421356237
```

Sample Output:

```
yes
```

Sample Input:

```
2.0

2.0

2.0
```

Sample Output:

```
no
```

Sample Input:

```
1.0

2.0

2.31478581012
```

Sample Output:

```
no
```

Problem 7. Write a C++ program that takes a positive integer n as input and prints the factorial of n as the output. The factorial of a positive integer n, denoted as $n!$, is the product of all positive integers less than or equal to n. For example, $4! = 24$ because $4! = 4 \times 3 \times 2 \times 1$. Specific Policy: It is not permitted to use functions to implement this program.

Sample Input:

```
4
```

Sample Output:

```
24
```

Sample Input:

```
8
```

Sample Output:

```
40320
```

Sample Input:

```
1
```

Sample Output:

```
1
```

Sample Input:

```
5
```

Sample Output:

```
120
```

Sample Input:

```
10
```

Sample Output:

```
3628800
```

Problem 8. A prime number is a positive integer greater than 1 that has no positive divisors other than 1 and itself. In other words, a prime number has exactly two distinct positive divisors. For example, 2, 3, 5, 7, 11, 13, 17, 19, 23, and so on, are all prime numbers. Your task is to write a C++ program that takes an integer number as input and checks if it is a prime number or not. If the input number is prime, it prints `yes` to the console. If the input is not prime, it prints `no` to the console.
If you are not familiar with prime numbers, let's go through a few examples to better understand what prime numbers are:

Number: 2

Divisors: 1, 2

Explanation: The number 2 is divisible only by 1 and itself. It has exactly two distinct positive divisors, so it is a prime number.

Number: 7

Divisors: 1, 7

Explanation: The number 7 is divisible only by 1 and itself. It has exactly two distinct positive divisors, so it is a prime number.

Number: 12

Divisors: 1, 2, 3, 4, 6, 12

Explanation: The number 12 has more than two divisors, including 1, 2, 3, 4, 6, and 12. Therefore, it is not a prime number.

Sample Input:

```
8
```

Sample Output:

```
no
```

Sample Input:

```
104729
```

Sample Output:

```
yes
```

Sample Input:

```
0
```

Sample Output:

```
no
```

Sample Input:

```
1
```

Sample Output:

```
no
```

Sample Input:

```
-11
```

Sample Output:

```
no
```

Sample Input:

```
19
```

Sample Output:

```
yes
```

Problem 9. Write a C++ program that computes the "summation" of the Collatz sequence for any positive integer provided by the user. The Collatz Conjecture is a mathematical hypothesis that applies to sequences defined as follows:
Given a number n:

- If n is even, the next term is $n/2$.
- If n is odd, the next term is $3n + 1$.

Repeat the process with the resulting value. The conjecture states that no matter what number you start with, you will always eventually reach 1. Here are a few examples of the Collatz sequence:

- If the starting number is 6, the sequence is: 6, 3, 10, 5, 16, 8, 4, 2, 1
- If the starting number is 7, the sequence is: 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1

Note that the output of your program must be the summation of the sequence. For example, if the input is 6, then the output must be 55, because $6+3+10+5+16+8+4+2+1 = 55$.

Sample Input:

```
6
```

Sample Output:

```
55
```

Sample Input:

```
7
```

Sample Output:

```
288
```

Sample Input:

```
2023
```

Sample Output:

```
268757
```

Problem 10. Write a C++ code that takes a non-negative integer n as input and calculates the n-th Pell number (P_n). The Pell numbers are a sequence of integers defined as:

$$\begin{aligned} P_0 &= 0 \quad \text{if } n = 0, \\ P_1 &= 1 \quad \text{if } n = 1, \\ P_n &= 2P_{n-1} + P_{n-2} \quad \text{if } n \geq 2. \end{aligned}$$

For example, if the input is 2 (i.e., $n = 2$), the output should be 2, because $P_2 = 2P_1 + P_0 = 2(1) + 0 = 2$.
Specific Policy: It is not permitted to use functions to implement this program.

Sample Input:

```
0
```

Sample Output:

```
0
```

Sample Input:

```
1
```

Sample Output:

```
1
```

Sample Input:

```
3
```

Sample Output:

```
5
```

Sample Input:

```
5
```

Sample Output:

```
29
```

Problem 11. Write a C++ program that takes a positive integer and outputs the digit with the highest value.

Sample Input:

```
4
```

Sample Output:

```
4
```

Sample Input:

```
1233
```

Sample Output:

```
3
```

Sample Input:

```
784530281
```

Sample Output:

```
8
```

Sample Input:

```
22222
```

Sample Output:

```
2
```

Problem 12. Write a C++ program that takes a number in base 2 (binary) and returns the corresponding number in base 10 (decimal) as an `integer` number. To convert a number from base 2 (binary) to base 10 (decimal), the process involves multiplying each digit of the binary number by 2 raised to the power of its position, and then summing up all these values. The positions are counted from right to left, starting with 0. If a binary number has digits $b_n b_{n-1} \dots b_2 b_1 b_0$, its decimal equivalent is calculated as:

$$b_0 \times 2^0 + b_1 \times 2^1 + b_2 \times 2^2 + \ldots + b_{n-1} \times 2^{n-1} + b_n \times 2^n$$

Here, b_i represents the digit at position i (either 0 or 1), and n is the highest position number in the binary number. For example, let us convert the binary number 1011 to decimal.

1. Break it down by position: $1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
2. Calculate each term:
   - $1 \times 2^3 = 1 \times 8 = 8$
   - $0 \times 2^2 = 0 \times 4 = 0$
   - $1 \times 2^1 = 1 \times 2 = 2$
   - $1 \times 2^0 = 1 \times 1 = 1$
3. Add them up: $8 + 0 + 2 + 1 = 11$

So, the binary number 1011 is equivalent to the decimal number 11.

Sample Input:

```
1011
```

Sample Output:

```
11
```

Sample Input:

```
101110111
```

Sample Output:

```
375
```

Sample Input:

```
0
```

Sample Output:

```
0
```

Problem 13. Write a C++ program that checks if a positive integer is "strong" based on a specific criterion. In this problem, a "strong" integer is defined as one where each digit, except for the rightmost digit, is greater than half of its immediately following (right) digit. The program should output `yes` if the number is strong according to this definition, and `no` otherwise.

Sample Input:

```
9
```

Sample Output:

```
yes
```

Sample Input:

```
4678990
```

Sample Output:

```
yes
```

Sample Input:

```
986314
```

Sample Output:

```
no
```

Sample Input:

```
1111
```

Sample Output:

```
yes
```

Sample Input:

```
813923
```

Sample Output:

```
no
```

Problem 14. Write a C++ program that takes a string as input from the user and outputs the reverse of that string. For this specific problem, we assume the string does not contain any space.

Sample Input:

```
Programming!
```

Sample Output:

```
!gnimmargorP
```

Sample Input:

```
UniversityOfCaliforniaAtLosAngeles
```

Sample Output:

```
selegnAsoLtAainrofilaCfOytisrevinU
```

Sample Input:

```
A
```

Sample Output:

```
A
```

Problem 15. A number pyramid is a pattern of numbers arranged in a manner where the first line contains 1 number, the second line contains 2 numbers, the third line contains 3 numbers, and so on. The numbers in each line start from 1 and go up to the line number. Write a C++ program that draws a number pyramid of a given height. Note that there is no space between numbers in each line (as can be seen in the following example). Moreover, you must use `endl` at the end of each line. Do **not** use space at the end of each line.

Sample Input:

```
1
```

Sample Output:

```
1
```

Sample Input:

```
3
```

Sample Output:

```
1
12
123
```

Sample Input:

```
5
```

Sample Output:

```
1
12
123
1234
12345
```

Problem 16. Parentheses are symbols used in programming, mathematics, and various written languages to group elements or information together. For this exercise, we are concerned with round brackets: ( ). The term "valid parentheses" in the context of round brackets means:

- Every opening parenthesis (must have a corresponding closing parenthesis). They should correctly pair up.
- Parentheses must close in the correct order, respecting their nesting. An opening parenthesis cannot be closed by another opening one, and a closing parenthesis cannot appear before its corresponding opening one.

Write a C++ program that takes a string as input from the user and checks if the parentheses (i.e., round brackets) in a given string are valid. If it is valid, the output is `yes`, if it is not valid, the output is `no`. For this specific problem, we assume the string does not contain any space.

Sample Input:

```
(HW)(3)((PIC)))(10A)(string))()
```

Sample Output:

```
no
```

Sample Input:

```
(math((()()))Programming(()U)C)L)A(()))((C++))
```

Sample Output:

```
yes
```

Sample Input:

```
((((a)bc)d)efg((h))(i)j)klmno((12345))+-((!!!))(?)?(?)
```

Sample Output:

```
yes
```

Sample Input:

```
)123)4)5(67(8(9)10)1112)13(14(1516(17
```

Sample Output:

```
no
```

Sample Input:

```
):()
```

Sample Output:

```
no
```

Sample Input:

```
AbcdEfgh
```

Sample Output:

```
yes
```

Sample Input:

```
)(..)(
```

Sample Output:

```
no
```

Problem 17. An alphabetic pyramid is a pattern of lowercase letters arranged in a manner where the first line contains the first letter of alphabetic (i.e., `a`), the second line contains the first two lowercase letters, the third line contains the first three letters, and so on. Write a C++ program that draws an alphabetic pyramid of a given height such as `n`, where `n` is a positive integer number less than 27. Note that there is no space between letters in each line (as can be seen in the following example). Moreover, you must use `endl` at the end of each line. Do **not** use space at the end of each line.

Hint: Create a table of information that contains a `string` of alphabetical letters.

Sample Input:

```
3
```

Sample Output:

```
a
ab
abc
```

Sample Input:

```
7
```

Sample Output:

```
a
ab
abc
abcd
abcde
abcdef
abcdefg
```

Sample Input:

```
1
```

Sample Output:

```
a
```

Problem 18. Write a C++ program that takes two space-less strings (with at least one character) as input and merges these two strings. A merge occurs when a substring from the end of the first string exactly matches a substring from the beginning of the second string. The program should combine these two strings into one, overlapping the matching substring, and output the merged string. The merged string should be one with the minimum possible length. If a merge is not possible, the program should output the first string unaltered. For instance, merging "Caltech" with "technology" would result in "Caltechnology". As another example, merging "Apple" with "orange" would result in "Apple". For example, merging "AB===" with "===BA" would lead in "AB===BA", and **not** "AB=====BA". Do not use any `endl` in your code.

Sample Input:

```
Sun

Sunset
```

Sample Output:

```
Sunset
```

Sample Input:

```
Sun

sunset
```

Sample Output:

```
Sun
```

Sample Input:

```
cross

crossroad
```

Sample Output:

```
crossroad
```

Sample Input:

```
sunset

set
```

Sample Output:

```
sunset
```

Sample Input:

```
overlap

lapping
```

Sample Output:

```
overlapping
```

Sample Input:

```
Caltech

technology
```

Sample Output:

```
Caltechnology
```

Sample Input:

```
a

b
```

Sample Output:

```
a
```

Sample Input:

```
%%%$$##

##()@#^&
```

Sample Output:

```
%%%$$##()@#^&
```

Sample Input:

```
A*()*A++==

+==A*()*A
```

Sample Output:

```
A*()*A++==A*()*A
```

Sample Input:

```
AB===

===BA
```

Sample Output:

```
AB===BA
```

Sample Input:

```
Python

Python
```

Sample Output:

```
Python
```

Problem 19. During World War II, armies used various mechanisms for secure communication, often involving complex coding and decoding systems. These systems were crucial for transmitting confidential information while preventing the enemy from intercepting and understanding their messages. In this problem, you are tasked with designing a decoding system inspired by these historical cryptographic methods.

Your task is to write C++ code to decode a message string using a key string composed of unique lowercase letters. The key has a maximum of 9 letters. The message includes letters from the key and digits from 1 to 9, each represented as single-character strings. Per a contract, the message will always end with a letter. Additionally, the message cannot include sequences of two or more digits. The decoding process involves converting each letter in the message to its corresponding position in the key string. Additionally, if a character in the message is a digit, first the right adjacent letter of the digit within the message is identified. Then the position of this letter is identified in the key. Looking at the key and starting at this position, we move cyclically from left to right by a count equal to the digit to reach a character in the key. Finally, the digit is converted into this character.

For example, consider a key `bny` and a message `nbn2y5n`. Here, `n` is translated to `2` (its position in the key). The digit `2` is translated to `n`. Because the next letter of `2` is `y`, and `y` cyclically, moving from left to right in the key, becomes `n` by `2` shifts. In other words, in this sequence, `y` $\rightarrow$ `b`, and `b` $\rightarrow$ `n`. As another example, `5` becomes `b`, because `n` $\rightarrow$ `y`, `y` $\rightarrow$ `b`, `b` $\rightarrow$ `n`, `n` $\rightarrow$ `y`, and `y` $\rightarrow$ `b`. As explained earlier, per the contract, the message must always end with a letter; thus, a message like `nbey2` is not acceptable. Additionally, the message cannot include sequences of two or more digits, and hence a message like `n34y789b` is not acceptable. It is not necessary that your code checks if the message is acceptable or not. This is the problem assumption.

Your code should accept two strings: the first is the key, and the second is the message. It should then output the decoded message. Do not use any `endl` statements in your code.

Policy: You are not permitted to convert strings to integers or vice versa.

Hint: Create two tables of information. The first table is the input key, and the second table is a string of the digits, such as `string number = "123456789";`. Note that if your character is `2` (i.e., `char c = '2';`), you can simply obtain its corresponding `int` number by using `size_t index = number.find(c) + 1;`.

Sample Input:

```
bny

nbn2y5n
```

Sample Output:

```
212n3b2
```

Sample Input:

```
abcde

abcde
```

Sample Output:

```
12345
```

Sample Input:

```
uibtzrqx

z1i2r3b4q5z6x7t8u9x9q8r7i6x5b4z3x2t1z
```

Sample Output:

```
5b2x6r3b7i5r8b4u1u8x7r6u2r8x3u5b8r4r5
```

Sample Input:

```
xzy

1x1yx2z8z
```

Sample Output:

```
z1x31x2x2
```

Sample Input:

```
bcdea

9c9e9d9d8abcde7e
```

Sample Output:

```
b2d4c3c3d51234b4
```

Sample Input:

```
xypoewtm

xpw4mw2y
```

Sample Output:

```
136o86o2
```

Sample Input:

```
xypoewtm

1x2y3p4o5e6w7t8m
```

Sample Output:

```
y1o2w3m4y5o6w7m8
```

Sample Input:

```
xypoewtm

8x8y8p8o8e8w8t8m
```

Sample Output:

```
x1y2p3o4e5w6t7m8
```

Sample Input:

```
xypoewtm

mtweopyx
```

Sample Output:

```
87654321
```

Sample Input:

```
xypoewtm

1x1y1p1o1e1w1t1m
```

Sample Output:

```
y1p2o3e4w5t6m7x8
```

Problem 20. Write a C++ program that takes a non-negative integer n as input and calculates the n-th S number (S_n). The S numbers are a sequence of integers defined as:

$$
\begin{aligned}
&S_0 = 0 \quad \text{for } n = 0,\\
&S_1 = 1 \quad \text{for } n = 1,\\
&S_2 = 1 \quad \text{for } n = 2,\\
&S_n = S_{n-1} + 2S_{n-2} + S_{n-3} \quad \text{for } n \geq 3.
\end{aligned}
$$

Sample Input:

```
0
```

Sample Output:

```
0
```

Sample Input:

```
2
```

Sample Output:

```
1
```

Sample Input:

```
4
```

Sample Output:

```
6
```

Sample Input:

```
5
```

Sample Output:

```
13
```

Problem 21. Write a C++ program that takes a positive integer and outputs the average value of its digits. For example, if the input is `1313`, the output should be `2`. This is calculated as follows: the sum of the digits (1+3+1+3) is 8, and dividing 8 by the number of digits (4) gives an average of 2.

Sample Input:

```
1
```

Sample Output:

```
1
```

Sample Input:

```
1313
```

Sample Output:

```
2
```

Sample Input:

```
97856
```

Sample Output:

```
7
```

Sample Input:

```
22222
```

Sample Output:

```
2
```

Sample Input:

```
2323
```

Sample Output:

```
2.5
```

Sample Input:

```
1000
```

Sample Output:

```
0.25
```

Problem 22. Write a C++ program that takes a positive integer number (n) and prints a right-aligned numeric pattern. The pattern should consist of n lines. On the 1st line, print the number 1 right-aligned with spaces. On the 2nd line, print the numbers 1 and 2 aligned in the same manner, and so on until the nth line, which prints the numbers from 1 to n consecutively. The last line should not have any leading spaces.
Note that "right-aligned spaces" means that the numbers in each line of the output should be aligned along the right margin, and any empty space to the left of these numbers should be filled with the space character and/or string.

Sample Input:

```
3
```

Sample Output:

```
  1
 12
123
```

Sample Input:

```
5
```

Sample Output:

```
    1
   12
  123
 1234
12345
```

Sample Input:

```
1
```

Sample Output:

```
1
```

Problem 23. Write a C++ program that computes the summation of the prime numbers strictly smaller than any positive integer provided by the user. For example, if the input is 11, then the output must be 17 because prime numbers smaller than 11 are 2, 3, 5, and 7. Therefore, the sum is $2 + 3 + 5 + 7 = 17$.

Specific policies:

- You must define at least one function, except `main()`, in your code to handle this problem.
- You must declare the functions with a **prototype** at the beginning, before any function defining or calls, as discussed in class.

Advice: Consider the task of identifying whether a number is prime. Instead of repeating the prime-checking logic every time you need it, you could encapsulate this logic within a dedicated function, say:

```
bool isPrime(int number){
    // return true or false ...
}
```

Sample Input:

```
11
```

Sample Output:

```
17
```

Sample Input:

```
100
```

Sample Output:

```
1060
```

Problem 24. Goldbach's Conjecture says that every even integer greater than 2 can be expressed as the sum of two prime numbers. Mathematically, it can be stated as: For any even integer $a > 2$, there exist prime numbers p_1 and p_2 such that $a = p_1 + p_2$. Write a C++ program that inputs an even number greater than 2, finds two prime numbers (p_1 and p_2) whose sum equals the input number according to Goldbach's Conjecture, and outputs the product of these two prime numbers (i.e., $p_1 \times p_2$). The program should ensure that p_1 is the smallest possible prime number that satisfies the conjecture for the given even number. For example, if the input of your code is 10, the output must be 21, because $10 = 3 + 7$, and $3 \times 7 = 21$. Note that although we can write $10 = 5 + 5$, 25 is not an acceptable answer because 5 is not the smallest possible prime number that can be used to sum up to 10. In this case, 3 is the smallest prime number that, when added to another prime number (7), equals 10. Therefore, the correct output is the product of 3 and 7, which is 21.

Specific policies:

- You must define at least one function, except `main()`, in your code to handle this problem.
- You must declare the functions with a **prototype** at the beginning, before any function defining or calls, as discussed in class.

Advice: Use the `isPrime` function you defined in Problem 23.

Sample Input:

```
24
```

Sample Output:

```
95
```

Sample Input:

```
10
```

Sample Output:

```
21
```

Sample Input:

```
1000
```

Sample Output:

```
2991
```

Sample Input:

```
4
```

Sample Output:

```
4
```

Problem 25. Write a C++ program that takes a positive integer greater than 1 as input, decomposes it into its prime factors, and outputs the sum of these prime factors. For example, let us explain the prime factorization and the sum of these prime factors for the numbers 7 and 18.
Prime factorization of 7:

- 7 is a prime number itself. Prime numbers are numbers greater than 1 that have no divisors other than 1 and themselves.
- Therefore, the prime factorization of 7 is just 7.
- The sum of the prime factors of 7 is simply 7 (since there is only one prime factor).

Prime factorization of 18:

- To factorize 18, we start with the smallest prime number, which is 2.
- 18 is divisible by 2, giving us $18 \div 2 = 9$.
- Now we consider the quotient, 9. The next smallest prime number is 3.
- 9 is divisible by 3, giving us $9 \div 3 = 3$.
- The quotient is again 3, which is a prime number itself.
- Therefore, the prime factorization of 18 is $2 \times 3 \times 3$.
- The sum of these prime factors is $2 + 3 + 3 = 8$.

Specific policy:

- You must define at least one function, except `main()`, in your code to handle this problem.
- You are **not** allowed to use recursive functions, as this concept is beyond this course.

Advice: Use the `isPrime` function you defined in Problem 23.

Sample Input:

```
11
```

Sample Output:

```
11
```

Sample Input:

```
18
```

Sample Output:

```
8
```

Sample Input:

```
2024
```

Sample Output:

```
40
```

Sample Input:

```
859
```

Sample Output:

```
859
```

Sample Input:

```
493
```

Sample Output:

```
46
```

Problem 26. In this problem, you are required to simulate a scenario involving a number of soldiers standing in a row. Some soldiers are facing to the right, while others are facing to the left. When commanded by their leader, every pair of soldiers standing face-to-face will simultaneously turn around, ending up back-to-back. This process will be repeated until no soldiers are standing face-to-face. Write a C++ code that reads input as a string and outputs the number of iterations required to reach the state where no soldiers are facing each other. The input string provides the initial direction of each soldier, using the letters 'L' (for left) and 'R' (for right). Your program should calculate and output the number of turns required to reach the stable state. For example, if the initial stage is RLLLL, we have

Input RLLLL
move #1: LRLLL
move #2: LLRLL
move #3: LLLRL
move #4: LLLLR

As another example, if the initial stage is LRLLLRLL, we have

Input LRLLLRLL
move #1: LLRLLLRL
move #2: LLLRLLLR
move #3: LLLLRLLR
move #4: LLLLLRLR
move #5: LLLLLLRR

Specific policies:

- You must define at least one function, except `main()`, in your code to handle this problem.
- You must declare the functions with a **prototype** at the beginning, before any function defining or calls, as discussed in class.
- You must use **pass-by-reference** for at least one of your functions.
- You are **not** allowed to use recursive functions, as this concept is beyond this course.

Advice: There are two suggested strategies to handle this problem. The first is to define a function that modifies the string of soldiers after each move and checks for any changes, such as

```
bool move(string &soldiers){
    // return true or false (if the string is modified return true,
    // otherwise false.)
}
```

The second strategy involves splitting these tasks into different functions; one function solely modifies the string, while another function takes two strings to check for any changes, such as

```
void move(string &soldiers){
    // ...
    // return ;
}
```

```
bool check(string before_move, string after_move){
    // return true if before_move and after_move are the same; otherwise,
    // return false)
}
```

Note that these are merely suggestions, and you are free to define your functions as long as they satisfy the above specific policies.

Sample Input:

```
RLLLL
```

Sample Output:

```
4
```

Sample Input:

```
LLL
```

Sample Output:

```
0
```

Sample Input:

```
RRRRRRRRRRRRR
```

Sample Output:

```
0
```

Sample Input:

```
LLLLRR
```

Sample Output:

```
0
```

Sample Input:

```
LLLRL
```

Sample Output:

```
1
```

Sample Input:

```
LRLLLRLL
```

Sample Output:

```
5
```

Sample Input:

```
RRLLRRRRLLLLLLRL
```

Sample Output:

```
12
```

Sample Input:

```
RLRLRLRLRL
```

Sample Output:

```
5
```

Sample Input:

```
RRLLRRLLLL
```

Sample Output:

```
7
```

Sample Input:

```
RRRRRLLLLL
```

Sample Output:

```
9
```

Sample Input:

```
RLLRLLLRLLLLRLLLLL
```

Sample Output:

```
14
```

Problem 27. Multiplicative persistence is the number of times you must multiply the digits of a number together until you reach a single digit. For example, consider the number 39:

step#1: $3 \times 9 = 27$
step#2: $2 \times 7 = 14$
step#3: $1 \times 4 = 4$

The multiplicative persistence of 39 is 3 because it took three steps to reduce it to a single-digit number. Write a C++ program that takes a positive integer number and returns its multiplicative persistence.

Note: When working exercises that involve potentially large numbers or calculations, please ensure you use the `long long` data type instead of `int` for storing and manipulating these numbers. The `int` data type might not be able to hold very large values, and you could encounter unexpected results or overflows. The `long long` data type provides a much larger range, ensuring accuracy and stability in your calculations. For this problem make sure to use `long long` to implement your code.

Specific policies:

- You must define at least one function, except `main()`, in your code to handle this problem.
- You must declare the functions with a **prototype** at the beginning, before any function defining or calls, as discussed in class.
- You must define **default values** for the function parameters (at least for one input parameter).
- You are **not** permitted to convert strings to integers or vice versa, as this concept has not been covered in the class yet.

Advice: You could create a function that calculates the product of the digits of an integer, such as:

```
long long multiplyDigits(long long number){
    // ...
}
```

Sample Input:

```
39
```

Sample Output:

```
3
```

Sample Input:

```
7
```

Sample Output:

```
0
```

Sample Input:

```
11112211
```

Sample Output:

```
1
```

Sample Input:

```
2023
```

Sample Output:

```
1
```

Sample Input:

```
277777788888899
```

Sample Output:

```
11
```

Problem 28. Write a C++ program that computes the summation of the Armstrong numbers strictly smaller than any positive integer provided by the user. An Armstrong number of a given number of digits is an integer such that the sum of its own digits raised to the power of the number of digits is equal to the number itself. For instance, 153 is an Armstrong number because:

$$1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$$

For example, if the input is 400, then the output must be 939 because Armstrong numbers smaller than 400 are 1, 2, 3, 4, 5, 6, 7, 8, 9, 153, 370, and 371. Therefore, the sum is $1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 153 + 370 + 371 = 939$.

Note: When working exercises that involve potentially large numbers or calculations, please ensure you use the `long long` data type instead of `int` for storing and manipulating these numbers. The `int` data type might not be able to hold very large values, and you could encounter unexpected results or overflows. The `long long` data type provides a much larger range, ensuring accuracy and stability in your calculations. For this problem make sure to use `long long` to implement your code.

Specific policies:

- You must define at least one function, except `main()`, in your code to handle this problem.
- You must declare the functions with a **prototype** at the beginning, before any function defining or calls, as discussed in class.
- You must define **default values** for the function parameters (at least for one input parameter).
- You are **not** permitted to convert strings to integers or vice versa, as this concept has not been covered in the class yet.

Advice: Create a power function that computes the power of a base number raised to an exponent, such as

```
long long myPower(int digit, int n){
    // return digit to the power of n
}
```

Similarly, a function can assist you by counting the number of digits in the number you are evaluating, for example:

```
int countDigits(long long number){
    // ...
}
```

Additionally, you might consider the task of identifying whether a number is Armstrong. Instead of repeating the checking logic every time you need it, you could encapsulate this logic within a dedicated function, such as:

```
bool isArmstrong(long long number){
    // return true or false ...
}
```

Sample Input:

```
400
```

Sample Output:

```
939
```

Sample Input:

```
4
```

Sample Output:

```
6
```

Sample Input:

```
60000
```

Sample Output:

```
75410
```

Sample Input:

```
25000000
```

Sample Output:

```
75845831
```

Problem 29. Write a C++ code that takes a single, space-less string (with at least three characters) as input and determines the sub-string with the highest frequency within it, ensuring that the minimum acceptable length for a sub-string is 2 characters. If there are multiple sub-strings with this highest frequency, the code should output the longest one. Further, if these highest frequency sub-strings are of equal length, your program should opt for the one that appears first in the main string, from left to right.

Sample Input:

```
abcdabefabgh
```

Sample Output:

```
ab
```

Sample Input:

```
Apbcd!cDbcd!LApbcd!!!BcD
```

Sample Output:

```
bcd!
```

Sample Input:

```
UCLA
```

Sample Output:

```
UCLA
```

Sample Input:

```
ArchCircleArchCircleArchCircleArch
```

Sample Output:

```
rc
```

Sample Input:

```
#$%ABBA$%ABBA###$%ABBA()()$%ABBA*$%ABBA&$%ABBA
```

Sample Output:

```
$%ABBA
```

Sample Input:

```
WEPointCloudWEPointCloudPointCloudPointCloudWEPointCloudWEWEPointCloud
```

Sample Output:

```
PointCloud
```

Sample Input:

```
rrrr
```

Sample Output:

```
rr
```

Problem 30. In chess, a queen can move any number of squares vertically, horizontally, or diagonally. Write a C++ program that takes an 8×8 chessboard configuration with 8 queens and checks whether any of the queens can attack each other. If any of the queens can attack another, the program should print `yes`. If none of the queens can attack any other, it should print `no`. The input will be a two-dimensional `array` (8×8) representing the chessboard. Each element of the list will be either `*` or `Q`, where `*` represents an empty square and `Q` represents a square occupied by a queen. We further assume there will always be exactly 8 queens on the board. The **starter code** is given to you as `main.cpp`. Accordingly, you **must** use the following format to get the board from the user:

```
    string board[8][8] = {{"n","n","n","n","n","n","n","n"},
                          {"n","n","n","n","n","n","n","n"},
                          {"n","n","n","n","n","n","n","n"},
                          {"n","n","n","n","n","n","n","n"},
                          {"n","n","n","n","n","n","n","n"},
                          {"n","n","n","n","n","n","n","n"},
                          {"n","n","n","n","n","n","n","n"},
                          {"n","n","n","n","n","n","n","n"}};

    string line;
    for (int i = 0; i < 8; i++){
        cin >> line;
        for (int j = 0; j < 8; j++){
            board[i][j] = line.at(j);
        }
    }
```

Note that with this format, each input the user enters represents a row of the board as a spaceless string, such as `***Q****`. This process creates the chessboard as a two-dimensional array for you. For instance, after reading the input, you can print your board using the following piece of code.

```
for (int i = 0; i < 8; i++){
    for (int j = 0; j < 8; j++){
        cout << board[i][j];
    }
    cout << endl;
}
```

Sample Input:

```
***Q****
******Q*
****Q***
**Q*****
Q*******
*****Q**
*******Q
*Q******
```

Sample Output:

```
no
```

Sample Input:

```
***Q****
******Q*
****Q***
*Q******
Q*******
*****Q**
*******Q
***Q****
```

Sample Output:

```
yes
```

Sample Input:

```
Q*******
*****Q**
***Q****
*Q******
******Q*
****Q***
**Q*****
*******Q
```

Sample Output:

```
yes
```

Sample Input:

```
Q*******
******Q*
****Q***
*******Q
*Q******
***Q****
*****Q**
**Q*****
```

Sample Output:

```
no
```

Problem 31. A palindrome is a sequence of characters that reads the same forwards as it does backward. When this concept is applied to numbers, a palindrome number is an integer that remains the same when its digits are reversed. In other words, a number is palindromic if it is the same when read from left to right and from right to left. Write a C++ program that takes a string of numbers as input and outputs the "minimum number of characters" that should be added to the end of the string to make it a palindrome. For example, for the input string of '123', the output of your code must be 2, because by adding '21' to the end of '123', we will have '12321', which is a palindrome number.

Note: The input to your code must be a `string` that contains only the digits 0 through 9. The input string can also start with a 0.

Advice: While it is technically possible to write your entire C++ program in the `main()` function without ever using any additional functions, this is not a recommended approach. You can break the final tasks into smaller tasks. For instance, you can create a function that checks if a string is a palindrome or not, such as

```
bool isPalindrome(string number){
    // return true or false ...
}
```

Sample Input:

```
123
```

Sample Output:

```
2
```

Sample Input:

```
9
```

Sample Output:

```
0
```

Sample Input:

```
01
```

Sample Output:

```
1
```

Sample Input:

```
122
```

Sample Output:

```
1
```

Sample Input:

```
121
```

Sample Output:

```
0
```

Sample Input:

```
999991
```

Sample Output:

```
5
```

Sample Input:

```
112358132523
```

Sample Output:

```
7
```

Sample Input:

```
112344
```

Sample Output:

```
4
```

Problem 32. In chess, the knight moves in an L-shape: either two squares in a horizontal direction and then one square vertically, or two squares in a vertical direction and then one square horizontally. This unique movement allows the knight to 'jump over' other pieces. Write a C++ program that takes the configuration of a chessboard with only knights and checks whether any of the knights can attack each other in a single move. If any knight can attack another, the program should print `yes`. If no knight can attack another, it should print `no`. The input will be a two-dimensional `array` (4×4) representing the chessboard. Each element of the list will be either `*` or `K`, where `*` represents an empty square and `K` represents a square occupied by a knight. We further assume the number of knights on the board can vary from 0 to 16. You **must** use the following format to get the board from the user:

```
    string board[4][4] = {{"n","n","n","n"},
                          {"n","n","n","n"},
                          {"n","n","n","n"},
                          {"n","n","n","n"}};

    string line;
    for (int i = 0; i < 4; i++){
        cin >> line;
        for (int j = 0; j < 4; j++){
            board[i][j] = line.at(j);
        }
    }
```

Note that with this format, each input the user enters represents a row of the board as a space-less string, such as `**K*`. This process creates the chessboard as a two-dimensional array for you. For instance, after reading the input, you can print your board using the following piece of code.

```
for (int i = 0; i < 4; i++){
    for (int j = 0; j < 4; j++){
        cout << board[i][j];
    }
    cout << endl;
}
```

Sample Input:

```
****
**K*
****
K***
```

Sample Output:

```
no
```

Sample Input:

```
****
****
KKKK
****
```

Sample Output:

```
no
```

Sample Input:

```
****
****
****
****
```

Sample Output:

```
no
```

Sample Input:

```
***K
*K**
****
K***
```

Sample Output:

```
yes
```

Sample Input:

```
*K**
*K**
K*K*
*K**
```

Sample Output:

```
yes
```

Problem 33. Consider an operation, denoted as the $\mathcal{D}$ operator, which processes a string containing a sequence of mathematical terms. Each term in the string consists of a numerical coefficient and a variable `y`, which may be repeated to represent its multiplicity. For example, a term traditionally expressed as $2y^3$ is represented in the string as `2*y*y*y`. The $\mathcal{D}$ operator modifies each term in the string according to the following rules:

- Multiplying the number of `y` in the term by the coefficient and then decreasing the count of `y`s by one.
- Eliminating the term if it does not contain `y`.

For example, a term like `2*y*y*y` is transformed to `6*y*y`, and `-4*y*y*y` becomes `-12*y*y`. A term like `-5`, which does not include `y`, is removed. Write a C++ code that receives a string input formatted in this manner, and applies the $\mathcal{D}$ operator to it, and outputs the resulting string. Note that the input string format may include several terms; however, we assume that each term has a distinct number of `y`s. For instance, a string like `2*y*y-5*y+y*y` cannot be an input for the problem because it contains two terms with the same number of `y`s, specifically `2*y*y` and `y*y`. Additionally, the string does not contain spaces between terms. For example, let us consider the following string as input:

```
2*y*y*y-3*y*y+y-5
```

When the $\mathcal{D}$ operator is applied to this string, the following transformations occur:

- `2*y*y*y` is transformed to `6*y*y`.
- `-3*y*y` changes to `-6*y`.
- `y` (implicitly `1*y`) becomes `1`.
- `-5`, as it does not contain `y`, is excluded.

Hence, the output of your code must be `6*y*y-6*y+1`.
Advice: To handle this problem, one idea is to split each string into sub-terms and define a function such as:

```
string operatorD(string term){
    // return ...
}
```

Then, concatenate the resulting terms. Note that at some point, you will need to convert strings to integers and vice versa.

Sample Input:

```
-5*y*y*y
```

Sample Output:

```
-15*y*y
```

Sample Input:

```
2*y*y*y*y*y*y*y*y+4*y-5
```

Sample Output:

```
14*y*y*y*y*y*y*y+4
```

Sample Input:

```
y*y*y*y*y-123*y*y
```

Sample Output:

```
5*y*y*y*y-246*y
```

Sample Input:

```
2024
```

Sample Output:

Problem 34. Please understand that this problem does not have a unique solution. Be creative in your approach. Moreover, note that there is no input for your program (i.e., do not use `cin` in your code). Submit the solution for this problem using the filename: `main.cpp`, `theclass.cpp`, and `theclass.h`. Take the following steps.

1. In `theclass.h`
   (a) Choose an object of interest.
   (b) Define a `class` representing your chosen object. The class name should reflect the object. Note that it does NOT need to be `theclass`.
   (c) Include at least three `private` member variables and one `private` member function.
   (d) Define a constructor, as a `public` member, with at least three parameters, one of which, at least, should have a default value.
   (e) Define at least three member functions, as `public` members. Include at least one of them with the `const` keyword.
   (f) Define a destructor for your `class`, as a `public` member.
   (g) **Only** declare function **prototypes** in this header file (i.e., `theclass.h`).
2. In `theclass.cpp`
   (a) Implement the bodies of the functions, constructor, and destructor you declared in the header file (i.e., `theclass.h`).
   (b) In at least one of these functions, use `this->` to access a class member.
3. In `main.cpp`
   (a) Create three instances of your class. Use the default constructor values for at least one of these instances.
   (b) Call at least one of the `public` member functions for each of the instances you have created in the `main` function.
   (c) Create a `vector` that holds objects of your `class` type.
   (d) Use the `push_back` method to add the three instances to this vector.
   (e) Write a function, `void printMyObjects`, that takes the vector as input in the format of **pass-by-reference to const** and prints a feature (e.g., a `private` member) of the objects (like `string name`). Ensure you have appropriate getter methods in your class to access `private` variables.
   (f) Call the `printMyObjects` function in the `main` function.
   (g) Create two more instances of your `class`.
   (h) Write a function, `void swapObjects`, which takes two objects of your `class` using **pass by-reference** and swaps them.

(i) In your `main()` function, call `swapObjects` and then print their `private` members (through appropriate getter methods) to confirm they have been swapped.

(j) Create a function that takes the defined object as input, using **pass-by-value**.

(k) Create a function that returns the defined object as its output.

(l) Call the functions in the two previous steps in the `main` function.

(m) Create another instance of your object and then define a pointer (e.g., `ptr` or any other name you like) that addresses this instance.

(n) Use the pointer to call a member function of your object, using `*` and `.` format (e.g., `(*ptr).`)

(o) Use the pointer to call a member function of your object, using `->` format (e.g., `ptr->`).

(p) You are encouraged to add any additional features or functions to your code that you think will enhance the functionality or demonstrate your understanding of the concepts of object-oriented programming in C++.

(q) As output, your program should briefly explain the object that you modeled using `class` through a string representation and `cout`

Sample Output:

```
My object represents a car ... (the rest of your explanation)
```

Problem 35. Please understand that this problem does not have a unique solution. Be creative in your approach. Moreover, note that there is no input for your program (i.e., do not use `cin` in your code). Take the following steps.

1. Implement a `struct` to model an object-oriented design. This `struct` will represent an object of your choice.
2. Define a `struct` for an object that interests you.
3. Include at least two features (variables) as `private` members of the `struct`.
4. Define a constructor as a `public` member of your `struct`. This constructor should appropriately initialize the object.
5. Define a destructor as a `public` member.
6. Implement at least two functions as `public` members, based on your choice. They could be getters or setters.
7. Add at least one variable member as a `public` member of your `struct`. This could be a status flag, a counter, or any other relevant data.
8. In the `main` function, create three instances of your `struct`.
9. Call at least one of the public member functions for each of the instances you have created in the `main` function.
10. Create a vector that holds objects of your `struct` type.
11. Use the `push_back` method to add the three instances to this vector.
12. Create a function that takes the defined object as an input
13. Create a function that returns the defined object as its output
14. Call the functions in the two previous steps in the `main` function.
15. You are encouraged to add any additional features or functions to your code that you think will enhance the functionality or demonstrate your understanding of the concepts of object-oriented programming in C++.
16. As output, your program should briefly explain the object that you modeled using `struct` through a string representation and `cout`

Sample Output:

```
My object represents a car ... (the rest of your explanation)
```

A starter code would be similar to the following

```
#include <iostream>
#include <string>
#include <vector>
#include <cmath>
using namespace std;

//Define a struct for an object that interests you.



//Create a function that takes the defined object as an input.



//Create a function that returns the defined object as its output




int main(){

    //Create three instances of your struct


    //Call at least one of the public member functions for each of the
    //instances you have created.



    //Create a vector that holds objects of your struct type.



    //Use the push_back method to add the three instances to this vector.


    //Step#14

    //Step#16
    cout << "My object represents ...";

    return 0 ;
}
```