Views

December 20, 2024

[1]:
```
!git clone https://github.com/avinanakarmi/DL_Project.git
```

```
Cloning into 'DL_Project'…
remote: Enumerating objects: 215, done.
remote: Counting objects: 100% (215/215), done.
remote: Compressing objects: 100% (139/139), done.
remote: Total 215 (delta 67), reused 210 (delta 64), pack-reused 0 (from 0)
Receiving objects: 100% (215/215), 2.36 MiB | 6.04 MiB/s, done.
Resolving deltas: 100% (67/67), done.
```

[3]:
```
!pip install datasets
```

```
Collecting datasets
  Downloading datasets-3.2.0-py3-none-any.whl.metadata (20 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from datasets) (3.16.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-
packages (from datasets) (1.26.4)
Requirement already satisfied: pyarrow>=15.0.0 in
/usr/local/lib/python3.10/dist-packages (from datasets) (17.0.0)
Collecting dill<0.3.9,>=0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages
(from datasets) (2.2.2)
Requirement already satisfied: requests>=2.32.2 in
/usr/local/lib/python3.10/dist-packages (from datasets) (2.32.3)
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.10/dist-
packages (from datasets) (4.67.1)
Collecting xxhash (from datasets)
  Downloading
xxhash-3.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(12 kB)
Collecting multiprocess<0.70.17 (from datasets)
  Downloading multiprocess-0.70.16-py310-none-any.whl.metadata (7.2 kB)
Collecting fsspec<=2024.9.0,>=2023.1.0 (from
fsspec[http]<=2024.9.0,>=2023.1.0->datasets)
  Downloading fsspec-2024.9.0-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-
```

```
packages (from datasets) (3.11.10)
Requirement already satisfied: huggingface-hub>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from datasets) (0.27.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from datasets) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-
packages (from datasets) (6.0.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (2.4.4)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.3.2)
Requirement already satisfied: async-timeout<6.0,>=4.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (4.0.3)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->datasets) (24.3.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (6.1.0)
Requirement already satisfied: propcache>=0.2.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (0.2.1)
Requirement already satisfied: yarl<2.0,>=1.17.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.18.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.23.0->datasets)
(4.12.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datasets)
(3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests>=2.32.2->datasets) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datasets)
(2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datasets)
(2024.12.14)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas->datasets) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-
packages (from pandas->datasets) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.2->pandas->datasets) (1.17.0)
Downloading datasets-3.2.0-py3-none-any.whl (480 kB)
                         480.6/480.6 kB
8.6 MB/s eta 0:00:00
```

```
Downloading dill-0.3.8-py3-none-any.whl (116 kB)
                           116.3/116.3 kB
8.7 MB/s eta 0:00:00
Downloading fsspec-2024.9.0-py3-none-any.whl (179 kB)
                           179.3/179.3 kB
14.0 MB/s eta 0:00:00
Downloading multiprocess-0.70.16-py310-none-any.whl (134 kB)
                           134.8/134.8 kB
10.3 MB/s eta 0:00:00
Downloading
xxhash-3.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
                           194.1/194.1 kB
14.2 MB/s eta 0:00:00
Installing collected packages: xxhash, fsspec, dill, multiprocess,
datasets
  Attempting uninstall: fsspec
    Found existing installation: fsspec 2024.10.0
    Uninstalling fsspec-2024.10.0:
      Successfully uninstalled fsspec-2024.10.0
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the following
dependency conflicts.
gcsfs 2024.10.0 requires fsspec==2024.10.0, but you have fsspec 2024.9.0 which
is incompatible.
Successfully installed datasets-3.2.0 dill-0.3.8 fsspec-2024.9.0
multiprocess-0.70.16 xxhash-3.5.0
```

[4]:
```
from datasets import load_dataset
```

[13]:
```
data = load_dataset("json", data_files=r"/content/DL_Project/response_folder/
↪data-concept_gen-i_r-bac22-0-20241218_193626.jsonl",
                    encoding="utf-8",
                    encoding_errors="ignore")
```

```
Generating train split: 0 examples [00:00, ? examples/s]
```

[15]:
```
data
```

[15]:
```
DatasetDict({
    train: Dataset({
        features: ['prompt', 'fingerprint', 'seed', 'id', 'concepts',
'instruction', 'response'],
        num_rows: 202
    })
})
```

[26]:
```
n = int(input("How many values to display: "))
for i in range(n):
  print("Prompts")
 ␣
  ↪print("##########################################################################################
  print(data['train']['prompt'][i])
  print("Concepts")
 ␣
  ↪print("##########################################################################################
  print(data['train']['concepts'][i])
 ␣
  ↪print("##########################################################################################
  print(data['train']['instruction'][i])
 ␣
  ↪print("##########################################################################################
  print(data['train']['response'][i])
```

```
How many values to display: 2
Prompts
##########################################################################################
######
You are an extremely intelligent AI coding assistant. Please provide an accurate
and reliable response to each user instruction. After delivering your response,
verify its consistency and correctness by writing a series of executable tests.

## Example 1
<instruction>
Create a C++ function to encode an integer into a sequence of encoded
characters, where each character represents a 5-bit chunk of the integer. The
ASCII value of each chunk character is formed by adding 63 to the 5-bit chunk's
value. For all chunks except the last one, set the 6th bit to 1 to indicate
continuation. Employ bitwise operations for this encoding process.
</instruction>

<response>
To encode an integer into a sequence of encoded characters based on the
specified criteria, let's follow these steps:

1. Break down the integer into 5-bit chunks: Extract 5-bit chunks from the
integer using `& 31` (`0x1F`), because the binary representation of `31`
(`11111`) has the last five bits set, which will effectively extract the 5 bits
we are interested in. We start from the least significant bits (LSB).
2. Set continuation bit: For all chunks except the last one, set the 6th bit to
1 using `| 32` (`0x20`) because the binary representation of `32` (`100000`) has
the 6th bit set to 1. This indicates that there's another continuation chunk
following.
3. Shift right: Right-shift the value by 5 to process the next 5 bits.
```

4. Convert to characters: Use addition of `63` to each chunk (now effectively 6 bits due to the continuation bit) to convert it into a printable ASCII character.
5. Assembly: Assemble these characters into the final encoded string while maintaining the order of the chunks through `std::reverse`.

Here's how you can implement this function:

```cpp
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

std::string encodeIntegerToChars(int value) {
    if (value == 0) {
        return std::string(1, static_cast<char>(63));
    }

    std::vector<char> encodedChars;

    while (value > 0) {
        // Extract the last 5 bits and add 63 to form the ASCII value
        char chunk = static_cast<char>((value & 31) + 63);
        value >>= 5; // Right shift by 5 bits

        // If there are more chunks, set the continuation bit (6th bit)
        if (value > 0) {
            chunk |= 32; // Set the 6th bit to 1
        }

        // Add the chunk to the vector
        encodedChars.push_back(chunk);
    }

    // Reverse the order of the characters to preserve the correct encoding order
    std::reverse(encodedChars.begin(), encodedChars.end());

    return std::string(encodedChars.begin(), encodedChars.end());
}
```

Keep in mind that this function is designed for non-negative integers. In C++, right-shifting (`>>`) a negative integer preserves the sign bit, which could lead to infinite loops or other incorrect behaviors if the same encoding approach is applied to negative integers without appropriate adjustments.
</response>

<tests>
You can validate this function with the following test cases that cover scenarios from zero to a number requiring more than one chunk, ensuring the function operates as expected:

```cpp
#include <cassert>
#include <string>

int main() {
    assert(encodeIntegerToChars(0) == "?");
    assert(encodeIntegerToChars(1) == "@");
    assert(encodeIntegerToChars(31) == "_");
    assert(encodeIntegerToChars(33).length() > 1); // Multi-chunk encoding
    return 0;
}
```
</tests>

Example 2
<instruction>
Create a C++ function to format a file size in bytes into a human-readable string representation, using 'bytes', 'KB' (kilobytes), 'MB' (megabytes), 'GB' (gigabytes), or 'TB' (terabytes) units. The output should be rounded to one decimal place and follow the format: "{X.Y}{Unit}", where "Unit" is the appropriate size unit and "X.Y" is the rounded size. For example, `format_size(1024 * 512)` should return `'512.0KB'`, and `format_size(1024**2 - 1)` should return `'1024.0KB'`.
</instruction>

<response>
To create such a function, we can use a loop to iterate through a predefined list of units (`['bytes', 'KB', 'MB', 'GB', 'TB']`) and check if the given number of bytes fits into the current unit by comparing it with 1024.0 and -1024.0 (for handling negative sizes). We'll keep dividing the number by 1024.0 until the correct unit is found where the number is less than 1024.0. Then, we'll format the result using the appropriate unit and the rounded size.

```cpp
#include <iostream>
#include <string>
#include <cmath>
#include <iomanip>

std::string format_size(std::size_t bytes) {
    const char* units[] = {"bytes", "KB", "MB", "GB", "TB"};
    const std::size_t unit_count = sizeof(units) / sizeof(units[0]);
```

```
    double size = static_cast<double>(bytes);
    std::size_t unit_index = 0;

    while (size >= 1024.0 && unit_index < unit_count - 1) {
        size /= 1024.0;
        ++unit_index;
    }

    std::ostringstream formatted_size;
    formatted_size << std::fixed << std::setprecision(1) << size <<
units[unit_index];
    return formatted_size.str();
}
```

This implementation handles both positive and negative file sizes and returns the formatted size as a string, even though negative sizes are not common in real-world scenarios.
</response>

<tests>
We can write tests that cover both positive and negative sizes across different units to validate the correctness of the `format_size` function:

```cpp
void test_format_size() {
    // Test cases
    assert(format_size(512) == "512.0bytes");
    assert(format_size(1024) == "1.0KB");
    assert(format_size(1024 * 512) == "512.0KB");
    assert(format_size(1024 * 1024 - 1) == "1024.0KB");
    assert(format_size(1024 * 1024) == "1.0MB");
    assert(format_size(1024ULL * 1024 * 1024) == "1.0GB");
    assert(format_size(1024ULL * 1024 * 1024 * 1024) == "1.0TB");
    assert(format_size(1024ULL * 1024 * 1024 * 1024 * 1024) == "1024.0TB");

    std::cout << "All tests passed!" << std::endl;
}

int main() {
    test_format_size();
    return 0;
}
```
</tests>

Example 3

<instruction>
Implement a C++ class with a method `find_nth_prime_memoized(n)` that returns the nth prime number, where n is a positive integer. Utilize memoization to store already found prime numbers in a list within the class to avoid recalculating primes for multiple method calls. Ensure the method efficiently updates and references this list to find new primes as needed. Additionally, enhance performance by avoiding unnecessary prime checks.
</instruction>

<response>
Implementing the required C++ class with the method find_nth_prime_memoized(n) involves several key steps:

Initialization:
The class constructor initializes a list prime_list to store memoized prime numbers. Initially, the list contains the first prime, 2. This serves as the starting point for all subsequent calculations.
Memoization and Caching:
The find_nth_prime_memoized method first checks if the nth prime is already in the prime_list. If it is, the method retrieves it directly. If the list is insufficient, the method calculates additional primes starting from the last number in the cache until the list contains at least n primes.
Efficient Prime Checking:
- A helper method is_prime is defined to determine if a number is prime:
        It tests divisibility only with primes from the prime_list.
        It limits checks to primes less than or equal to √candidate, implemented by   breaking the loop when the square of the prime exceeds the candidate.
- The method skips redundant checks for even numbers since the cache starts with 2 and subsequent numbers are incremented by 1 during the prime-finding process.
Error Handling:
The find_nth_prime_memoized method includes a check to ensure the input n is a positive integer. If n is invalid, it throws an exception.

Here is how you could implement such a class:

```cpp
#include <vector>
#include <cmath>

class PrimeFinder {
private:
    std::vector<int> prime_list; // List to store calculated primes

    // Helper method to check if a number is prime
    bool is_prime(int n) {
        if (n < 2) return false;
        for (int prime : prime_list) {
            if (prime * prime > n) break; // Stop if prime exceeds sqrt(n)
```

```
            if (n % prime == 0) return false; // Divisible means not prime
        }
        return true;
    }

public:
    // Constructor initializes with the first prime
    PrimeFinder() {
        prime_list.push_back(2); // Start with the first prime
    }

    // Method to find the nth prime using memoization
    int find_nth_prime_memoized(int n) {
        if (n <= 0) throw std::invalid_argument("n must be a positive integer");

        // Continue generating primes until the nth prime is found
        int current = prime_list.back() + 1; // Start checking from the next
number
        while (prime_list.size() < static_cast<size_t>(n)) {
            if (is_prime(current)) {
                prime_list.push_back(current); // Add to the list if prime
            }
            current++;
        }

        return prime_list[n - 1]; // Return the nth prime (0-indexed)
    }
};
```
</response>

<tests>
Here's how you can test this class by verifying that the class correctly finds
the first few primes and that subsequent calls return the memoized results
directly:

```cpp
#include <iostream>
#include <cassert>

// Test function
void test_find_nth_prime_memoized() {
    // Create an instance of the PrimeFinder class
    PrimeFinder primeFinder;

    // Test 1: Validate the first few prime numbers
    assert(primeFinder.find_nth_prime_memoized(1) == 2);  // 1st prime
    assert(primeFinder.find_nth_prime_memoized(2) == 3);  // 2nd prime


```

```
    assert(primeFinder.find_nth_prime_memoized(3) == 5);  // 3rd prime
    assert(primeFinder.find_nth_prime_memoized(4) == 7);  // 4th prime
    assert(primeFinder.find_nth_prime_memoized(5) == 11); // 5th prime

    // Test 2: Validate higher primes
    assert(primeFinder.find_nth_prime_memoized(10) == 29);  // 10th prime
    assert(primeFinder.find_nth_prime_memoized(20) == 71);  // 20th prime
    assert(primeFinder.find_nth_prime_memoized(50) == 229); // 50th prime

    // Test 3: Test efficiency with larger primes
    assert(primeFinder.find_nth_prime_memoized(100) == 541);  // 100th prime
    assert(primeFinder.find_nth_prime_memoized(200) == 1223); // 200th prime
    assert(primeFinder.find_nth_prime_memoized(500) == 3571); // 500th prime

    // Test 4: Validate memoization (repeated calls for the same prime)
    int nthPrime = primeFinder.find_nth_prime_memoized(10); // Cache the 10th
prime
    assert(primeFinder.find_nth_prime_memoized(10) == nthPrime); // Verify the
cached value

    // Test 5: Invalid input handling
    try {
        primeFinder.find_nth_prime_memoized(0); // Should throw an exception
        assert(false); // If no exception, fail the test
    } catch (const std::invalid_argument& e) {
        assert(std::string(e.what()) == "n must be a positive integer.");
    }

    try {
        primeFinder.find_nth_prime_memoized(-5); // Should throw an exception
        assert(false); // If no exception, fail the test
    } catch (const std::invalid_argument& e) {
        assert(std::string(e.what()) == "n must be a positive integer.");
    }

    std::cout << "All tests passed!" << std::endl;
}

int main() {
    test_find_nth_prime_memoized();
    return 0;
}
```
</tests>

Example 4
<instruction>
Design a C++ function to encode a list of strings into a unique list with the

same length, order, and meaning. The new list is formed by appending asterisks (*) to duplicate strings. For instance, given ['a', 'a', 'b', 'a', 'c'], the function should return ['a', 'a*', 'b', 'a**', 'c'], while for ['a', 'b', 'c'] it should return ['a', 'b', 'c'] without any changes. Write assertions to ensure the input is a list of strings.

Solve the problem in two lines of code, one for the precondition check and the other for the main logic.
</instruction>

<response>
You can achieve this functionality by first ensuring the input meets your requirements using an `assert` statement with list comprehension. Then, use another list comprehension to iterate through the input list and append `'*'` to the strings. The number of `'*'` is the number of occurrences of the string before the current index.

Here's how you can do it in two lines of C++ code:

```cpp
#include <iostream>
#include <vector>
#include <string>
#include <unordered_map>
#include <cassert>

std::vector<std::string> encode_strings(const std::vector<std::string>& input) {
    assert(std::all_of(input.begin(), input.end(), [](const std::string& s) {
return !s.empty(); })); // Precondition check

    std::unordered_map<std::string, int> counts;
    std::vector<std::string> result(input.size());

    for (size_t i = 0; i < input.size(); ++i) {
        const std::string& str = input[i];
        result[i] = str + std::string(counts[str]++, '*');
    }

    return result;
}
```

Note that although the function is concise, it is not the most efficient solution for large lists, as the `count` method has a time complexity of O(n) for each iteration.
</response>

<tests>

You can test the function with the provided examples to ensure it correctly encodes the strings as expected:

```cpp
#include <iostream>
#include <cassert>
int main() {
        // Test cases
    assert((encode_strings({"a", "a", "b", "a", "c"}) ==
std::vector<std::string>{"a", "a*", "b", "a**", "c"}));
    assert((encode_strings({"a", "b", "c"}) == std::vector<std::string>{"a",
"b", "c"}));
    assert((encode_strings({"apple", "apple", "banana"}) ==
std::vector<std::string>{"apple", "apple*", "banana"}));
    assert((encode_strings({"hello", "hello", "hello"}) ==
std::vector<std::string>{"hello", "hello*", "hello**"}));
    assert((encode_strings({}) == std::vector<std::string>{}));

    std::cout << "All tests passed!" << std::endl;
    return 0;
}

```
</tests>

Example 5
<instruction>
Develop a C++ function `ceil_divide_without_div` that computes the ceiling of `n / 2**b` without using division, floating-point operations, built-in functions like `math.ceil`, or converting it into computing the floor. Instead, use bit manipulation to achieve the result. You write assertions to ensure both `n` and `b` are non-negative integers.
</instruction>

<response>
We can leverage the properties of bitwise operations to develop a C++ function that computes the ceiling of n/2b without using division, floating-point operations, or any built-in functions like math.ceil. This implementation avoids converting the problem into a floor computation and achieves the desired result through careful adjustment of the value before right-shifting.

In this context, dividing by 2b is equivalent to right-shifting n by b bits in binary representation. However, right-shifting inherently computes the floor of the division. To achieve the ceiling effect, we add 2b - 1 to n before performing the shift. This adjustment ensures that any remainder pushes the result up to the next integer.

Key Steps in the Implementation:

Bitwise Shift for Division: Right-shifting n by b bits (n>>b) effectively divides n by 2b, rounding down to the nearest integer.
Adjust for Ceiling: Adding 2b-1 (1<<b-1) to n before the shift ensures that the division rounds up if there is any remainder.
Handling Special Cases: When b=0, 2b =1, and the division becomes trivial. The code handles this case explicitly to avoid unnecessary calculations.
Assertions: The function uses assert statements to enforce that both n and b are non-negative integers. This guards against invalid inputs and ensures correctness.

```cpp
#include <cassert>
int ceil_divide_without_div(int n, int b) {
    assert(n >= 0 && "n must be a non-negative integer");
    assert(b >= 0 && "b must be a non-negative integer");
    if (b == 0) {
        return n; // Dividing by 1 is trivial
    }
    int divisor = 1 << b; // Compute 2^b using left shift
    return (n + divisor - 1) >> b; // Adjust n for ceiling and right-shift
}
```

This function calculates the ceiling of `n / 2**b` purely through bitwise manipulation and integer arithmetic. The assertions ensure the input constraints are met, as specified.
</response>

<tests>
To verify the correctness of the function, we can compare the results of our function with the expected results obtained using `math.ceil` to ensure the correctness of the ceiling division operation:

```cpp
#include <iostream>

int main() {
    // Basic assertions to verify the function
    assert(ceil_divide_without_div(0, 1) == 0);
    assert(ceil_divide_without_div(1, 1) == 1);
    assert(ceil_divide_without_div(3, 1) == 2);
    assert(ceil_divide_without_div(7, 2) == 2);
    assert(ceil_divide_without_div(15, 3) == 2);
    assert(ceil_divide_without_div(9, 2) == 3);

    // Edge cases
    assert(ceil_divide_without_div(0, 0) == 0); // 0 / 1 = 0
    assert(ceil_divide_without_div(5, 0) == 5); // 5 / 1 = 5
```

```
    std::cout << "All test cases passed!\n";
    return 0;
}
```
</tests>

Example 6
<instruction>
Here are two special formulas:

$$
f_1(a, b) = (a + 1) \cdot (b + 1) - 1
$$

$$
f_2(k) = \begin{cases}
    \frac{(k + 1)^2}{2} + k + 1 & \text{if } k \text{ is odd} \\
    \frac{k^2}{2} + 2k + 1 & \text{if } k \text{ is even}
\end{cases}
$$

Write a C++ function to return $f_2(f_1(a, b))$ for given `a` and `b`.
</instruction>

<response>
Based on the formulas you provided, we can define two C++ functions, `f1(a, b)` and `f2(k)`, respectively, and then combine them to calculate $f2(f1(a, b))$ for given `a` and `b`.

Here is the implementation:

```cpp
class SpecialFormulas {
public:
    // Function f1(a, b)
    static int f1(int a, int b) {
        return (a + 1) * (b + 1) - 1;
    }

    // Function f2(k)
    static int f2(int k) {
        if (k % 2 == 1) { // k is odd
            return ((k + 1) * (k + 1)) / 2 + k + 1;
        } else { // k is even
            return (k * k) / 2 + 2 * k + 1;
        }
    }
```

```
    // Function to compute f2(f1(a, b))
    static int compute(int a, int b) {
        int f1Result = f1(a, b);
        return f2(f1Result);
    }
};
```

In this implementation:

- `f1(a, b)` calculates the first formula you've given.
- `f2(k)` calculates the second formula, with a conditional branch for whether `k` is even or odd.
- `f1_f2_combined(a, b)` uses the output of `f1(a, b)` as the input for `f2(k)` and returns the result.
</response>

<tests>
To test this combined function, you can simply compare it with the direct application of `f2` to the result of `f1` for different values of `a` and `b`:

```cpp
#include <iostream>
#include <cassert>
int main() {
    // Test cases
    assert(SpecialFormulas::compute(1, 1) == 14);  // f1(1, 1) = 5, f2(5) = 14
    assert(SpecialFormulas::compute(2, 3) == 98);  // f1(2, 3) = 19, f2(19) = 98
    assert(SpecialFormulas::compute(0, 0) == 3);   // f1(0, 0) = 0, f2(0) = 3
    assert(SpecialFormulas::compute(5, 7) == 969); // f1(5, 7) = 47, f2(47) =
969

    std::cout << "All test cases passed!" << std::endl;
    return 0;
}
```
</tests>

Example 7
<instruction>
Construct a C++ function `create_folds(data, count)` to create a list of folds from the given data, where each fold is a subsequence of the original data. The length of each fold should be approximately equal with at most a difference of 1. The function should satisfy the following assertions:

```cpp
assert create_folds(range(12), 2) == [range(0, 6), range(6, 12)]
```

```
assert create_folds(range(8), 2) == [range(0, 4), range(4, 8)]
assert create_folds(range(25), -5) == []
assert create_folds(range(6), 2) == [range(0, 3), range(3, 6)]
```
</instruction>

<response>
To create a C++ function that generates a list of folds from given data with each fold having approximately equal length (with at most a difference of 1 between any two folds), you can follow these steps:

1. Return an empty list immediately if `count` is non-positive since it's not possible to create a valid number of folds.
2. Divide the total length of the data by `count`, rounding down, to get the minimum number of items each fold should have.
3. Calculate the remainder to see how many extra items need to be distributed among the folds. They are distributed one by one to the first `remainder` folds.
4. Use a loop to create each fold, adjusting the starting and ending indices based on the calculations above to ensure each fold has the correct number of items.

Here's how you can implement this:

```cpp
#include <iostream>
#include <vector>
#include <cassert>
#include <cmath>

std::vector<std::vector<int>> create_folds(const std::vector<int>& data, int count) {
    if (count <= 0 || data.empty()) {
        return {}; // Return empty if count is non-positive or data is empty
    }

    std::vector<std::vector<int>> folds;
    size_t fold_size = data.size() / count;
    size_t remainder = data.size() % count;
    size_t start = 0;

    for (int i = 0; i < count; ++i) {
        size_t current_fold_size = fold_size + (i < remainder ? 1 : 0);
        folds.emplace_back(data.begin() + start, data.begin() + start + current_fold_size);
        start += current_fold_size;
    }

    return folds;


```

```
}
```

This implementation ensures that each fold has an equal length with at most a difference of 1 between any two folds. It handles edge cases, such as a non-positive `count`, by returning an empty list as specified.
</response>

<tests>
You can test your code with the provided assertions to verify that the function works correctly:

```cpp
void test_create_folds() {
    // Helper function to create range
    auto range = [](int start, int end) {
        std::vector<int> r;
        for (int i = start; i < end; ++i) r.push_back(i);
        return r;
    };

    // Test cases
    assert(create_folds(range(0, 12), 2) ==
std::vector<std::vector<int>>{range(0, 6), range(6, 12)});
    assert(create_folds(range(0, 8), 2) ==
std::vector<std::vector<int>>{range(0, 4), range(4, 8)});
    assert(create_folds(range(0, 25), -5) == std::vector<std::vector<int>>{});
    assert(create_folds(range(0, 6), 2) ==
std::vector<std::vector<int>>{range(0, 3), range(3, 6)});

    std::cout << "All tests passed!" << std::endl;
}

int main() {
    test_create_folds();
    return 0;
}
```
</tests>

Example 8
<instruction>
Design a C++ function that takes a sorted array and a target value, and return a valid index where target can be inserted to maintain the array's sorted order. Optimize the function to run in logarithmic time complexity.

For example, given `array = [1, 3, 5, 5, 6]` and `target = 5`, the function should return either 2 or 3 because 5 presents at both indices 2 and 3.

</instruction>

<response>
To solve this problem efficiently and ensure logarithmic time complexity, we can use a binary search algorithm. Compared with a standard binary search that looks for an exact match, we can modify the algorithm such that when the target is not found, we return the `left` bound, which represents the index where the target can be inserted to maintain the array's sorted order. Since any valid index for insertion is acceptable, we can direclty return the index if there is an exact match.

Here is a C++ function that implements this approach:

```cpp
int findInsertionIndex(const std::vector<int>& array, int target) {
    int left = 0;
    int right = array.size(); // `right` is set to `array.size()` for a possible insertion at the end.

    while (left < right) {
        int mid = left + (right - left) / 2; // To prevent overflow

        if (array[mid] < target) {
            left = mid + 1; // Move the left pointer up
        } else {
            right = mid; // Move the right pointer down
        }
    }

    return left; // The insertion index
}
```

This implementation ensures that when `target` is not found, `left` represents the correct insertion index. This is because at the end of the loop, all elements to the left of `left` are smaller than `target`, all elements to the right of `left` are equal to or larger than `target`, and `left > right`. Therefore, `left` is the correct index for insertion to maintain the sorted order.
</response>

<tests>
You can test the function with scenarios where the target is found at multiple indices and where the target is not present in the array:

```cpp
#include <cassert>
#include <vector>
```

```
int main() {
    // Test case 1: Target exists in the array
    std::vector<int> array1 = {1, 3, 5, 5, 6};
    int result1 = findInsertionIndex(array1, 5);
    assert(result1 == 2 || result1 == 3); // Should be either 2 or 3

    // Test case 2: Target is smaller than the smallest element
    std::vector<int> array2 = {1, 3, 5, 5, 6};
    int result2 = findInsertionIndex(array2, 0);
    assert(result2 == 0); // Should be inserted at index 0

    // Test case 3: Target is greater than the largest element
    std::vector<int> array3 = {1, 3, 5, 5, 6};
    int result3 = findInsertionIndex(array3, 7);
    assert(result3 == 5); // Should be inserted at index 5

    // Test case 4: Target is between elements
    std::vector<int> array4 = {1, 3, 5, 5, 6};
    int result4 = findInsertionIndex(array4, 2);
    assert(result4 == 1); // Should be inserted at index 1

    // Test case 5: Empty array
    std::vector<int> array5 = {};
    int result5 = findInsertionIndex(array5, 5);
    assert(result5 == 0); // Should be inserted at index 0

    // Test case 6: Single element array, target less than element
    std::vector<int> array6 = {3};
    int result6 = findInsertionIndex(array6, 2);
    assert(result6 == 0); // Should be inserted at index 0

    // Test case 7: Single element array, target greater than element
    std::vector<int> array7 = {3};
    int result7 = findInsertionIndex(array7, 4);
    assert(result7 == 1); // Should be inserted at index 1

    std::cout << "All test cases passed!" << std::endl;
    return 0;
}
```
</tests>

Example 9
<instruction>
Implement a C++ function to calculate the `n`th term in the Fibonacci sequence using dynamic programming with memoization. The function should be recursive and utilize memoization to store previously computed values to avoid repeated

computations.
</instruction>

<response>
Concepts
##
######
['dynamic programming', 'recursion', 'memoization']
##
######
Implement a C++ function to calculate the `n`th term in the Fibonacci sequence
using dynamic programming with memoization. The function should be recursive and
utilize memoization to store previously computed values to avoid repeated
computations.
##
######
To implement a C++ function that calculates the `n`th term in the Fibonacci
sequence using dynamic programming with memoization, we can use a recursive
function with memoization. The function should recursively call itself with `n -
1` and `n - 2` as arguments to calculate the previous two terms. It should store
previously computed values in a hash table or array to avoid repeated
computations.

Here is a C++ function that implements this approach:

```cpp
class FibonacciSequence {
public:
    static int nthTerm(int n, std::vector<int>& memo) {
        if (n <= 1) {
            return n;
        }

        if (memo[n] != -1) { // Check if the value is already computed
            return memo[n];
        }

        memo[n] = nthTerm(n - 1, memo) + nthTerm(n - 2, memo); // Store the
computed value
        return memo[n];
    }
};
```

This implementation ensures that previously computed values are stored in a hash
table or array to avoid repeated computations. This is because any previously
computed value for a specific `n` is stored at the `n`th index of the
memoization array or hash table. Therefore, when the same value is needed again,

it can be directly retrieved from the memoization array or hash table instead of being recomputed. This optimization ensures logarithmic time complexity.
</response>

<tests>
You can test the function with different values of `n` to verify that it works correctly:

```cpp
#include <cassert>
#include <vector>

int main() {
    std::vector<int> memo(100, -1); // Create a memoization array of size 100 with all values set to -1
    assert(FibonacciSequence::nthTerm(0, memo) == 0);
    assert(FibonacciSequence::nthTerm(1, memo) == 1);
    assert(FibonacciSequence::nthTerm(2, memo) == 1);
    assert(FibonacciSequence::nthTerm(3, memo) == 2);
    assert(FibonacciSequence::nthTerm(4, memo) == 3);
    assert(FibonacciSequence::nthTerm(5, memo) == 5);
    assert(FibonacciSequence::nthTerm(6, memo) == 8);
    assert(FibonacciSequence::nthTerm(7, memo) == 13);
    assert(FibonacciSequence::nthTerm(8, memo) == 21);
    assert(FibonacciSequence::nthTerm(9, memo) == 34);

    std::cout << "All test cases passed!" << std::endl;
    return 0;
}
```
Prompts
##
######
You are an extremely intelligent AI coding assistant. Please provide an accurate and reliable response to each user instruction. After delivering your response, verify its consistency and correctness by writing a series of executable tests.

Example 1
<instruction>
Here are two special formulas:

$$
f_1(a, b) = (a + 1) \cdot (b + 1) - 1
$$

$$
f_2(k) = \begin{cases}
    \frac{(k + 1)^2}{2} + k + 1 & \text{if } k \text{ is odd} \\

```
    \frac{k^2}{2} + 2k + 1 & \text{if } k \text{ is even}
\end{cases}
$$

Write a C++ function to return $f_2(f_1(a, b))$ for given `a` and `b`.
</instruction>

<response>
Based on the formulas you provided, we can define two C++ functions, `f1(a, b)`
and `f2(k)`, respectively, and then combine them to calculate $f2(f1(a, b))$ for
given `a` and `b`.

Here is the implementation:

```cpp
class SpecialFormulas {
public:
    // Function f1(a, b)
    static int f1(int a, int b) {
        return (a + 1) * (b + 1) - 1;
    }

    // Function f2(k)
    static int f2(int k) {
        if (k % 2 == 1) { // k is odd
            return ((k + 1) * (k + 1)) / 2 + k + 1;
        } else { // k is even
            return (k * k) / 2 + 2 * k + 1;
        }
    }

    // Function to compute f2(f1(a, b))
    static int compute(int a, int b) {
        int f1Result = f1(a, b);
        return f2(f1Result);
    }
};
```

In this implementation:

- `f1(a, b)` calculates the first formula you've given.
- `f2(k)` calculates the second formula, with a conditional branch for whether
`k` is even or odd.
- `f1_f2_combined(a, b)` uses the output of `f1(a, b)` as the input for `f2(k)`
and returns the result.
</response>
```

<tests>
To test this combined function, you can simply compare it with the direct application of `f2` to the result of `f1` for different values of `a` and `b`:

```cpp
#include <iostream>
#include <cassert>
int main() {
    // Test cases
    assert(SpecialFormulas::compute(1, 1) == 14);  // f1(1, 1) = 5, f2(5) = 14
    assert(SpecialFormulas::compute(2, 3) == 98);  // f1(2, 3) = 19, f2(19) = 98
    assert(SpecialFormulas::compute(0, 0) == 3);   // f1(0, 0) = 0, f2(0) = 3
    assert(SpecialFormulas::compute(5, 7) == 969); // f1(5, 7) = 47, f2(47) =
969

    std::cout << "All test cases passed!" << std::endl;
    return 0;
}
```
</tests>

Example 2
<instruction>
Create a C++ function to format a file size in bytes into a human-readable string representation, using 'bytes', 'KB' (kilobytes), 'MB' (megabytes), 'GB' (gigabytes), or 'TB' (terabytes) units. The output should be rounded to one decimal place and follow the format: "{X.Y}{Unit}", where "Unit" is the appropriate size unit and "X.Y" is the rounded size. For example, `format_size(1024 * 512)` should return `'512.0KB'`, and `format_size(1024**2 - 1)` should return `'1024.0KB'`.
</instruction>

<response>
To create such a function, we can use a loop to iterate through a predefined list of units (`['bytes', 'KB', 'MB', 'GB', 'TB']`) and check if the given number of bytes fits into the current unit by comparing it with 1024.0 and -1024.0 (for handling negative sizes). We'll keep dividing the number by 1024.0 until the correct unit is found where the number is less than 1024.0. Then, we'll format the result using the appropriate unit and the rounded size.

```cpp
#include <iostream>
#include <string>
#include <cmath>
#include <iomanip>

std::string format_size(std::size_t bytes) {
    const char* units[] = {"bytes", "KB", "MB", "GB", "TB"};
```

```
    const std::size_t unit_count = sizeof(units) / sizeof(units[0]);

    double size = static_cast<double>(bytes);
    std::size_t unit_index = 0;

    while (size >= 1024.0 && unit_index < unit_count - 1) {
        size /= 1024.0;
        ++unit_index;
    }

    std::ostringstream formatted_size;
    formatted_size << std::fixed << std::setprecision(1) << size <<
units[unit_index];
    return formatted_size.str();
}
```

This implementation handles both positive and negative file sizes and returns the formatted size as a string, even though negative sizes are not common in real-world scenarios.
</response>

<tests>
We can write tests that cover both positive and negative sizes across different units to validate the correctness of the `format_size` function:

```cpp
void test_format_size() {
    // Test cases
    assert(format_size(512) == "512.0bytes");
    assert(format_size(1024) == "1.0KB");
    assert(format_size(1024 * 512) == "512.0KB");
    assert(format_size(1024 * 1024 - 1) == "1024.0KB");
    assert(format_size(1024 * 1024) == "1.0MB");
    assert(format_size(1024ULL * 1024 * 1024) == "1.0GB");
    assert(format_size(1024ULL * 1024 * 1024 * 1024) == "1.0TB");
    assert(format_size(1024ULL * 1024 * 1024 * 1024 * 1024) == "1024.0TB");

    std::cout << "All tests passed!" << std::endl;
}

int main() {
    test_format_size();
    return 0;
}
```
</tests>

Example 3
<instruction>
Design a C++ function to encode a list of strings into a unique list with the same length, order, and meaning. The new list is formed by appending asterisks (*) to duplicate strings. For instance, given ['a', 'a', 'b', 'a', 'c'], the function should return ['a', 'a*', 'b', 'a**', 'c'], while for ['a', 'b', 'c'] it should return ['a', 'b', 'c'] without any changes. Write assertions to ensure the input is a list of strings.

Solve the problem in two lines of code, one for the precondition check and the other for the main logic.
</instruction>

<response>
You can achieve this functionality by first ensuring the input meets your requirements using an `assert` statement with list comprehension. Then, use another list comprehension to iterate through the input list and append `'*'` to the strings. The number of `'*'` is the number of occurrences of the string before the current index.

Here's how you can do it in two lines of C++ code:

```cpp
#include <iostream>
#include <vector>
#include <string>
#include <unordered_map>
#include <cassert>

std::vector<std::string> encode_strings(const std::vector<std::string>& input) {
    assert(std::all_of(input.begin(), input.end(), [](const std::string& s) {
return !s.empty(); })); // Precondition check

    std::unordered_map<std::string, int> counts;
    std::vector<std::string> result(input.size());

    for (size_t i = 0; i < input.size(); ++i) {
        const std::string& str = input[i];
        result[i] = str + std::string(counts[str]++, '*');
    }

    return result;
}
```

Note that although the function is concise, it is not the most efficient solution for large lists, as the `count` method has a time complexity of O(n) for each iteration.

</response>

<tests>
You can test the function with the provided examples to ensure it correctly encodes the strings as expected:

```cpp
#include <iostream>
#include <cassert>
int main() {
        // Test cases
    assert((encode_strings({"a", "a", "b", "a", "c"}) ==
std::vector<std::string>{"a", "a*", "b", "a**", "c"}));
    assert((encode_strings({"a", "b", "c"}) == std::vector<std::string>{"a",
"b", "c"}));
    assert((encode_strings({"apple", "apple", "banana"}) ==
std::vector<std::string>{"apple", "apple*", "banana"}));
    assert((encode_strings({"hello", "hello", "hello"}) ==
std::vector<std::string>{"hello", "hello*", "hello**"}));
    assert((encode_strings({}) == std::vector<std::string>{}));

    std::cout << "All tests passed!" << std::endl;
    return 0;
}

```
</tests>

Example 4
<instruction>
Develop a C++ function `ceil_divide_without_div` that computes the ceiling of `n / 2**b` without using division, floating-point operations, built-in functions like `math.ceil`, or converting it into computing the floor. Instead, use bit manipulation to achieve the result. You write assertions to ensure both `n` and `b` are non-negative integers.
</instruction>

<response>
We can leverage the properties of bitwise operations to develop a C++ function that computes the ceiling of n/2b without using division, floating-point operations, or any built-in functions like math.ceil. This implementation avoids converting the problem into a floor computation and achieves the desired result through careful adjustment of the value before right-shifting.

In this context, dividing by 2b is equivalent to right-shifting n by b bits in binary representation. However, right-shifting inherently computes the floor of the division. To achieve the ceiling effect, we add 2b - 1 to n before performing the shift. This adjustment ensures that any remainder pushes the

result up to the next integer.

Key Steps in the Implementation:
Bitwise Shift for Division: Right-shifting n by b bits (n>>b) effectively divides n by 2b, rounding down to the nearest integer.
Adjust for Ceiling: Adding 2b-1 (1<<b-1) to n before the shift ensures that the division rounds up if there is any remainder.
Handling Special Cases: When b=0, 2b =1, and the division becomes trivial. The code handles this case explicitly to avoid unnecessary calculations.
Assertions: The function uses assert statements to enforce that both n and b are non-negative integers. This guards against invalid inputs and ensures correctness.

```cpp
#include <cassert>
int ceil_divide_without_div(int n, int b) {
    assert(n >= 0 && "n must be a non-negative integer");
    assert(b >= 0 && "b must be a non-negative integer");
    if (b == 0) {
        return n; // Dividing by 1 is trivial
    }
    int divisor = 1 << b; // Compute 2^b using left shift
    return (n + divisor - 1) >> b; // Adjust n for ceiling and right-shift
}
```

This function calculates the ceiling of `n / 2**b` purely through bitwise manipulation and integer arithmetic. The assertions ensure the input constraints are met, as specified.
</response>

<tests>
To verify the correctness of the function, we can compare the results of our function with the expected results obtained using `math.ceil` to ensure the correctness of the ceiling division operation:

```cpp
#include <iostream>

int main() {
    // Basic assertions to verify the function
    assert(ceil_divide_without_div(0, 1) == 0);
    assert(ceil_divide_without_div(1, 1) == 1);
    assert(ceil_divide_without_div(3, 1) == 2);
    assert(ceil_divide_without_div(7, 2) == 2);
    assert(ceil_divide_without_div(15, 3) == 2);
    assert(ceil_divide_without_div(9, 2) == 3);
```

```
    // Edge cases
    assert(ceil_divide_without_div(0, 0) == 0); // 0 / 1 = 0
    assert(ceil_divide_without_div(5, 0) == 5); // 5 / 1 = 5

    std::cout << "All test cases passed!\n";
    return 0;
}
```
</tests>

Example 5
<instruction>
Construct a C++ function `create_folds(data, count)` to create a list of folds from the given data, where each fold is a subsequence of the original data. The length of each fold should be approximately equal with at most a difference of 1. The function should satisfy the following assertions:

```cpp
assert create_folds(range(12), 2) == [range(0, 6), range(6, 12)]
assert create_folds(range(8), 2) == [range(0, 4), range(4, 8)]
assert create_folds(range(25), -5) == []
assert create_folds(range(6), 2) == [range(0, 3), range(3, 6)]
```
</instruction>

<response>
To create a C++ function that generates a list of folds from given data with each fold having approximately equal length (with at most a difference of 1 between any two folds), you can follow these steps:

1. Return an empty list immediately if `count` is non-positive since it's not possible to create a valid number of folds.
2. Divide the total length of the data by `count`, rounding down, to get the minimum number of items each fold should have.
3. Calculate the remainder to see how many extra items need to be distributed among the folds. They are distributed one by one to the first `remainder` folds.
4. Use a loop to create each fold, adjusting the starting and ending indices based on the calculations above to ensure each fold has the correct number of items.

Here's how you can implement this:

```cpp
#include <iostream>
#include <vector>
#include <cassert>
#include <cmath>
```

```
std::vector<std::vector<int>> create_folds(const std::vector<int>& data, int
count) {
    if (count <= 0 || data.empty()) {
        return {}; // Return empty if count is non-positive or data is empty
    }

    std::vector<std::vector<int>> folds;
    size_t fold_size = data.size() / count;
    size_t remainder = data.size() % count;
    size_t start = 0;

    for (int i = 0; i < count; ++i) {
        size_t current_fold_size = fold_size + (i < remainder ? 1 : 0);
        folds.emplace_back(data.begin() + start, data.begin() + start +
current_fold_size);
        start += current_fold_size;
    }

    return folds;
}
```

This implementation ensures that each fold has an equal length with at most a difference of 1 between any two folds. It handles edge cases, such as a non-positive `count`, by returning an empty list as specified.
</response>

<tests>
You can test your code with the provided assertions to verify that the function works correctly:

```cpp
void test_create_folds() {
    // Helper function to create range
    auto range = [](int start, int end) {
        std::vector<int> r;
        for (int i = start; i < end; ++i) r.push_back(i);
        return r;
    };

    // Test cases
    assert(create_folds(range(0, 12), 2) ==
std::vector<std::vector<int>>{range(0, 6), range(6, 12)});
    assert(create_folds(range(0, 8), 2) ==
std::vector<std::vector<int>>{range(0, 4), range(4, 8)});
    assert(create_folds(range(0, 25), -5) == std::vector<std::vector<int>>{});
    assert(create_folds(range(0, 6), 2) ==
std::vector<std::vector<int>>{range(0, 3), range(3, 6)});
```

```
    std::cout << "All tests passed!" << std::endl;
}

int main() {
    test_create_folds();
    return 0;
}
```
</tests>

Example 6
<instruction>
Implement a C++ class with a method `find_nth_prime_memoized(n)` that returns the nth prime number, where n is a positive integer. Utilize memoization to store already found prime numbers in a list within the class to avoid recalculating primes for multiple method calls. Ensure the method efficiently updates and references this list to find new primes as needed. Additionally, enhance performance by avoiding unnecessary prime checks.
</instruction>

<response>
Implementing the required C++ class with the method find_nth_prime_memoized(n) involves several key steps:

Initialization:
The class constructor initializes a list prime_list to store memoized prime numbers. Initially, the list contains the first prime, 2. This serves as the starting point for all subsequent calculations.
Memoization and Caching:
The find_nth_prime_memoized method first checks if the nth prime is already in the prime_list. If it is, the method retrieves it directly. If the list is insufficient, the method calculates additional primes starting from the last number in the cache until the list contains at least n primes.
Efficient Prime Checking:
- A helper method is_prime is defined to determine if a number is prime:
        It tests divisibility only with primes from the prime_list.
        It limits checks to primes less than or equal to √candidate, implemented by   breaking the loop when the square of the prime exceeds the candidate.
- The method skips redundant checks for even numbers since the cache starts with 2 and subsequent numbers are incremented by 1 during the prime-finding process.
Error Handling:
The find_nth_prime_memoized method includes a check to ensure the input n is a positive integer. If n is invalid, it throws an exception.

Here is how you could implement such a class:

```cpp
```

```
#include <vector>
#include <cmath>

class PrimeFinder {
private:
    std::vector<int> prime_list; // List to store calculated primes

    // Helper method to check if a number is prime
    bool is_prime(int n) {
        if (n < 2) return false;
        for (int prime : prime_list) {
            if (prime * prime > n) break; // Stop if prime exceeds sqrt(n)
            if (n % prime == 0) return false; // Divisible means not prime
        }
        return true;
    }

public:
    // Constructor initializes with the first prime
    PrimeFinder() {
        prime_list.push_back(2); // Start with the first prime
    }

    // Method to find the nth prime using memoization
    int find_nth_prime_memoized(int n) {
        if (n <= 0) throw std::invalid_argument("n must be a positive integer");

        // Continue generating primes until the nth prime is found
        int current = prime_list.back() + 1; // Start checking from the next
number
        while (prime_list.size() < static_cast<size_t>(n)) {
            if (is_prime(current)) {
                prime_list.push_back(current); // Add to the list if prime
            }
            current++;
        }

        return prime_list[n - 1]; // Return the nth prime (0-indexed)
    }
};
```
</response>

<tests>
Here's how you can test this class by verifying that the class correctly finds
the first few primes and that subsequent calls return the memoized results
directly:

```cpp
#include <iostream>
#include <cassert>

// Test function
void test_find_nth_prime_memoized() {
    // Create an instance of the PrimeFinder class
    PrimeFinder primeFinder;

    // Test 1: Validate the first few prime numbers
    assert(primeFinder.find_nth_prime_memoized(1) == 2);  // 1st prime
    assert(primeFinder.find_nth_prime_memoized(2) == 3);  // 2nd prime
    assert(primeFinder.find_nth_prime_memoized(3) == 5);  // 3rd prime
    assert(primeFinder.find_nth_prime_memoized(4) == 7);  // 4th prime
    assert(primeFinder.find_nth_prime_memoized(5) == 11); // 5th prime

    // Test 2: Validate higher primes
    assert(primeFinder.find_nth_prime_memoized(10) == 29);  // 10th prime
    assert(primeFinder.find_nth_prime_memoized(20) == 71);  // 20th prime
    assert(primeFinder.find_nth_prime_memoized(50) == 229); // 50th prime

    // Test 3: Test efficiency with larger primes
    assert(primeFinder.find_nth_prime_memoized(100) == 541);  // 100th prime
    assert(primeFinder.find_nth_prime_memoized(200) == 1223); // 200th prime
    assert(primeFinder.find_nth_prime_memoized(500) == 3571); // 500th prime

    // Test 4: Validate memoization (repeated calls for the same prime)
    int nthPrime = primeFinder.find_nth_prime_memoized(10); // Cache the 10th
prime
    assert(primeFinder.find_nth_prime_memoized(10) == nthPrime); // Verify the
cached value

    // Test 5: Invalid input handling
    try {
        primeFinder.find_nth_prime_memoized(0); // Should throw an exception
        assert(false); // If no exception, fail the test
    } catch (const std::invalid_argument& e) {
        assert(std::string(e.what()) == "n must be a positive integer.");
    }

    try {
        primeFinder.find_nth_prime_memoized(-5); // Should throw an exception
        assert(false); // If no exception, fail the test
    } catch (const std::invalid_argument& e) {
        assert(std::string(e.what()) == "n must be a positive integer.");
    }

    std::cout << "All tests passed!" << std::endl;
```

```
}

int main() {
    test_find_nth_prime_memoized();
    return 0;
}
```
</tests>

Example 7
<instruction>
Design a C++ function that takes a sorted array and a target value, and return a valid index where target can be inserted to maintain the array's sorted order. Optimize the function to run in logarithmic time complexity.

For example, given `array = [1, 3, 5, 5, 6]` and `target = 5`, the function should return either 2 or 3 because 5 presents at both indices 2 and 3.
</instruction>

<response>
To solve this problem efficiently and ensure logarithmic time complexity, we can use a binary search algorithm. Compared with a standard binary search that looks for an exact match, we can modify the algorithm such that when the target is not found, we return the `left` bound, which represents the index where the target can be inserted to maintain the array's sorted order. Since any valid index for insertion is acceptable, we can direclty return the index if there is an exact match.

Here is a C++ function that implements this approach:

```cpp
int findInsertionIndex(const std::vector<int>& array, int target) {
    int left = 0;
    int right = array.size(); // `right` is set to `array.size()` for a possible insertion at the end.

    while (left < right) {
        int mid = left + (right - left) / 2; // To prevent overflow

        if (array[mid] < target) {
            left = mid + 1; // Move the left pointer up
        } else {
            right = mid; // Move the right pointer down
        }
    }

    return left; // The insertion index
}
```

```

This implementation ensures that when `target` is not found, `left` represents the correct insertion index. This is because at the end of the loop, all elements to the left of `left` are smaller than `target`, all elements to the right of `left` are equal to or larger than `target`, and `left > right`. Therefore, `left` is the correct index for insertion to maintain the sorted order.
</response>

<tests>
You can test the function with scenarios where the target is found at multiple indices and where the target is not present in the array:

```cpp
#include <cassert>
#include <vector>

int main() {
    // Test case 1: Target exists in the array
    std::vector<int> array1 = {1, 3, 5, 5, 6};
    int result1 = findInsertionIndex(array1, 5);
    assert(result1 == 2 || result1 == 3); // Should be either 2 or 3

    // Test case 2: Target is smaller than the smallest element
    std::vector<int> array2 = {1, 3, 5, 5, 6};
    int result2 = findInsertionIndex(array2, 0);
    assert(result2 == 0); // Should be inserted at index 0

    // Test case 3: Target is greater than the largest element
    std::vector<int> array3 = {1, 3, 5, 5, 6};
    int result3 = findInsertionIndex(array3, 7);
    assert(result3 == 5); // Should be inserted at index 5

    // Test case 4: Target is between elements
    std::vector<int> array4 = {1, 3, 5, 5, 6};
    int result4 = findInsertionIndex(array4, 2);
    assert(result4 == 1); // Should be inserted at index 1

    // Test case 5: Empty array
    std::vector<int> array5 = {};
    int result5 = findInsertionIndex(array5, 5);
    assert(result5 == 0); // Should be inserted at index 0

    // Test case 6: Single element array, target less than element
    std::vector<int> array6 = {3};
    int result6 = findInsertionIndex(array6, 2);
    assert(result6 == 0); // Should be inserted at index 0
```
```

```
    // Test case 7: Single element array, target greater than element
    std::vector<int> array7 = {3};
    int result7 = findInsertionIndex(array7, 4);
    assert(result7 == 1); // Should be inserted at index 1

    std::cout << "All test cases passed!" << std::endl;
    return 0;
}
```
</tests>

Example 8
<instruction>
Write a C++ function `huffman_decompress` that takes two arguments: `compressed_data`, a byte array of 0/1 sequence representing Huffman compressed data, and `huffman_tree`, a Huffman tree that stores the actual data (each leaf is associated with a character). The function should return the original uncompressed data as `bytes`. Define the `HuffmanNode` class yourself as a general tree structure with a `children` list.
</instruction>

<response>
The provided C++ code defines the HuffmanNode class as a general tree structure with a children list and an optional value attribute for leaf nodes. The huffman_decompress function processes the compressed_data using integer steps to traverse the Huffman tree, appending characters from leaf nodes to the decompressed output. It handles traversal efficiently and includes error checking for invalid steps.

You can test it with different Huffman trees and compressed data as needed.

Here's how we can implement this:

```cpp
#include <vector>
#include <iostream>
#include <string>
#include <unordered_map>
#include <memory>

// HuffmanNode class definition
class HuffmanNode {
public:
    char value; // Character value for leaf nodes
    std::shared_ptr<HuffmanNode> left; // Left child
    std::shared_ptr<HuffmanNode> right; // Right child
```

```
    // Constructor for leaf nodes
    HuffmanNode(char val) : value(val), left(nullptr), right(nullptr) {}

    // Constructor for internal nodes
    HuffmanNode(std::shared_ptr<HuffmanNode> l, std::shared_ptr<HuffmanNode> r)
: value('\0'), left(l), right(r) {}
};

// Function to decompress Huffman encoded data
std::string huffman_decompress(const std::vector<bool>& compressed_data, const
std::shared_ptr<HuffmanNode>& huffman_tree) {
    std::string decompressed_data;
    auto current_node = huffman_tree;

    for (bool bit : compressed_data) {
        if (bit) {
            current_node = current_node->right;
        } else {
            current_node = current_node->left;
        }

        // Check if we reached a leaf node
        if (!current_node->left && !current_node->right) {
            decompressed_data += current_node->value;
            current_node = huffman_tree; // Reset to root for the next symbol
        }
    }

    return decompressed_data;
}
```

In this implementation, each byte in `compressed_data` represents a bit, guiding the traversal of the Huffman tree. Characters are appended upon reaching leaf nodes and the result is then encoded to `bytes`.
</response>

<tests>
You can test this function with a sample Huffman tree and compressed data to verify that the decompression works correctly:

```cpp
void test_huffman_decompress() {
    // Test 1: Simple Huffman tree with "ab"
    auto root = std::make_shared<HuffmanNode>();
    auto left_child = std::make_shared<HuffmanNode>('a');
    auto right_child = std::make_shared<HuffmanNode>('b');
```

```
    root->children.push_back(left_child); // Step 0 -> 'a'
    root->children.push_back(right_child); // Step 1 -> 'b'

    std::vector<int> compressed_data1 = {0, 1};
    std::string result1 = huffman_decompress(compressed_data1, root);
    assert(result1 == "ab");

    // Test 2: More complex tree with "abc"
    auto mid_node = std::make_shared<HuffmanNode>();
    auto third_child = std::make_shared<HuffmanNode>('c');

    mid_node->children.push_back(left_child); // Step 0 -> 'a'
    mid_node->children.push_back(right_child); // Step 1 -> 'b'
    root->children.clear();
    root->children.push_back(mid_node); // Step 0 -> Mid-node
    root->children.push_back(third_child); // Step 1 -> 'c'

    std::vector<int> compressed_data2 = {0, 0, 1, 1};
    std::string result2 = huffman_decompress(compressed_data2, root);
    assert(result2 == "abc");

    // Test 3: Edge case with single leaf node
    auto single_node_tree = std::make_shared<HuffmanNode>('z');
    std::vector<int> compressed_data3 = {0, 0, 0};
    std::string result3 = huffman_decompress(compressed_data3,
single_node_tree);
    assert(result3 == "zzz");

    std::cout << "All tests passed!" << std::endl;
}

int main() {
    test_huffman_decompress();
    return 0;
}
```
</tests>

Example 9
<instruction>
Implement a C++ function that processes sensor readings and returns the detected values. The sensor readings are represented as a list of floating-point numbers. The function should process the readings and return a list of detected values. The detection algorithm is based on the following criteria:
1. The detected value should be greater than 100.
2. The detected value should be less than 200.
3. The detected value should be the maximum value in the readings.

The function should be designed to be efficient and accurate in detecting the values.
</instruction>

<response>
Concepts
##
######
['sensor reading and processing', 'software design for efficient and accurate detection']
##
######
Implement a C++ function that processes sensor readings and returns the detected values. The sensor readings are represented as a list of floating-point numbers. The function should process the readings and return a list of detected values. The detection algorithm is based on the following criteria:
1. The detected value should be greater than 100.
2. The detected value should be less than 200.
3. The detected value should be the maximum value in the readings.

The function should be designed to be efficient and accurate in detecting the values.
##
######
Implementing the provided C++ function involves several key steps:

1. The function processes the sensor readings and returns the detected values. It starts with an empty list of detected values. It then processes each reading individually using a loop.
2. If the reading is greater than 100 and less than 200, it is considered as a detected value and appended to the list.
3. At the end of the loop, the function returns the list of detected values.
4. The function includes a check to ensure the detected values are greater than 100 and less than 200. It also includes an error handling for invalid readings.

Here is a C++ function that implements this approach:

```cpp
class Sensor {
public:
    std::vector<float> readings; // Sensor readings

    Sensor(const std::vector<float>& readings) : readings(readings) {}

    // Function to process the sensor readings and return the detected values
    std::vector<float> process_readings() {
        std::vector<float> detected_values;
```

```
        for (float reading : readings) {
            if (reading > 100 && reading < 200) {
                detected_values.push_back(reading);
            }
        }

        return detected_values;
    }
};

// Function to process the sensor readings and return the detected values
std::vector<float> process_sensor_readings(const std::vector<float>& readings) {
    std::vector<float> detected_values;

    for (float reading : readings) {
        if (reading > 100 && reading < 200) {
            detected_values.push_back(reading);
        }
    }

    return detected_values;
}
```

This implementation ensures that the detected values are greater than 100 and
less than 200. It also includes error handling for invalid readings.
</response>

<tests>
You can test this function with different sensor readings and the detected
values as needed.

```cpp
void test_process_sensor_readings() {
    std::vector<float> readings1 = {150, 101, 50, 201, 100, 102};
    std::vector<float> detected_values1 = process_sensor_readings(readings1);
    assert(detected_values1.size() == 2);
    assert(detected_values1[0] == 101);
    assert(detected_values1[1] == 102);

    std::vector<float> readings2 = {101, 103, 105, 106, 107, 108};
    std::vector<float> detected_values2 = process_sensor_readings(readings2);
    assert(detected_values2.size() == 6);
    for (float detected_value : detected_values2) {
        assert(detected_value > 100 && detected_value < 200);
    }

    std::vector<float> readings3 = {50, 101, 102, 200, 300};
```

```
    std::vector<float> detected_values3 = process_sensor_readings(readings3);
    assert(detected_values3.empty());

    std::vector<float> readings4 = {};
    std::vector<float> detected_values4 = process_sensor_readings(readings4);
    assert(detected_values4.empty());

    std::cout << "All tests passed!" << std::endl;
}

int main() {
    test_process_sensor_readings();
    return 0;
}
```

[27]:
```
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
!pip install pypandoc
```

```
Reading package lists… Done
Building dependency tree… Done
Reading state information… Done
The following additional packages will be installed:
  dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-
texgyre
  fonts-urw-base35 libapache-pom-java libcmark-gfm-extensions0.29.0.gfm.3
libcmark-gfm0.29.0.gfm.3
  libcommons-logging-java libcommons-parent-java libfontbox-java libfontenc1
libgs9 libgs9-common
  libidn12 libijs-0.35 libjbig2dec0 libkpathsea6 libpdfbox-java libptexenc1
libruby3.0 libsynctex2
  libteckit0 libtexlua53 libtexluajit2 libwoff1 libzzip-0-13 lmodern pandoc-data
poppler-data
  preview-latex-style rake ruby ruby-net-telnet ruby-rubygems ruby-webrick ruby-
xmlrpc ruby3.0
  rubygems-integration t1utils teckit tex-common tex-gyre texlive-base texlive-
binaries
  texlive-fonts-recommended texlive-latex-base texlive-latex-recommended
texlive-pictures
  texlive-plain-generic tipa xfonts-encodings xfonts-utils
Suggested packages:
  fonts-noto fonts-freefont-otf | fonts-freefont-ttf libavalon-framework-java
  libcommons-logging-java-doc libexcalibur-logkit-java liblog4j1.2-java texlive-
luatex
  pandoc-citeproc context wkhtmltopdf librsvg2-bin groff ghc nodejs php python
libjs-mathjax
  libjs-katex citation-style-language-styles poppler-utils ghostscript fonts-
japanese-mincho
```

```
  | fonts-ipafont-mincho fonts-japanese-gothic | fonts-ipafont-gothic fonts-
arphic-ukai
  fonts-arphic-uming fonts-nanum ri ruby-dev bundler debhelper gv | postscript-
viewer perl-tk xpdf
  | pdf-viewer xzdec texlive-fonts-recommended-doc texlive-latex-base-doc
python3-pygments
  icc-profiles libfile-which-perl libspreadsheet-parseexcel-perl texlive-latex-
extra-doc
  texlive-latex-recommended-doc texlive-pstricks dot2tex prerex texlive-
pictures-doc vprerex
  default-jre-headless tipa-doc
The following NEW packages will be installed:
  dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-
texgyre
  fonts-urw-base35 libapache-pom-java libcmark-gfm-extensions0.29.0.gfm.3
libcmark-gfm0.29.0.gfm.3
  libcommons-logging-java libcommons-parent-java libfontbox-java libfontenc1
libgs9 libgs9-common
  libidn12 libijs-0.35 libjbig2dec0 libkpathsea6 libpdfbox-java libptexenc1
libruby3.0 libsynctex2
  libteckit0 libtexlua53 libtexluajit2 libwoff1 libzzip-0-13 lmodern pandoc
pandoc-data
  poppler-data preview-latex-style rake ruby ruby-net-telnet ruby-rubygems ruby-
webrick ruby-xmlrpc
  ruby3.0 rubygems-integration t1utils teckit tex-common tex-gyre texlive
texlive-base
  texlive-binaries texlive-fonts-recommended texlive-latex-base texlive-latex-
extra
  texlive-latex-recommended texlive-pictures texlive-plain-generic texlive-xetex
tipa
  xfonts-encodings xfonts-utils
0 upgraded, 59 newly installed, 0 to remove and 49 not upgraded.
Need to get 202 MB of archives.
After this operation, 728 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-droid-fallback all
1:6.0.1r16-1.1build1 [1,805 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-lato all 2.0-2.1
[2,696 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/main amd64 poppler-data all
0.4.11-1 [2,171 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tex-common all 6.17
[33.7 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-urw-base35 all
20200910-1 [6,367 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgs9-common
all 9.55.0~dfsg1-0ubuntu5.10 [752 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libidn12 amd64
1.38-4ubuntu1 [60.0 kB]
```

```
Get:8 http://archive.ubuntu.com/ubuntu jammy/main amd64 libijs-0.35 amd64
0.35-15build2 [16.5 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy/main amd64 libjbig2dec0 amd64
0.19-3build2 [64.7 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgs9 amd64
9.55.0~dfsg1-0ubuntu5.10 [5,031 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libkpathsea6
amd64 2021.20210626.59705-1ubuntu0.2 [60.4 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy/main amd64 libwoff1 amd64
1.0.2-1build4 [45.2 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy/universe amd64 dvisvgm amd64
2.13.1-1 [1,221 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy/universe amd64 fonts-lmodern all
2.004.5-6.1 [4,532 kB]
Get:15 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-noto-mono all
20201225-1build1 [397 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy/universe amd64 fonts-texgyre all
20180621-3.1 [10.2 MB]
Get:17 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libapache-pom-java
all 18-1 [4,720 B]
Get:18 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcmark-
gfm0.29.0.gfm.3 amd64 0.29.0.gfm.3-3 [115 kB]
Get:19 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcmark-gfm-
extensions0.29.0.gfm.3 amd64 0.29.0.gfm.3-3 [25.1 kB]
Get:20 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcommons-parent-
java all 43-1 [10.8 kB]
Get:21 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcommons-logging-
java all 1.2-2 [60.3 kB]
Get:22 http://archive.ubuntu.com/ubuntu jammy/main amd64 libfontenc1 amd64
1:1.1.4-1build3 [14.7 kB]
Get:23 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libptexenc1
amd64 2021.20210626.59705-1ubuntu0.2 [39.1 kB]
Get:24 http://archive.ubuntu.com/ubuntu jammy/main amd64 rubygems-integration
all 1.18 [5,336 B]
Get:25 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 ruby3.0 amd64
3.0.2-7ubuntu2.8 [50.1 kB]
Get:26 http://archive.ubuntu.com/ubuntu jammy/main amd64 ruby-rubygems all
3.3.5-2 [228 kB]
Get:27 http://archive.ubuntu.com/ubuntu jammy/main amd64 ruby amd64 1:3.0~exp1
[5,100 B]
Get:28 http://archive.ubuntu.com/ubuntu jammy/main amd64 rake all 13.0.6-2 [61.7
kB]
Get:29 http://archive.ubuntu.com/ubuntu jammy/main amd64 ruby-net-telnet all
0.1.1-2 [12.6 kB]
Get:30 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 ruby-webrick
all 1.7.0-3ubuntu0.1 [52.1 kB]
Get:31 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 ruby-xmlrpc all
0.3.2-1ubuntu0.1 [24.9 kB]
```

```
Get:32 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libruby3.0
amd64 3.0.2-7ubuntu2.8 [5,113 kB]
Get:33 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libsynctex2
amd64 2021.20210626.59705-1ubuntu0.2 [55.6 kB]
Get:34 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libteckit0 amd64
2.5.11+ds1-1 [421 kB]
Get:35 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libtexlua53
amd64 2021.20210626.59705-1ubuntu0.2 [120 kB]
Get:36 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libtexluajit2
amd64 2021.20210626.59705-1ubuntu0.2 [267 kB]
Get:37 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libzzip-0-13 amd64
0.13.72+dfsg.1-1.1 [27.0 kB]
Get:38 http://archive.ubuntu.com/ubuntu jammy/main amd64 xfonts-encodings all
1:1.0.5-0ubuntu2 [578 kB]
Get:39 http://archive.ubuntu.com/ubuntu jammy/main amd64 xfonts-utils amd64
1:7.7+6build2 [94.6 kB]
Get:40 http://archive.ubuntu.com/ubuntu jammy/universe amd64 lmodern all
2.004.5-6.1 [9,471 kB]
Get:41 http://archive.ubuntu.com/ubuntu jammy/universe amd64 pandoc-data all
2.9.2.1-3ubuntu2 [81.8 kB]
Get:42 http://archive.ubuntu.com/ubuntu jammy/universe amd64 pandoc amd64
2.9.2.1-3ubuntu2 [20.3 MB]
Get:43 http://archive.ubuntu.com/ubuntu jammy/universe amd64 preview-latex-style
all 12.2-1ubuntu1 [185 kB]
Get:44 http://archive.ubuntu.com/ubuntu jammy/main amd64 t1utils amd64
1.41-4build2 [61.3 kB]
Get:45 http://archive.ubuntu.com/ubuntu jammy/universe amd64 teckit amd64
2.5.11+ds1-1 [699 kB]
Get:46 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tex-gyre all
20180621-3.1 [6,209 kB]
Get:47 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 texlive-
binaries amd64 2021.20210626.59705-1ubuntu0.2 [9,860 kB]
Get:48 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-base all
2021.20220204-1 [21.0 MB]
Get:49 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-fonts-
recommended all 2021.20220204-1 [4,972 kB]
Get:50 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-base
all 2021.20220204-1 [1,128 kB]
Get:51 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-
recommended all 2021.20220204-1 [14.4 MB]
Get:52 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive all
2021.20220204-1 [14.3 kB]
Get:53 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libfontbox-java all
1:1.8.16-2 [207 kB]
Get:54 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libpdfbox-java all
1:1.8.16-2 [5,199 kB]
Get:55 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-pictures
all 2021.20220204-1 [8,720 kB]
```

```
Get:56 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-extra
all 2021.20220204-1 [13.9 MB]
Get:57 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-plain-
generic all 2021.20220204-1 [27.5 MB]
Get:58 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tipa all 2:1.3-21
[2,967 kB]
Get:59 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-xetex all
2021.20220204-1 [12.4 MB]
Fetched 202 MB in 13s (15.8 MB/s)
Extracting templates from packages: 100%
Preconfiguring packages …
Selecting previously unselected package fonts-droid-fallback.
(Reading database … 123634 files and directories currently installed.)
Preparing to unpack …/00-fonts-droid-fallback_1%3a6.0.1r16-1.1build1_all.deb
…
Unpacking fonts-droid-fallback (1:6.0.1r16-1.1build1) …
Selecting previously unselected package fonts-lato.
Preparing to unpack …/01-fonts-lato_2.0-2.1_all.deb …
Unpacking fonts-lato (2.0-2.1) …
Selecting previously unselected package poppler-data.
Preparing to unpack …/02-poppler-data_0.4.11-1_all.deb …
Unpacking poppler-data (0.4.11-1) …
Selecting previously unselected package tex-common.
Preparing to unpack …/03-tex-common_6.17_all.deb …
Unpacking tex-common (6.17) …
Selecting previously unselected package fonts-urw-base35.
Preparing to unpack …/04-fonts-urw-base35_20200910-1_all.deb …
Unpacking fonts-urw-base35 (20200910-1) …
Selecting previously unselected package libgs9-common.
Preparing to unpack …/05-libgs9-common_9.55.0~dfsg1-0ubuntu5.10_all.deb …
Unpacking libgs9-common (9.55.0~dfsg1-0ubuntu5.10) …
Selecting previously unselected package libidn12:amd64.
Preparing to unpack …/06-libidn12_1.38-4ubuntu1_amd64.deb …
Unpacking libidn12:amd64 (1.38-4ubuntu1) …
Selecting previously unselected package libijs-0.35:amd64.
Preparing to unpack …/07-libijs-0.35_0.35-15build2_amd64.deb …
Unpacking libijs-0.35:amd64 (0.35-15build2) …
Selecting previously unselected package libjbig2dec0:amd64.
Preparing to unpack …/08-libjbig2dec0_0.19-3build2_amd64.deb …
Unpacking libjbig2dec0:amd64 (0.19-3build2) …
Selecting previously unselected package libgs9:amd64.
Preparing to unpack …/09-libgs9_9.55.0~dfsg1-0ubuntu5.10_amd64.deb …
Unpacking libgs9:amd64 (9.55.0~dfsg1-0ubuntu5.10) …
Selecting previously unselected package libkpathsea6:amd64.
Preparing to unpack …/10-libkpathsea6_2021.20210626.59705-1ubuntu0.2_amd64.deb
…
Unpacking libkpathsea6:amd64 (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package libwoff1:amd64.
```

```
Preparing to unpack …/11-libwoff1_1.0.2-1build4_amd64.deb …
Unpacking libwoff1:amd64 (1.0.2-1build4) …
Selecting previously unselected package dvisvgm.
Preparing to unpack …/12-dvisvgm_2.13.1-1_amd64.deb …
Unpacking dvisvgm (2.13.1-1) …
Selecting previously unselected package fonts-lmodern.
Preparing to unpack …/13-fonts-lmodern_2.004.5-6.1_all.deb …
Unpacking fonts-lmodern (2.004.5-6.1) …
Selecting previously unselected package fonts-noto-mono.
Preparing to unpack …/14-fonts-noto-mono_20201225-1build1_all.deb …
Unpacking fonts-noto-mono (20201225-1build1) …
Selecting previously unselected package fonts-texgyre.
Preparing to unpack …/15-fonts-texgyre_20180621-3.1_all.deb …
Unpacking fonts-texgyre (20180621-3.1) …
Selecting previously unselected package libapache-pom-java.
Preparing to unpack …/16-libapache-pom-java_18-1_all.deb …
Unpacking libapache-pom-java (18-1) …
Selecting previously unselected package libcmark-gfm0.29.0.gfm.3:amd64.
Preparing to unpack …/17-libcmark-gfm0.29.0.gfm.3_0.29.0.gfm.3-3_amd64.deb …
Unpacking libcmark-gfm0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) …
Selecting previously unselected package libcmark-gfm-
extensions0.29.0.gfm.3:amd64.
Preparing to unpack …/18-libcmark-gfm-
extensions0.29.0.gfm.3_0.29.0.gfm.3-3_amd64.deb …
Unpacking libcmark-gfm-extensions0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) …
Selecting previously unselected package libcommons-parent-java.
Preparing to unpack …/19-libcommons-parent-java_43-1_all.deb …
Unpacking libcommons-parent-java (43-1) …
Selecting previously unselected package libcommons-logging-java.
Preparing to unpack …/20-libcommons-logging-java_1.2-2_all.deb …
Unpacking libcommons-logging-java (1.2-2) …
Selecting previously unselected package libfontenc1:amd64.
Preparing to unpack …/21-libfontenc1_1%3a1.1.4-1build3_amd64.deb …
Unpacking libfontenc1:amd64 (1:1.1.4-1build3) …
Selecting previously unselected package libptexenc1:amd64.
Preparing to unpack …/22-libptexenc1_2021.20210626.59705-1ubuntu0.2_amd64.deb
…
Unpacking libptexenc1:amd64 (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package rubygems-integration.
Preparing to unpack …/23-rubygems-integration_1.18_all.deb …
Unpacking rubygems-integration (1.18) …
Selecting previously unselected package ruby3.0.
Preparing to unpack …/24-ruby3.0_3.0.2-7ubuntu2.8_amd64.deb …
Unpacking ruby3.0 (3.0.2-7ubuntu2.8) …
Selecting previously unselected package ruby-rubygems.
Preparing to unpack …/25-ruby-rubygems_3.3.5-2_all.deb …
Unpacking ruby-rubygems (3.3.5-2) …
Selecting previously unselected package ruby.
```

```
Preparing to unpack …/26-ruby_1%3a3.0~exp1_amd64.deb …
Unpacking ruby (1:3.0~exp1) …
Selecting previously unselected package rake.
Preparing to unpack …/27-rake_13.0.6-2_all.deb …
Unpacking rake (13.0.6-2) …
Selecting previously unselected package ruby-net-telnet.
Preparing to unpack …/28-ruby-net-telnet_0.1.1-2_all.deb …
Unpacking ruby-net-telnet (0.1.1-2) …
Selecting previously unselected package ruby-webrick.
Preparing to unpack …/29-ruby-webrick_1.7.0-3ubuntu0.1_all.deb …
Unpacking ruby-webrick (1.7.0-3ubuntu0.1) …
Selecting previously unselected package ruby-xmlrpc.
Preparing to unpack …/30-ruby-xmlrpc_0.3.2-1ubuntu0.1_all.deb …
Unpacking ruby-xmlrpc (0.3.2-1ubuntu0.1) …
Selecting previously unselected package libruby3.0:amd64.
Preparing to unpack …/31-libruby3.0_3.0.2-7ubuntu2.8_amd64.deb …
Unpacking libruby3.0:amd64 (3.0.2-7ubuntu2.8) …
Selecting previously unselected package libsynctex2:amd64.
Preparing to unpack …/32-libsynctex2_2021.20210626.59705-1ubuntu0.2_amd64.deb
…
Unpacking libsynctex2:amd64 (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package libteckit0:amd64.
Preparing to unpack …/33-libteckit0_2.5.11+ds1-1_amd64.deb …
Unpacking libteckit0:amd64 (2.5.11+ds1-1) …
Selecting previously unselected package libtexlua53:amd64.
Preparing to unpack …/34-libtexlua53_2021.20210626.59705-1ubuntu0.2_amd64.deb
…
Unpacking libtexlua53:amd64 (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package libtexluajit2:amd64.
Preparing to unpack
…/35-libtexluajit2_2021.20210626.59705-1ubuntu0.2_amd64.deb …
Unpacking libtexluajit2:amd64 (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package libzzip-0-13:amd64.
Preparing to unpack …/36-libzzip-0-13_0.13.72+dfsg.1-1.1_amd64.deb …
Unpacking libzzip-0-13:amd64 (0.13.72+dfsg.1-1.1) …
Selecting previously unselected package xfonts-encodings.
Preparing to unpack …/37-xfonts-encodings_1%3a1.0.5-0ubuntu2_all.deb …
Unpacking xfonts-encodings (1:1.0.5-0ubuntu2) …
Selecting previously unselected package xfonts-utils.
Preparing to unpack …/38-xfonts-utils_1%3a7.7+6build2_amd64.deb …
Unpacking xfonts-utils (1:7.7+6build2) …
Selecting previously unselected package lmodern.
Preparing to unpack …/39-lmodern_2.004.5-6.1_all.deb …
Unpacking lmodern (2.004.5-6.1) …
Selecting previously unselected package pandoc-data.
Preparing to unpack …/40-pandoc-data_2.9.2.1-3ubuntu2_all.deb …
Unpacking pandoc-data (2.9.2.1-3ubuntu2) …
Selecting previously unselected package pandoc.
```

```
Preparing to unpack …/41-pandoc_2.9.2.1-3ubuntu2_amd64.deb …
Unpacking pandoc (2.9.2.1-3ubuntu2) …
Selecting previously unselected package preview-latex-style.
Preparing to unpack …/42-preview-latex-style_12.2-1ubuntu1_all.deb …
Unpacking preview-latex-style (12.2-1ubuntu1) …
Selecting previously unselected package t1utils.
Preparing to unpack …/43-t1utils_1.41-4build2_amd64.deb …
Unpacking t1utils (1.41-4build2) …
Selecting previously unselected package teckit.
Preparing to unpack …/44-teckit_2.5.11+ds1-1_amd64.deb …
Unpacking teckit (2.5.11+ds1-1) …
Selecting previously unselected package tex-gyre.
Preparing to unpack …/45-tex-gyre_20180621-3.1_all.deb …
Unpacking tex-gyre (20180621-3.1) …
Selecting previously unselected package texlive-binaries.
Preparing to unpack …/46-texlive-
binaries_2021.20210626.59705-1ubuntu0.2_amd64.deb …
Unpacking texlive-binaries (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package texlive-base.
Preparing to unpack …/47-texlive-base_2021.20220204-1_all.deb …
Unpacking texlive-base (2021.20220204-1) …
Selecting previously unselected package texlive-fonts-recommended.
Preparing to unpack …/48-texlive-fonts-recommended_2021.20220204-1_all.deb …
Unpacking texlive-fonts-recommended (2021.20220204-1) …
Selecting previously unselected package texlive-latex-base.
Preparing to unpack …/49-texlive-latex-base_2021.20220204-1_all.deb …
Unpacking texlive-latex-base (2021.20220204-1) …
Selecting previously unselected package texlive-latex-recommended.
Preparing to unpack …/50-texlive-latex-recommended_2021.20220204-1_all.deb …
Unpacking texlive-latex-recommended (2021.20220204-1) …
Selecting previously unselected package texlive.
Preparing to unpack …/51-texlive_2021.20220204-1_all.deb …
Unpacking texlive (2021.20220204-1) …
Selecting previously unselected package libfontbox-java.
Preparing to unpack …/52-libfontbox-java_1%3a1.8.16-2_all.deb …
Unpacking libfontbox-java (1:1.8.16-2) …
Selecting previously unselected package libpdfbox-java.
Preparing to unpack …/53-libpdfbox-java_1%3a1.8.16-2_all.deb …
Unpacking libpdfbox-java (1:1.8.16-2) …
Selecting previously unselected package texlive-pictures.
Preparing to unpack …/54-texlive-pictures_2021.20220204-1_all.deb …
Unpacking texlive-pictures (2021.20220204-1) …
Selecting previously unselected package texlive-latex-extra.
Preparing to unpack …/55-texlive-latex-extra_2021.20220204-1_all.deb …
Unpacking texlive-latex-extra (2021.20220204-1) …
Selecting previously unselected package texlive-plain-generic.
Preparing to unpack …/56-texlive-plain-generic_2021.20220204-1_all.deb …
Unpacking texlive-plain-generic (2021.20220204-1) …
```

```
Selecting previously unselected package tipa.
Preparing to unpack …/57-tipa_2%3a1.3-21_all.deb …
Unpacking tipa (2:1.3-21) …
Selecting previously unselected package texlive-xetex.
Preparing to unpack …/58-texlive-xetex_2021.20220204-1_all.deb …
Unpacking texlive-xetex (2021.20220204-1) …
Setting up fonts-lato (2.0-2.1) …
Setting up fonts-noto-mono (20201225-1build1) …
Setting up libwoff1:amd64 (1.0.2-1build4) …
Setting up libtexlua53:amd64 (2021.20210626.59705-1ubuntu0.2) …
Setting up libijs-0.35:amd64 (0.35-15build2) …
Setting up libtexluajit2:amd64 (2021.20210626.59705-1ubuntu0.2) …
Setting up libfontbox-java (1:1.8.16-2) …
Setting up rubygems-integration (1.18) …
Setting up libzzip-0-13:amd64 (0.13.72+dfsg.1-1.1) …
Setting up fonts-urw-base35 (20200910-1) …
Setting up poppler-data (0.4.11-1) …
Setting up tex-common (6.17) …
update-language: texlive-base not installed and configured, doing nothing!
Setting up libfontenc1:amd64 (1:1.1.4-1build3) …
Setting up libjbig2dec0:amd64 (0.19-3build2) …
Setting up libteckit0:amd64 (2.5.11+ds1-1) …
Setting up libapache-pom-java (18-1) …
Setting up ruby-net-telnet (0.1.1-2) …
Setting up xfonts-encodings (1:1.0.5-0ubuntu2) …
Setting up t1utils (1.41-4build2) …
Setting up libidn12:amd64 (1.38-4ubuntu1) …
Setting up fonts-texgyre (20180621-3.1) …
Setting up libkpathsea6:amd64 (2021.20210626.59705-1ubuntu0.2) …
Setting up ruby-webrick (1.7.0-3ubuntu0.1) …
Setting up libcmark-gfm0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) …
Setting up fonts-lmodern (2.004.5-6.1) …
Setting up libcmark-gfm-extensions0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) …
Setting up fonts-droid-fallback (1:6.0.1r16-1.1build1) …
Setting up pandoc-data (2.9.2.1-3ubuntu2) …
Setting up ruby-xmlrpc (0.3.2-1ubuntu0.1) …
Setting up libsynctex2:amd64 (2021.20210626.59705-1ubuntu0.2) …
Setting up libgs9-common (9.55.0~dfsg1-0ubuntu5.10) …
Setting up teckit (2.5.11+ds1-1) …
Setting up libpdfbox-java (1:1.8.16-2) …
Setting up libgs9:amd64 (9.55.0~dfsg1-0ubuntu5.10) …
Setting up preview-latex-style (12.2-1ubuntu1) …
Setting up libcommons-parent-java (43-1) …
Setting up dvisvgm (2.13.1-1) …
Setting up libcommons-logging-java (1.2-2) …
Setting up xfonts-utils (1:7.7+6build2) …
Setting up libptexenc1:amd64 (2021.20210626.59705-1ubuntu0.2) …
Setting up pandoc (2.9.2.1-3ubuntu2) …
```

```
Setting up texlive-binaries (2021.20210626.59705-1ubuntu0.2) …
update-alternatives: using /usr/bin/xdvi-xaw to provide /usr/bin/xdvi.bin
(xdvi.bin) in auto mode
update-alternatives: using /usr/bin/bibtex.original to provide /usr/bin/bibtex
(bibtex) in auto mode
Setting up lmodern (2.004.5-6.1) …
Setting up texlive-base (2021.20220204-1) …
/usr/bin/ucfr
/usr/bin/ucfr
/usr/bin/ucfr
/usr/bin/ucfr
mktexlsr: Updating /var/lib/texmf/ls-R-TEXLIVEDIST…
mktexlsr: Updating /var/lib/texmf/ls-R-TEXMFMAIN…
mktexlsr: Updating /var/lib/texmf/ls-R…
mktexlsr: Done.
tl-paper: setting paper size for dvips to a4:
/var/lib/texmf/dvips/config/config-paper.ps
tl-paper: setting paper size for dvipdfmx to a4:
/var/lib/texmf/dvipdfmx/dvipdfmx-paper.cfg
tl-paper: setting paper size for xdvi to a4: /var/lib/texmf/xdvi/XDvi-paper
tl-paper: setting paper size for pdftex to a4: /var/lib/texmf/tex/generic/tex-
ini-files/pdftexconfig.tex
Setting up tex-gyre (20180621-3.1) …
Setting up texlive-plain-generic (2021.20220204-1) …
Setting up texlive-latex-base (2021.20220204-1) …
Setting up texlive-latex-recommended (2021.20220204-1) …
Setting up texlive-pictures (2021.20220204-1) …
Setting up texlive-fonts-recommended (2021.20220204-1) …
Setting up tipa (2:1.3-21) …
Setting up texlive (2021.20220204-1) …
Setting up texlive-latex-extra (2021.20220204-1) …
Setting up texlive-xetex (2021.20220204-1) …
Setting up rake (13.0.6-2) …
Setting up libruby3.0:amd64 (3.0.2-7ubuntu2.8) …
Setting up ruby3.0 (3.0.2-7ubuntu2.8) …
Setting up ruby (1:3.0~exp1) …
Setting up ruby-rubygems (3.3.5-2) …
Processing triggers for man-db (2.10.2-1) …
Processing triggers for fontconfig (2.13.1-4.2ubuntu5) …
Processing triggers for libc-bin (2.35-0ubuntu3.4) …
/sbin/ldconfig.real: /usr/local/lib/libtbb.so.12 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libur_loader.so.0 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtcm.so.1 is not a symbolic link
```

```
/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc_proxy.so.2 is not a symbolic
link

/sbin/ldconfig.real: /usr/local/lib/libur_adapter_opencl.so.0 is not a symbolic
link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_0.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libur_adapter_level_zero.so.0 is not a
symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtcm_debug.so.1 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libhwloc.so.15 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libumf.so.0 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc.so.2 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_5.so.3 is not a symbolic link

Processing triggers for tex-common (6.17) …
Running updmap-sys. This may take some time… done.
Running mktexlsr /var/lib/texmf … done.
Building format(s) --all.
        This may take some time… done.
Collecting pypandoc
  Downloading pypandoc-1.14-py3-none-any.whl.metadata (16 kB)
Downloading pypandoc-1.14-py3-none-any.whl (21 kB)
Installing collected packages: pypandoc
Successfully installed pypandoc-1.14
```

[28]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[29]:
```
!cp /content/drive/MyDrive/Colab Notebooks/Jose'sIndependentResearch/Views.
 ↪ipynb ./
```

```
/bin/bash: -c: line 1: unexpected EOF while looking for matching `''
/bin/bash: -c: line 2: syntax error: unexpected end of file
```

[ ]:
```

```