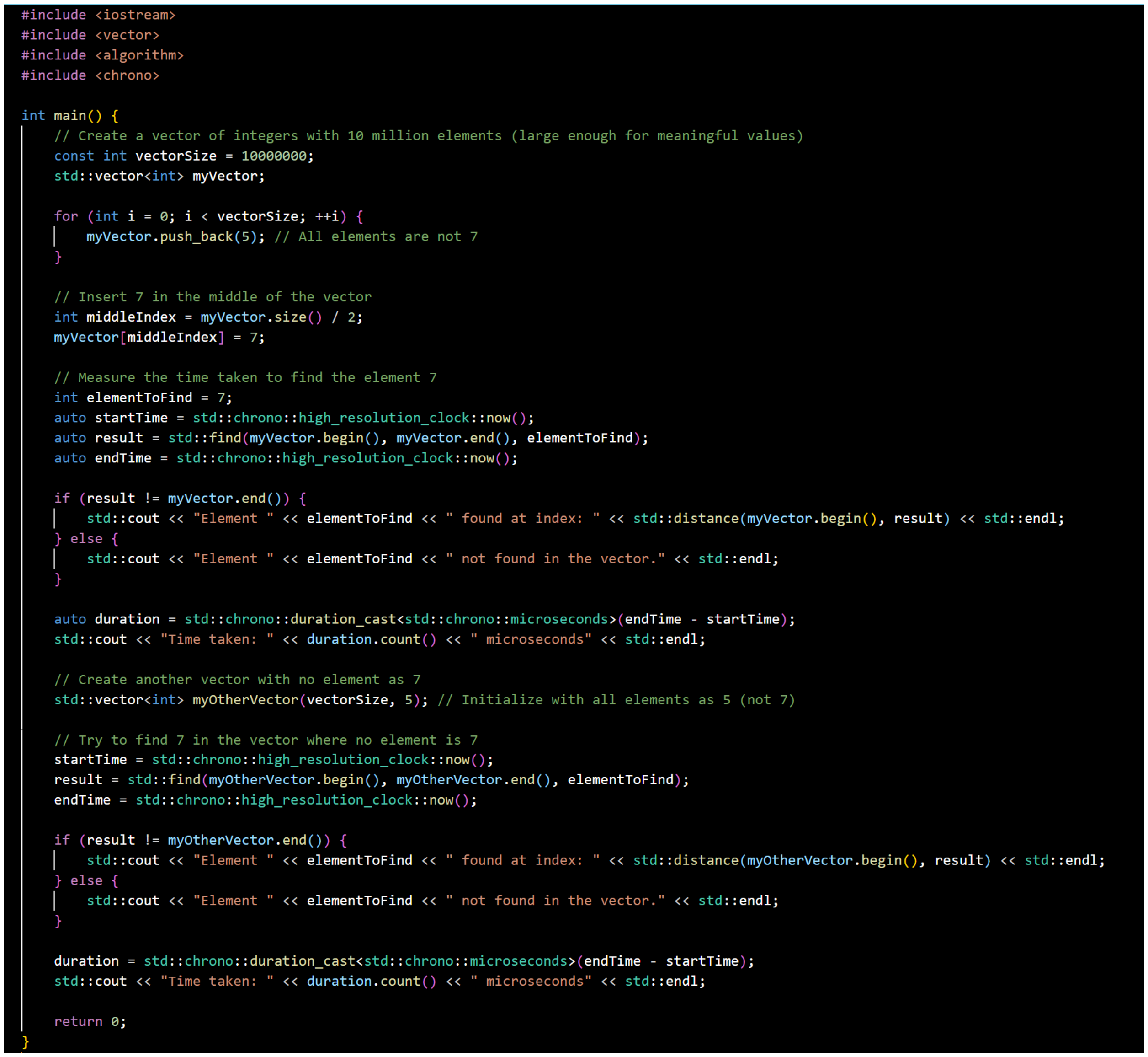
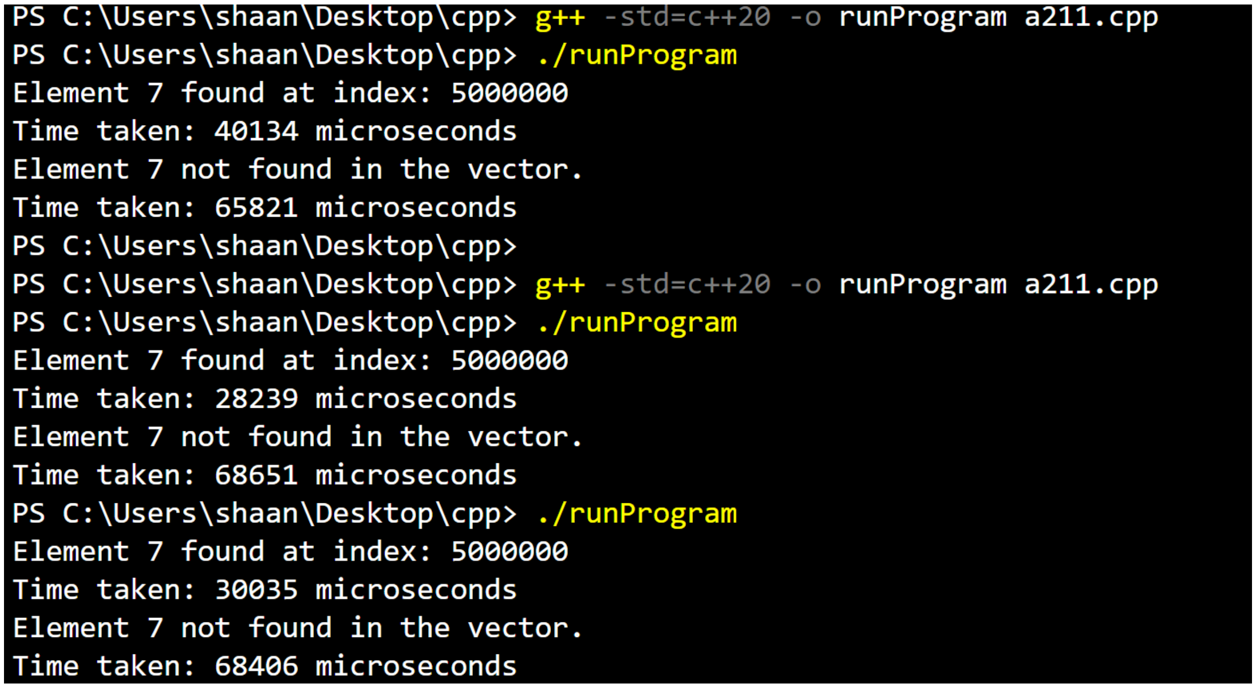
747993_SHAANREHSI_A2

std::find() on a vector<int>:
Use a vector large enough to get meaningful values on your machine (1,000,000 elements are likely not enough). Measure the time needed to
Find the 7 in the middle of a vector<int> where all other elements are not 7.

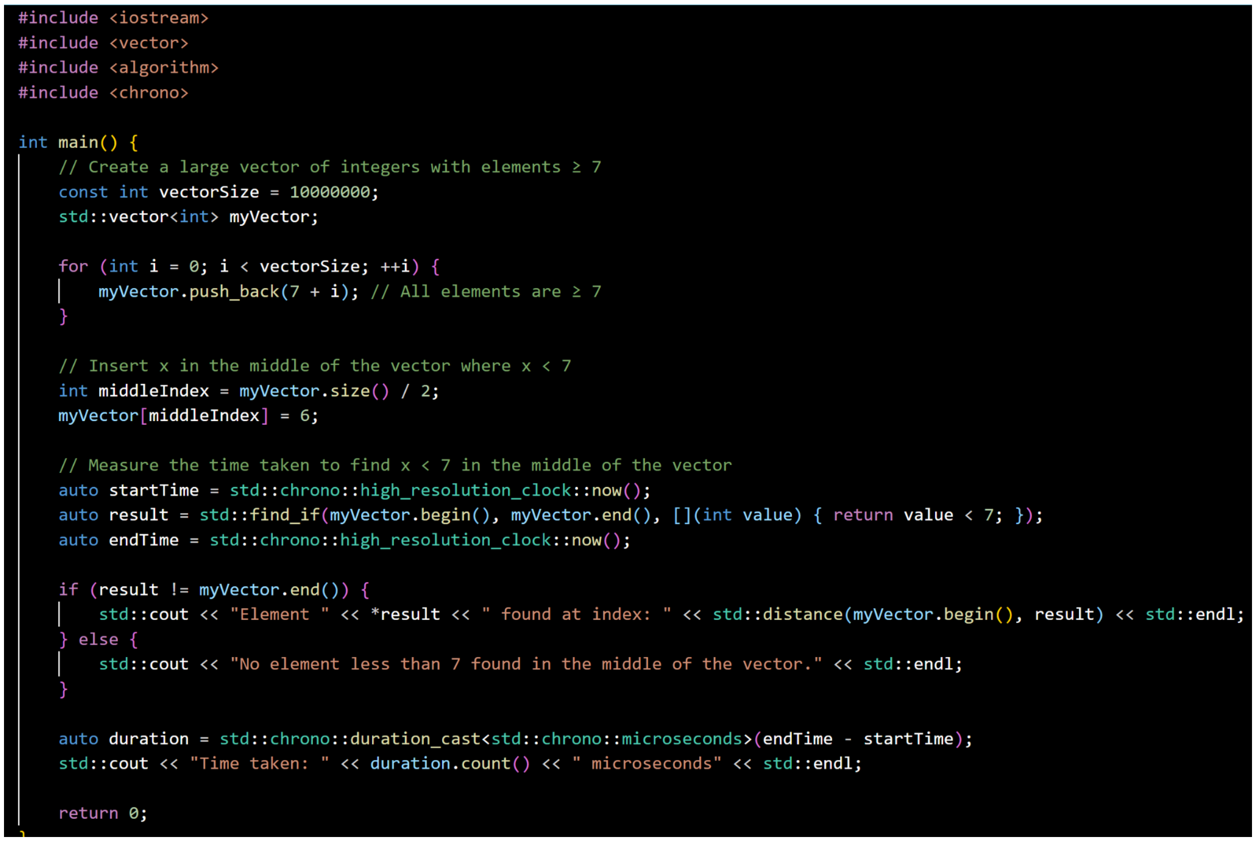


```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <chrono>

int main() {
    // Create a vector of integers with 10 million elements (large enough for meaningful values)
    const int vectorSize = 10000000;
    std::vector<int> myVector;

    for (int i = 0; i < vectorSize; ++i) {
        myVector.push_back(5); // All elements are not 7
    }

    // Insert 7 in the middle of the vector
    int middleIndex = myVector.size() / 2;
    myVector[middleIndex] = 7;

    // Measure the time taken to find the element 7
    int elementToFind = 7;
    auto startTime = std::chrono::high_resolution_clock::now();
    auto result = std::find(myVector.begin(), myVector.end(), elementToFind);
    auto endTime = std::chrono::high_resolution_clock::now();

    if (result != myVector.end()) {
        std::cout << "Element " << elementToFind << " found at index: " << std::distance(myVector.begin(), result) << std::endl;
    } else {
        std::cout << "Element " << elementToFind << " not found in the vector." << std::endl;
    }

    auto duration = std::chrono::duration_cast<std::chrono::microseconds>(endTime - startTime);
    std::cout << "Time taken: " << duration.count() << " microseconds" << std::endl;

    // Create another vector with no element as 7
    std::vector<int> myOtherVector(vectorSize, 5); // Initialize with all elements as 5 (not 7)

    // Try to find 7 in the vector where no element is 7
    startTime = std::chrono::high_resolution_clock::now();
    result = std::find(myOtherVector.begin(), myOtherVector.end(), elementToFind);
    endTime = std::chrono::high_resolution_clock::now();

    if (result != myOtherVector.end()) {
        std::cout << "Element " << elementToFind << " found at index: " << std::distance(myOtherVector.begin(), result) << std::endl;
    } else {
        std::cout << "Element " << elementToFind << " not found in the vector." << std::endl;
    }

    duration = std::chrono::duration_cast<std::chrono::microseconds>(endTime - startTime);
    std::cout << "Time taken: " << duration.count() << " microseconds" << std::endl;

    return 0;
}
```

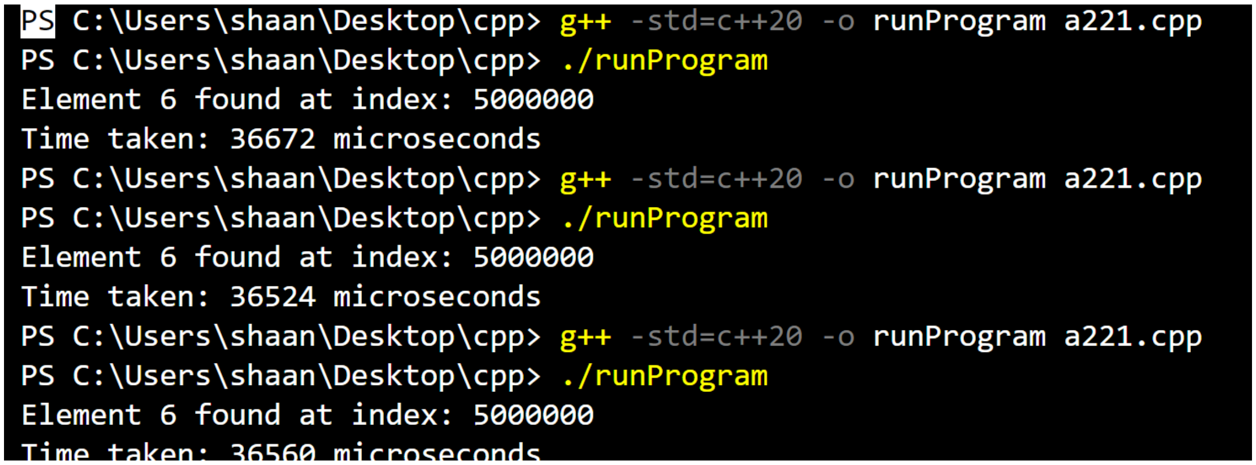


```
PS C:\Users\shaan\Desktop\cpp> g++ -std=c++20 -o runProgram a211.cpp
PS C:\Users\shaan\Desktop\cpp> ./runProgram
Element 7 found at index: 5000000
Time taken: 40134 microseconds
Element 7 not found in the vector.
Time taken: 65821 microseconds
PS C:\Users\shaan\Desktop\cpp>
PS C:\Users\shaan\Desktop\cpp> g++ -std=c++20 -o runProgram a211.cpp
PS C:\Users\shaan\Desktop\cpp> ./runProgram
Element 7 found at index: 5000000
Time taken: 28239 microseconds
Element 7 not found in the vector.
Time taken: 68651 microseconds
PS C:\Users\shaan\Desktop\cpp> ./runProgram
Element 7 found at index: 5000000
Time taken: 30035 microseconds
Element 7 not found in the vector.
Time taken: 68406 microseconds
```

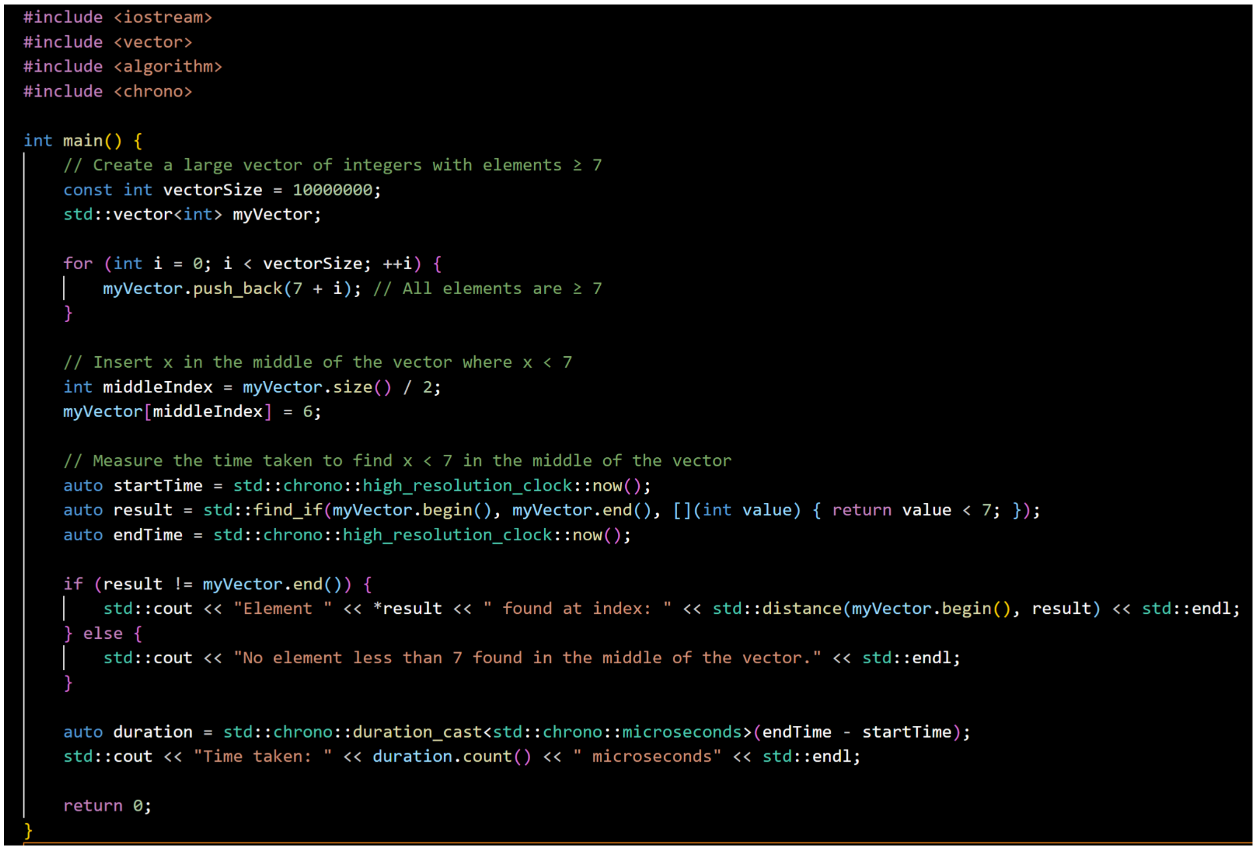
std::find_if() on a vector<int>:
Use a vector large enough to get meaningful values on your machine. Measure the time needed to
Find the x < 7 in the middle of a vector<int> where all other elements are ≥7.

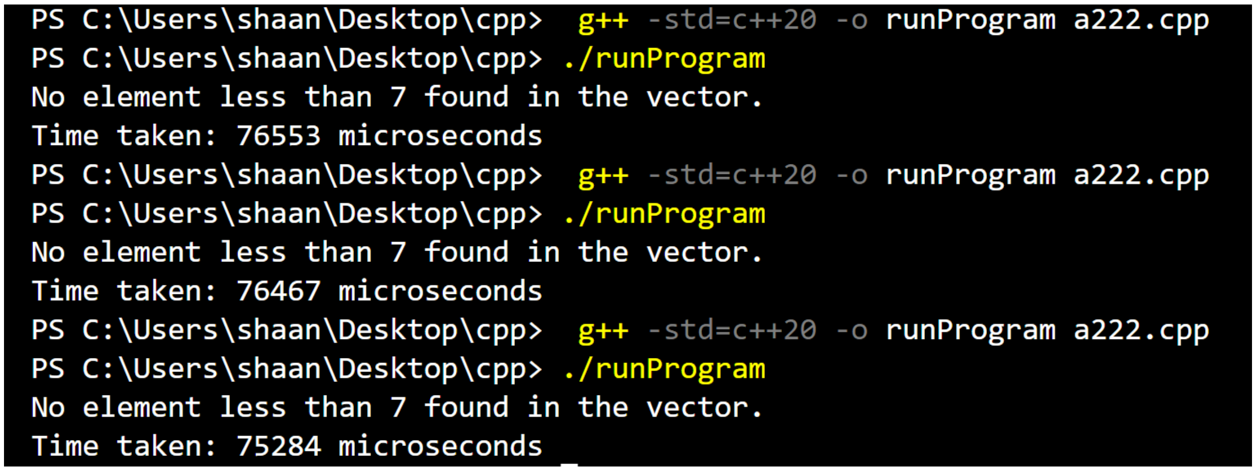


```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <chrono>

int main() {
    // Create a large vector of integers with elements ≥ 7
    const int vectorSize = 10000000;
    std::vector<int> myVector;

    for (int i = 0; i < vectorSize; ++i) {
        myVector.push_back(7 + i); // All elements are ≥ 7
    }

    // Insert x in the middle of the vector where x < 7
    int middleIndex = myVector.size() / 2;
    myVector[middleIndex] = 6;

    // Measure the time taken to find x < 7 in the middle of the vector
    auto startTime = std::chrono::high_resolution_clock::now();
    auto result = std::find_if(myVector.begin(), myVector.end(), [](int value) { return value < 7; });
    auto endTime = std::chrono::high_resolution_clock::now();

    if (result != myVector.end()) {
        std::cout << "Element " << *result << " found at index: " << std::distance(myVector.begin(), result) << std::endl;
    } else {
        std::cout << "No element less than 7 found in the middle of the vector." << std::endl;
    }

    auto duration = std::chrono::duration_cast<std::chrono::microseconds>(endTime - startTime);
    std::cout << "Time taken: " << duration.count() << " microseconds" << std::endl;

    return 0;
```


```
PS C:\Users\shaan\Desktop\cpp> g++ -std=c++20 -o runProgram a221.cpp
PS C:\Users\shaan\Desktop\cpp> ./runProgram
Element 6 found at index: 5000000
Time taken: 36672 microseconds
PS C:\Users\shaan\Desktop\cpp> g++ -std=c++20 -o runProgram a221.cpp
PS C:\Users\shaan\Desktop\cpp> ./runProgram
Element 6 found at index: 5000000
Time taken: 36524 microseconds
PS C:\Users\shaan\Desktop\cpp> g++ -std=c++20 -o runProgram a221.cpp
PS C:\Users\shaan\Desktop\cpp> ./runProgram
Element 6 found at index: 5000000
Time taken: 36560 microseconds
```

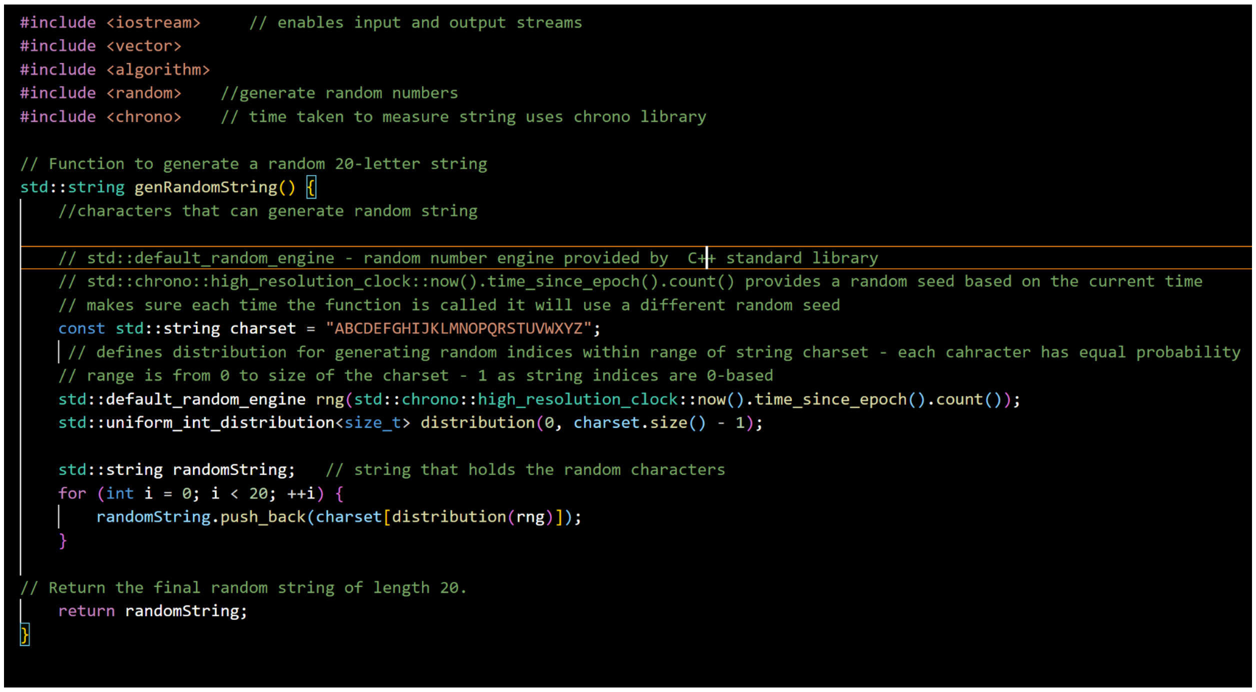
Try to find an x < 7 in a vector<int> where all elements are ≥7.


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

int main() {
    // Create a large vector of integers with elements ≥ 7
    const int vectorSize = 10000000;
    std::vector<int> myVector;

    for (int i = 0; i < vectorSize; ++i) {
        myVector.push_back(7 + i); // All elements are ≥ 7
    }

    // Insert x in the middle of the vector where x < 7
    int middleIndex = myVector.size() / 2;
    myVector[middleIndex] = 6;

    // Measure the time taken to find x < 7 in the middle of the vector
    auto startTime = std::chrono::high_resolution_clock::now();
    auto result = std::find_if(myVector.begin(), myVector.end(), [](int value) { return value < 7; });
    auto endTime = std::chrono::high_resolution_clock::now();

    if (result != myVector.end()) {
        std::cout << "Element " << *result << " found at index: " << std::distance(myVector.begin(), result) << std::endl;
    } else {
        std::cout << "No element less than 7 found in the middle of the vector." << std::endl;
    }

    auto duration = std::chrono::duration_cast<std::chrono::microseconds>(endTime - startTime);
    std::cout << "Time taken: " << duration.count() << " microseconds" << std::endl;

    return 0;
}
```



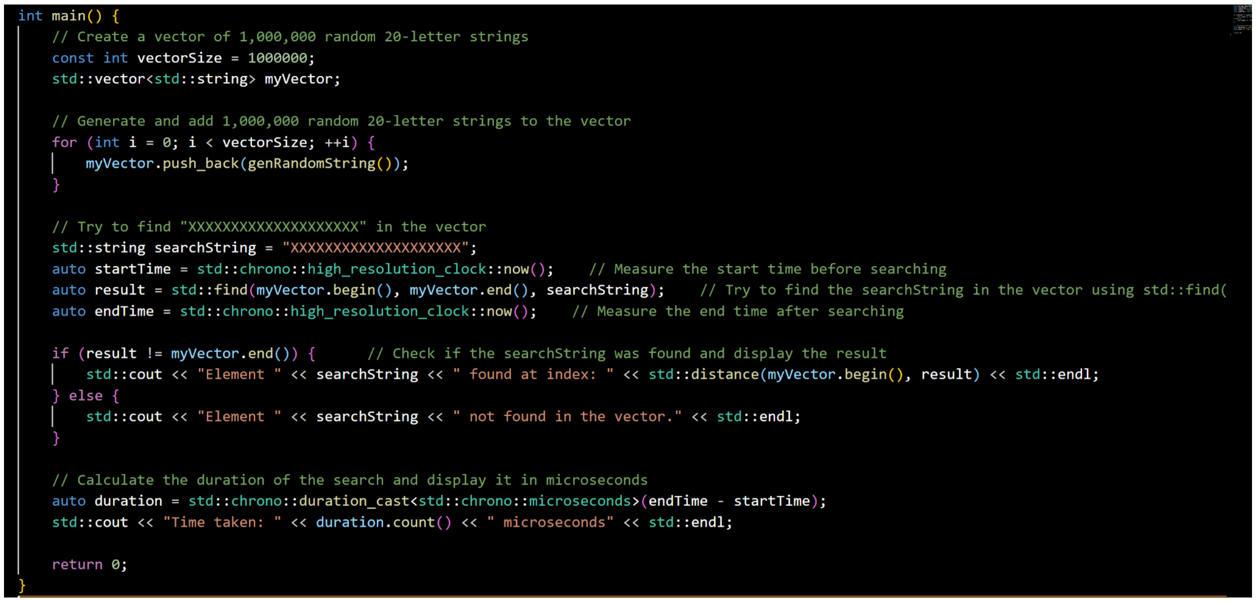
```
PS C:\Users\shaan\Desktop\cpp>  g++ -std=c++20 -o runProgram a222.cpp
PS C:\Users\shaan\Desktop\cpp> ./runProgram
No element less than 7 found in the vector.
Time taken: 76553 microseconds
PS C:\Users\shaan\Desktop\cpp>  g++ -std=c++20 -o runProgram a222.cpp
PS C:\Users\shaan\Desktop\cpp> ./runProgram
No element less than 7 found in the vector.
Time taken: 76467 microseconds
PS C:\Users\shaan\Desktop\cpp>  g++ -std=c++20 -o runProgram a222.cpp
PS C:\Users\shaan\Desktop\cpp> ./runProgram
No element less than 7 found in the vector.
Time taken: 75284 microseconds
```

std::find() on a vector<std::string>:
Generate a vector<string> containing 1,000,000 random 20-letter strings (don't bother timing this). Measure the time needed to
Try to find XXXXXXXXXXXXXXXXXXXX (20 Xs) in that vector using std::find().

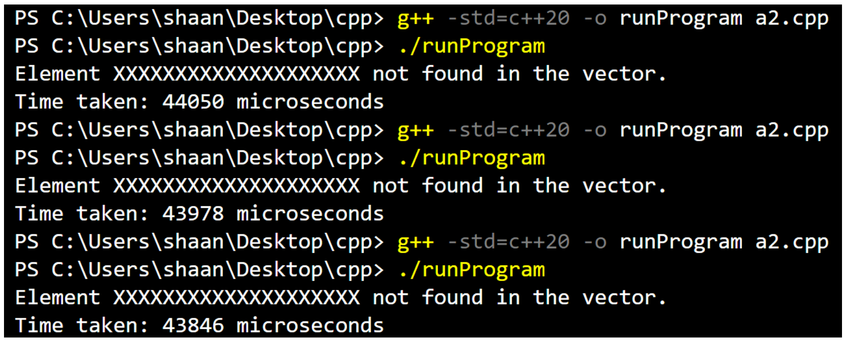


```cpp
#include <iostream>     // enables input and output streams
#include <vector>
#include <algorithm>
#include <random>    //generate random numbers
#include <chrono>    // time taken to measure string uses chrono library

// Function to generate a random 20-letter string
std::string genRandomString() {
    //characters that can generate random string

    // std::default_random_engine - random number engine provided by  C++ standard library
    // std::chrono::high_resolution_clock::now().time_since_epoch().count() provides a random seed based on the current time
    // makes sure each time the function is called it will use a different random seed
    const std::string charset = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    // defines distribution for generating random indices within range of string charset - each cahracter has equal probability
    // range is from 0 to size of the charset - 1 as string indices are 0-based
    std::default_random_engine rng(std::chrono::high_resolution_clock::now().time_since_epoch().count());
    std::uniform_int_distribution<size_t> distribution(0, charset.size() - 1);

    std::string randomString;   // string that holds the random characters
    for (int i = 0; i < 20; ++i) {
        randomString.push_back(charset[distribution(rng)]);
    }

// Return the final random string of length 20.
    return randomString;
}
```

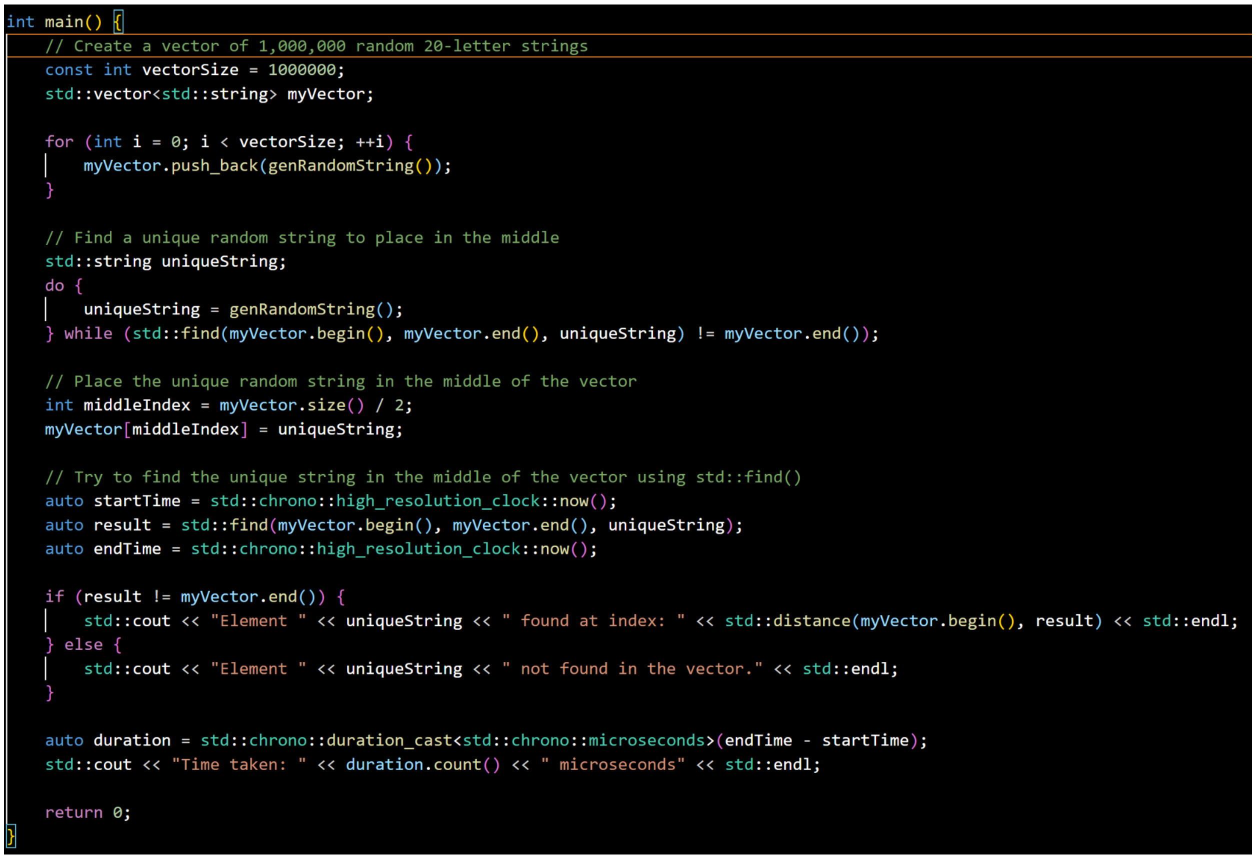
Place XXXXXXXXXXXXXXXXXXXX (or whatever value is not present) in the middle and find it using std::find().



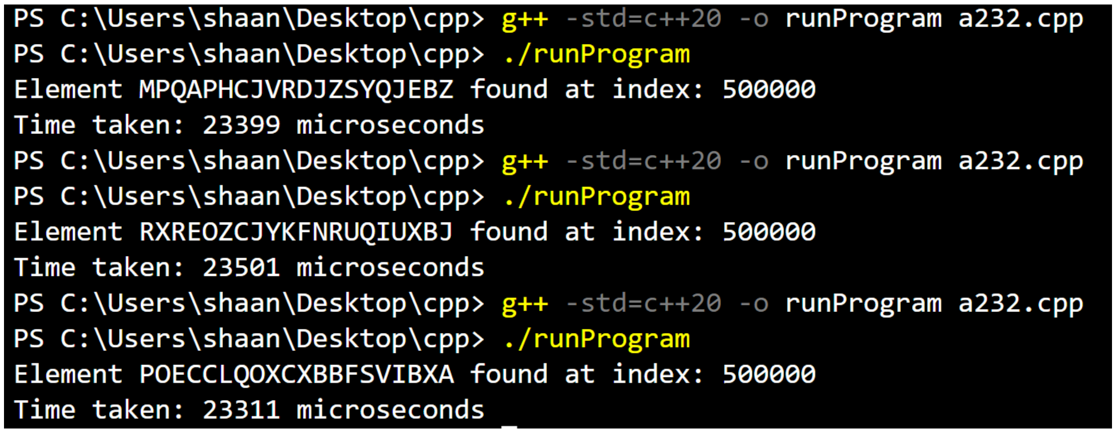
```cpp
int main() {
    // Create a vector of 1,000,000 random 20-letter strings
    const int vectorSize = 1000000;
    std::vector<std::string> myVector;

    // Generate and add 1,000,000 random 20-letter strings to the vector
    for (int i = 0; i < vectorSize; ++i) {
        myVector.push_back(genRandomString());
    }

    // Try to find "XXXXXXXXXXXXXXXXXXXX" in the vector
    std::string searchString = "XXXXXXXXXXXXXXXXXXXX";
    auto startTime = std::chrono::high_resolution_clock::now();    // Measure the start time before searching
    auto result = std::find(myVector.begin(), myVector.end(), searchString);    // Try to find the searchString in the vector using std::find(
    auto endTime = std::chrono::high_resolution_clock::now();    // Measure the end time after searching

    if (result != myVector.end()) {      // Check if the searchString was found and display the result
        std::cout << "Element " << searchString << " found at index: " << std::distance(myVector.begin(), result) << std::endl;
    } else {
        std::cout << "Element " << searchString << " not found in the vector." << std::endl;
    }

    // Calculate the duration of the search and display it in microseconds
    auto duration = std::chrono::duration_cast<std::chrono::microseconds>(endTime - startTime);
    std::cout << "Time taken: " << duration.count() << " microseconds" << std::endl;

    return 0;
}
```



```
PS C:\Users\shaan\Desktop\cpp> g++ -std=c++20 -o runProgram a2.cpp
PS C:\Users\shaan\Desktop\cpp> ./runProgram
Element XXXXXXXXXXXXXXXXXXXX not found in the vector.
Time taken: 44050 microseconds
PS C:\Users\shaan\Desktop\cpp> g++ -std=c++20 -o runProgram a2.cpp
PS C:\Users\shaan\Desktop\cpp> ./runProgram
Element XXXXXXXXXXXXXXXXXXXX not found in the vector.
Time taken: 43978 microseconds
PS C:\Users\shaan\Desktop\cpp> g++ -std=c++20 -o runProgram a2.cpp
PS C:\Users\shaan\Desktop\cpp> ./runProgram
Element XXXXXXXXXXXXXXXXXXXX not found in the vector.
Time taken: 43846 microseconds
```



```cpp
int main() {
    // Create a vector of 1,000,000 random 20-letter strings
    const int vectorSize = 1000000;
    std::vector<std::string> myVector;

    for (int i = 0; i < vectorSize; ++i) {
        myVector.push_back(genRandomString());
    }

    // Find a unique random string to place in the middle
    std::string uniqueString;
    do {
        uniqueString = genRandomString();
    } while (std::find(myVector.begin(), myVector.end(), uniqueString) != myVector.end());

    // Place the unique random string in the middle of the vector
    int middleIndex = myVector.size() / 2;
    myVector[middleIndex] = uniqueString;

    // Try to find the unique string in the middle of the vector using std::find()
    auto startTime = std::chrono::high_resolution_clock::now();
    auto result = std::find(myVector.begin(), myVector.end(), uniqueString);
    auto endTime = std::chrono::high_resolution_clock::now();

    if (result != myVector.end()) {
        std::cout << "Element " << uniqueString << " found at index: " << std::distance(myVector.begin(), result) << std::endl;
    } else {
        std::cout << "Element " << uniqueString << " not found in the vector." << std::endl;
    }

    auto duration = std::chrono::duration_cast<std::chrono::microseconds>(endTime - startTime);
    std::cout << "Time taken: " << duration.count() << " microseconds" << std::endl;

    return 0;
}
```



```
PS C:\Users\shaan\Desktop\cpp> g++ -std=c++20 -o runProgram a232.cpp
PS C:\Users\shaan\Desktop\cpp> ./runProgram
Element MPQAPHCJVRDJZSYQJEBZ found at index: 500000
Time taken: 23399 microseconds
PS C:\Users\shaan\Desktop\cpp> g++ -std=c++20 -o runProgram a232.cpp
PS C:\Users\shaan\Desktop\cpp> ./runProgram
Element RXREOZCJYKFNRUQIUXBJ found at index: 500000
Time taken: 23501 microseconds
PS C:\Users\shaan\Desktop\cpp> g++ -std=c++20 -o runProgram a232.cpp
PS C:\Users\shaan\Desktop\cpp> ./runProgram
Element POECCLQOXCXBBFSVIBXA found at index: 500000
Time taken: 23311 microseconds
```

NOTE: a lot of code I have researched and added so this isn't entirely what I could do independently. I found these tasks quite challenging but I Understand what is going on. I just found it quite hard to do in the two days we had.