QUESTION 2:

PART A:

#include <iostream>

#include <vector>

#include <algorithm> // Include the algorithm header for find\_if

using namespace std;

// Structure to represent a product

struct Product {

int id;

string name;

double price;

int quantity;

};

// Function to add a new product to the inventory

void addProduct(vector<Product>& inventory, int id, const string& name, double price, int quantity) {

// Check if the product with the given ID already exists

for (const Product& product : inventory) {

if (product.id == id) {

cout << "Product with ID " << id << " already exists. Use a different ID.\n";

return;

}

}

// Create a new product and add it to the inventory

Product newProduct = {id, name, price, quantity};

inventory.push\_back(newProduct);

cout << "Product added successfully.\n";

}

void removeProduct(vector<Product>& inventory, int id) {

auto it = find\_if(inventory.begin(), inventory.end(), [id](const Product& product) {

return product.id == id;

});

if (it != inventory.end()) {

inventory.erase(it);

cout << "Product with ID " << id << " removed successfully.\n";

} else {

cout << "Product with ID " << id << " not found in the inventory.\n";

}

}

void displayInventory(const vector<Product>& inventory) {

cout << "Current Inventory:\n";

for (const Product& product : inventory) {

cout << "ID: " << product.id << ", Name: " << product.name << ", Price: " << product.price

<< ", Quantity: " << product.quantity << "\n";

}

}

int main() {

vector<Product> inventory;

addProduct(inventory, 1, "Laptop", 699.99, 5);

addProduct(inventory, 2, "Smartphone", 345.99, 10);

addProduct(inventory, 1, "Tablet", 299.99, 8);

displayInventory(inventory);

removeProduct(inventory, 2);

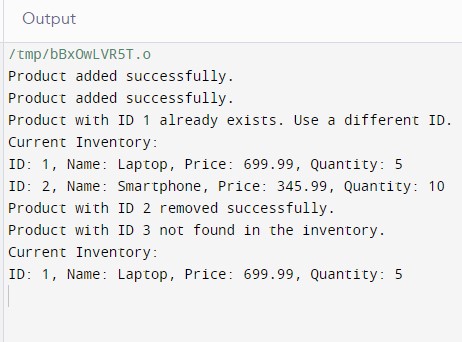
removeProduct(inventory, 3);

displayInventory(inventory);

return 0;

}

**OUTPUT:**



**PART B:**

**BUBBLE SORT:**

#include <iostream>

#include <vector>

#include <chrono>

using namespace std;

using namespace chrono;

void bubbleSort(vector<int>& arr) {

int n = arr.size();

for (int i = 0; i < n - 1; ++i) {

for (int j = 0; j < n - i - 1; ++j) {

if (arr[j] > arr[j + 1]) {

swap(arr[j], arr[j + 1]);

}

}

}

}

int main() {

const int size = 100000;

vector<int> data(size);

// Initialize the vector in descending order

for (int i = size; i > 0; --i) {

data.push\_back(i);

}

// Measure execution time for Bubble Sort

auto start = high\_resolution\_clock::now();

bubbleSort(data);

auto stop = high\_resolution\_clock::now();

auto duration = duration\_cast<milliseconds>(stop - start);

// Print execution time

cout << "Bubble Sort Execution Time: " << duration.count() << " milliseconds\n";

// Print first 10 and last 10 integers

cout << "First 10 integers: ";

for (int i = 0; i < 10; ++i) {

cout << data[i] << " ";

}

cout << "\n";

cout << "Last 10 integers: ";

for (int i = size - 10; i < size; ++i) {

cout << data[i] << " ";

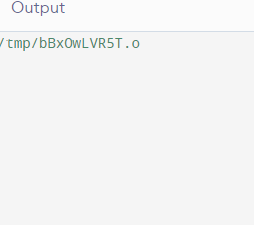
}

cout << "\n";

return 0;

}

**Output:**



**STL SORT:**

#include <iostream>

#include <vector>

#include <chrono>

using namespace std;

using namespace chrono;

void bubbleSort(vector<int>& arr) {

int n = arr.size();

for (int i = 0; i < n - 1; ++i) {

for (int j = 0; j < n - i - 1; ++j) {

if (arr[j] > arr[j + 1]) {

swap(arr[j], arr[j + 1]);

}

}

}

}

int main() {

const int size = 100000;

vector<int> data(size);

// Initialize the vector in descending order

for (int i = size; i > 0; --i) {

data.push\_back(i);

}

// Measure execution time for Bubble Sort

auto start = high\_resolution\_clock::now();

bubbleSort(data);

auto stop = high\_resolution\_clock::now();

auto duration = duration\_cast<milliseconds>(stop - start);

// Print execution time

cout << "Bubble Sort Execution Time: " << duration.count() << " milliseconds\n";

// Print first 10 and last 10 integers

cout << "First 10 integers: ";

for (int i = 0; i < 10; ++i) {

cout << data[i] << " ";

}

cout << "\n";

cout << "Last 10 integers: ";

for (int i = size - 10; i < size; ++i) {

cout << data[i] << " ";

}

cout << "\n";

return 0;

}

**OUTPUT:**

