**2. STL (Algorithms): You are tasked with evaluating the performance of Bubble Sort against the widely-used STL sort algorithm. The goal is to analyze the execution time of each algorithm when sorting a large vector of 100,000 integers in ascending order (initialize the vector in descending order).  
  
CODE:**

#include <iostream>

using namespace std;

void bubbleSort(int arr[], int size) {

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

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

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

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

void printArray(const int arr[], int size) {

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

cout << arr[i] << " ";

}

cout << endl;

}

int main() {

const int size = 100000;

int data[size];

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

data[i] = size - i;

}

clock\_t startBubbleSort = clock();

bubbleSort(data, size);

clock\_t endBubbleSort = clock();

double durationBubbleSort = double(endBubbleSort - startBubbleSort) / CLOCKS\_PER\_SEC \* 1000000.0;

cout << "Bubble Sort Execution Time: " << durationBubbleSort << " microseconds" << endl;

cout << "First 10 elements: ";

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

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

}

cout << "\nLast 10 elements: ";

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

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

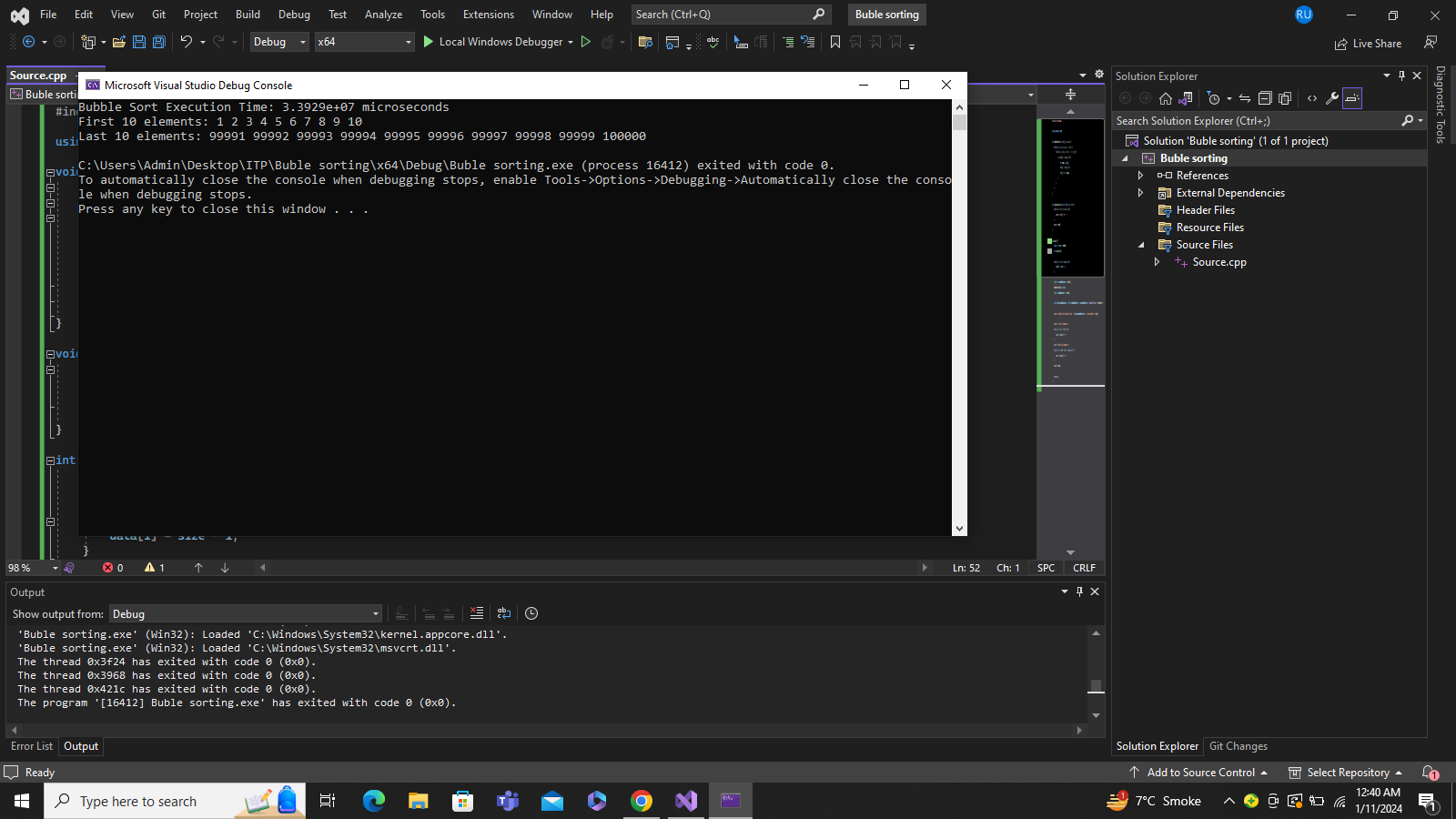
}

cout << endl;

return 0;

}

**OUTPUT SCREEN SHORT:**

****