

Part 1

```
// *****  
  
// Description: This program sorts and averages temperature  
// Author: Aaron Nguyen  
// COMSC-165-5065  
// Date: September 21, 2025  
// Status: Complete  
// *****  
  
  
#include <iostream>  
#include <vector>  
#include <iomanip>  
#include <string>  
  
  
  
std::vector<int> inputTemperatures();  
void bubbleSortTemperatures(std::vector<int>& arr);  
double calculateAverage(const std::vector<int>& arr);  
void printTemperatures(const std::vector<int>& arr, const  
std::string& label);  
  
int main() {  
    std::cout << "Welcome to the Temperature Analysis Program" <<  
std::endl;  
  
    std::vector<int> temperatures = inputTemperatures();  
    printTemperatures(temperatures, "\nOriginal temperatures:");  
}
```

```

    double averageTemp = calculateAverage(temperatures);

    bubbleSortTemperatures(temperatures);
    printTemperatures(temperatures, "Sorted temperatures
(ascending):");

    std::cout << "\nThe average temperature is:" << std::fixed <<
std::setprecision(2) << averageTemp << std::endl;

    return 0;
}

// Prompts user to enter the number of temperatures and then the
temp itself. Returns a list of numbers.

std::vector<int> inputTemperatures() {
    std::vector<int> temps;
    int numTemps = 0;

    while (true) {
        std::cout << "Enter the number of temperatures up to 10:
";

        std::cin >> numTemps;
        if (std::cin.good() && numTemps > 0 && numTemps <= 10) {
            break;
        }
    }

    for (int i = 0; i < numTemps; ++i) {
        int temp;

```

```

        std::cout << "Enter temperature for day " << i + 1 << ":
";
        std::cin >> temp;
        while (std::cin.fail()) {
            std::cout << "Invalid input. Please enter an
integer." << std::endl;
            std::cin.clear();
            std::cin.ignore(10000, '\n');
            std::cout << "Enter temperature for day " << i + 1 <<
": ";
            std::cin >> temp;
        }
        temps.push_back(temp);
    }
    return temps;
}

```

// works by repeatedly stepping through the list, comparing each pair of adjacent items and swapping them if they are in the wrong order. repeated until no swaps are needed,

```

void bubbleSortTemperatures(std::vector<int>& arr) {
    int n = arr.size();
    bool swapped;
    for (int i = 0; i < n - 1; i++) {
        swapped = false;
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                // Swap arr[j] and arr[j+1]
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
                swapped = true;
            }
        }
    }
}

```

```

        }

    }

    // If no two elements were swapped by inner loop = then
break
    if (!swapped) break;

}
}

// Calculates the average of a list of integers using a manual
loop.
double calculateAverage(const std::vector<int>& arr) {
    if (arr.empty()) return 0.0;

    double sum = 0.0;
    for (int temp : arr) {
        sum += temp;
    }

    return sum / arr.size();
}

// print the contents of the temperature vector.
void printTemperatures(const std::vector<int>& arr, const
std::string& label) {
    std::cout << label;
    for (int temp : arr) {
        std::cout << " " << temp;
    }
    std::cout << std::endl;
}

```



Part 2

```
// *****  
  
// Description: This program is plays tic tac toe  
// Author: Aaron Nguyen  
// COMSC-165-5065  
// Date: September 21, 2025  
// Status: Complete  
  
// *****  
  
#include <iostream>  
#include <vector>  
  
void printBoard(const char board[3][3]);  
void getPlayerInput(char currentPlayer, char board[3][3]);  
bool checkWin(const char board[3][3], char player);  
bool checkTie(const char board[3][3]);  
  
int main() {  
    char board[3][3];  
    char currentPlayer = 'X';  
    for (int i = 0; i < 3; ++i) {  
        for (int j = 0; j < 3; ++j) {  
            board[i][j] = '*';  
        }  
    }  
  
    while (true) {
```

```

        printBoard(board);

        getPlayerInput(currentPlayer, board);

        if (checkWin(board, currentPlayer)) {
            printBoard(board); // Display final board with
winning move
            std::cout << "\nPlayer " << currentPlayer << " wins!
Congratulations!" << std::endl;
            break;
        }

        if (checkTie(board)) {
            printBoard(board); // Display final board for tie
            std::cout << "\nThe game is a tie!" << std::endl;
            break;
        }

        if (currentPlayer == 'X') {
            currentPlayer = 'O';
        } else {
            currentPlayer = 'X';
        }
    }

    return 0;
}

// displays board
void printBoard(const char board[3][3]) {
    std::cout << "\n   1   2   3" << std::endl; // Column headers

```

```

std::cout << "  -----" << std::endl;
for (int i = 0; i < 3; ++i) {
    std::cout << i + 1 << " |"; // Row header (1-based for
user)
    for (int j = 0; j < 3; ++j) {
        std::cout << " " << board[i][j] << " |";
    }
    std::cout << std::endl;
    std::cout << "  -----" << std::endl;
}
}

//prompts player for their move and checks if its valid

void getPlayerInput(char currentPlayer, char board[3][3]) {
    int row = -1;
    int col = -1;

    while (true) {
        std::cout << "\nPlayer " << currentPlayer << "'s turn."
<< std::endl;
        std::cout << "Enter row (1-3): ";
        std::cin >> row;
        std::cout << "Enter column (1-3): ";
        std::cin >> col;

        //checks if input is legal
        if (std::cin.fail() || row < 1 || row > 3 || col < 1 ||
col > 3) {
            std::cout << "Invalid input. Please enter numbers
between 1 and 3 for row and column." << std::endl;

```



```

        } else if (board[row - 1][col - 1] != '*') {
            std::cout << "That spot is already taken! Please
select another location." << std::endl;
        } else {
            board[row - 1][col - 1] = currentPlayer;
            break;
        }
    }
}

// check if play has won,

bool checkWin(const char board[3][3], char player) {
    // Check rows and columns
    for (int i = 0; i < 3; ++i) {
        if ((board[i][0] == player && board[i][1] == player &&
board[i][2] == player) || // Check row i
            (board[0][i] == player && board[1][i] == player &&
board[2][i] == player)) { // Check column i
            return true;
        }
    }

    // Check diagonals
    if ((board[0][0] == player && board[1][1] == player &&
board[2][2] == player) || // Top-left to bottom-right
        (board[0][2] == player && board[1][1] == player &&
board[2][0] == player)) { // Top-right to bottom-left
        return true;
    }
}

```

```
    return false;
}

//
-----

//checks if tie or not
bool checkTie(const char board[3][3]) {
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            if (board[i][j] == '*') {
                return false;
            }
        }
    }
    return true;
}
```

The game is a tie!
aaronnguyen@aaa Assignment 3 % cd "/Users/aaronnguyen/Desktop/Projects/Assignment 3/" && g++ part2.cpp -o part2 && "/Users/aaronnguyen/Desktop/Projects/Assignment 3/"part2

```
  1  2  3
  ---
1 | * | * | * |
  ---
2 | * | * | * |
  ---
3 | * | * | * |
  ---
```

Player X's turn.
Enter row (1-3): 1
Enter column (1-3): 1

```
  1  2  3
  ---
1 | X | * | * |
  ---
2 | * | * | * |
  ---
3 | * | * | * |
  ---
```

Player O's turn.
Enter row (1-3): 1
Enter column (1-3): 1
That spot is already taken! Please select another location.

Player O's turn.
Enter row (1-3): 2
Enter column (1-3): 2

```
  1  2  3
  ---
1 | X | * | * |
  ---
2 | * | 0 | * |
  ---
3 | * | * | * |
  ---
```

Player X's turn.
Enter row (1-3): 3
Enter column (1-3): 3

```
  1  2  3
  ---
1 | X | * | * |
  ---
2 | * | 0 | * |
  ---
3 | * | * | X |
  ---
```

Player O's turn.
Enter row (1-3): 2
Enter column (1-3): 1

```
  1  2  3
  ---
1 | X | * | * |
  ---
2 | 0 | 0 | * |
  ---
3 | * | * | X |
  ---
```

Player X's turn.
Enter row (1-3): 3
Enter column (1-3): 3
That spot is already taken! Please select another location.

Player X's turn.
Enter row (1-3): 3
Enter column (1-3): 1

```
  1  2  3
  ---
1 | X | * | * |
  ---
2 | 0 | 0 | * |
  ---
3 | X | * | X |
  ---
```

Player O's turn.
Enter row (1-3): 2
Enter column (1-3): 3

```
  1  2  3
  ---
1 | X | * | * |
  ---
2 | 0 | 0 | 0 |
  ---
3 | X | * | X |
  ---
```

Player O wins! Congratulations!
aaronnguyen@aaa Assignment 3 % █

Part 3

```
#include <iostream>
#include <vector>          // Added for std::vector
using namespace std;

// Function Prototype (No changes needed here)
bool testPIN(const int [], const int [], int);

int main ()
{
    const int NUM_DIGITS = 7; // Number of digits in a PIN

    vector<int> pin1 = {2, 4, 1, 8, 7, 9, 0};
    vector<int> pin2 = {2, 4, 6, 8, 7, 9, 0};
    vector<int> pin3 = {1, 2, 3, 4, 5, 6, 7};

    if (testPIN(pin1.data(), pin2.data(), NUM_DIGITS))
        cout << "ERROR: pin1 and pin2 report to be the same.\n";
    else
        cout << "SUCCESS: pin1 and pin2 are different.\n";

    if (testPIN(pin1.data(), pin3.data(), NUM_DIGITS))
        cout << "ERROR: pin1 and pin3 report to be the same.\n";
    else
        cout << "SUCCESS: pin1 and pin3 are different.\n";

    if (testPIN(pin1.data(), pin1.data(), NUM_DIGITS))
        cout << "SUCCESS: pin1 and pin1 report to be the same.\n";
    else
        cout << "ERROR: pin1 and pin1 report to be different.\n";

    return 0;
}

//*****

// The following function accepts two int arrays. The arrays are      *
// compared. If they contain the same values, true is returned.      *
// If the contain different values, false is returned.                *
```

```
//*****
```

```
bool testPIN(const int custPIN[], const int databasePIN[], int size)
```

```
{
```

```
    for (int index = 0; index < size; index++)
```

```
    {
```

```
        if (custPIN[index] != databasePIN[index])
```

```
            return false; // We've found two different values.
```

```
    }
```

```
    return true; // If we make it this far, the values are the same.
```

```
}
```

```
aaronnguyen@aaa Assignment 3 % cd "/Users/aaronnguyen/Desktop/Projects/Assignment 3/" && g++ part3.cpp -o part3 && "/Users/aaronnguyen/Desktop/Projects/Assignment 3/"part3
SUCCESS: pin1 and pin2 are different.
SUCCESS: pin1 and pin3 are different.
SUCCESS: pin1 and pin1 report to be the same.
aaronnguyen@aaa Assignment 3 %
```