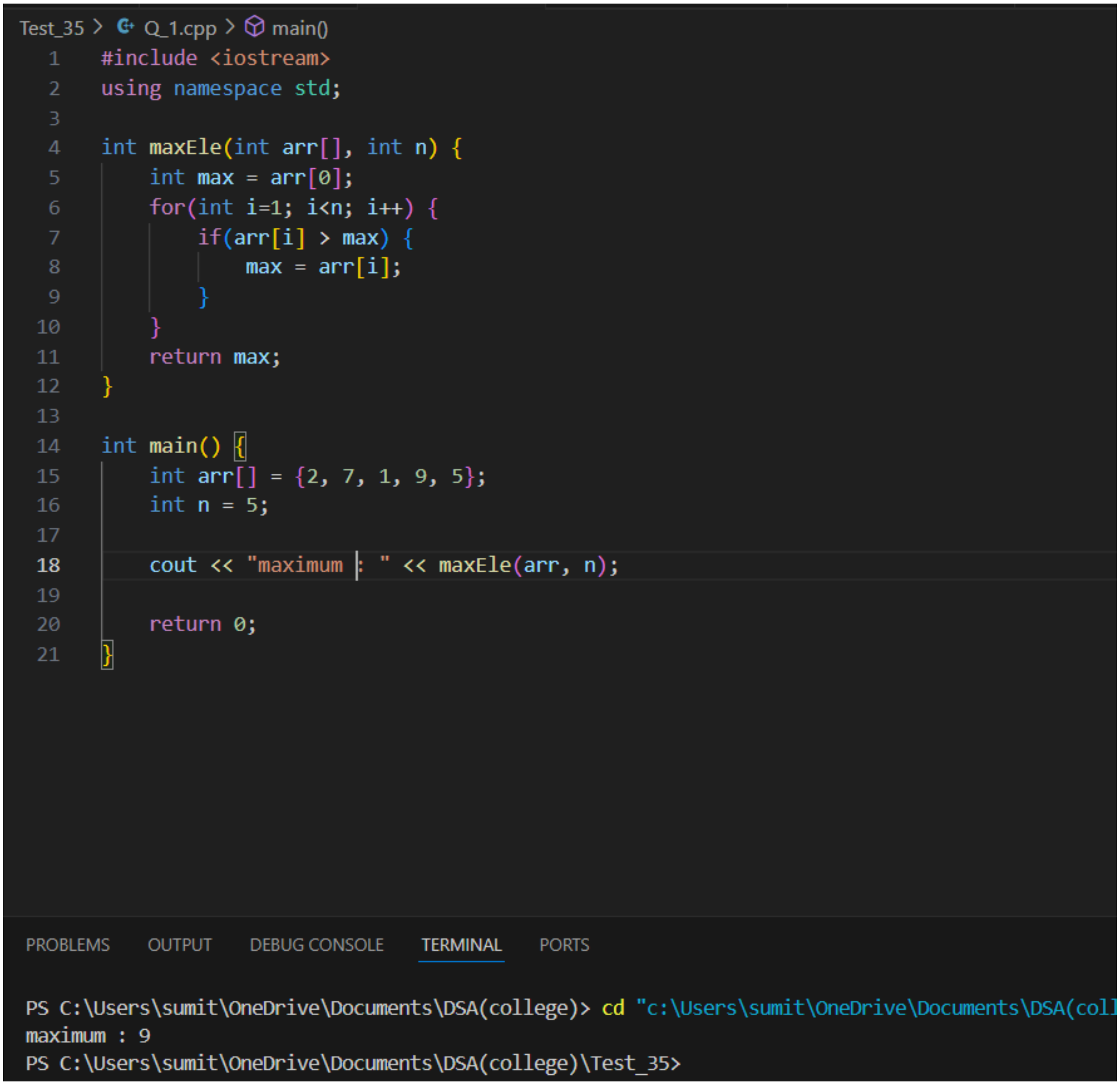
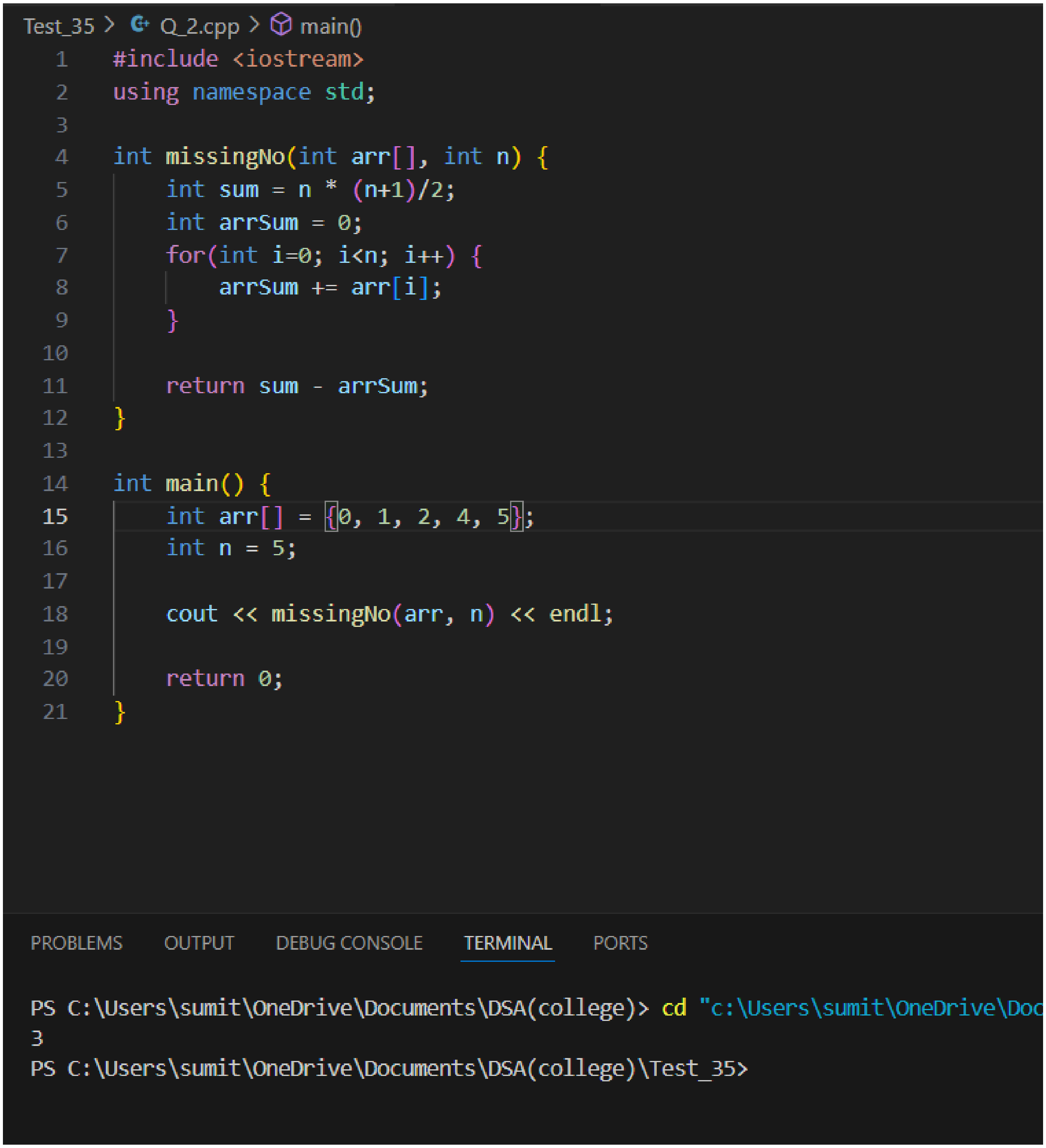


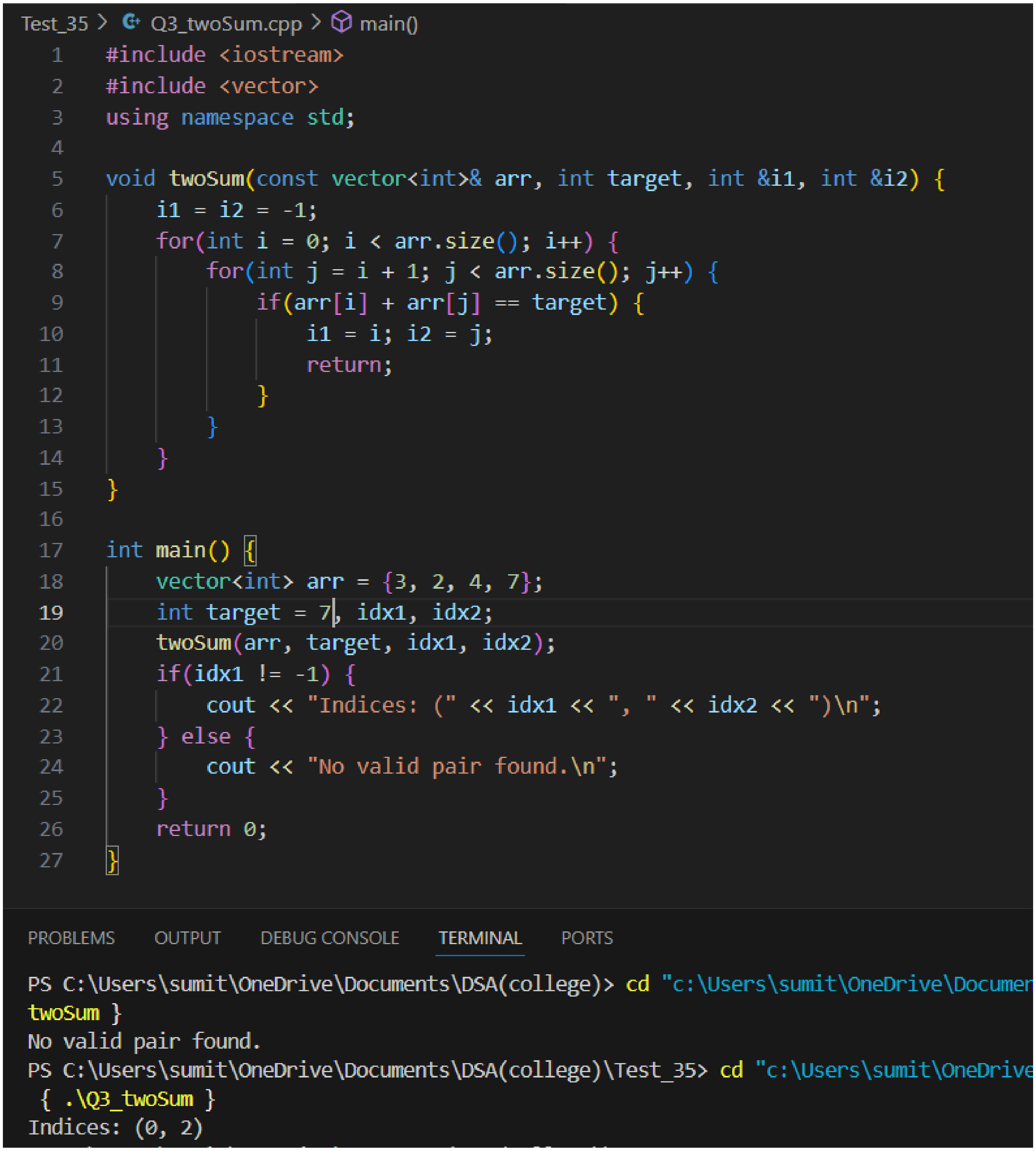
```cpp
#include <iostream>
using namespace std;

int maxEle(int arr[], int n) {
    int max = arr[0];
    for(int i=1; i<n; i++) {
        if(arr[i] > max) {
            max = arr[i];
        }
    }
    return max;
}

int main() {
    int arr[] = {2, 7, 1, 9, 5};
    int n = 5;

    cout << "maximum : " << maxEle(arr, n);

    return 0;
}
```

```
PS C:\Users\sumit\OneDrive\Documents\DSA(college)> cd "c:\Users\sumit\OneDrive\Documents\DSA(coll
maximum : 9
PS C:\Users\sumit\OneDrive\Documents\DSA(college)\Test_35>
```



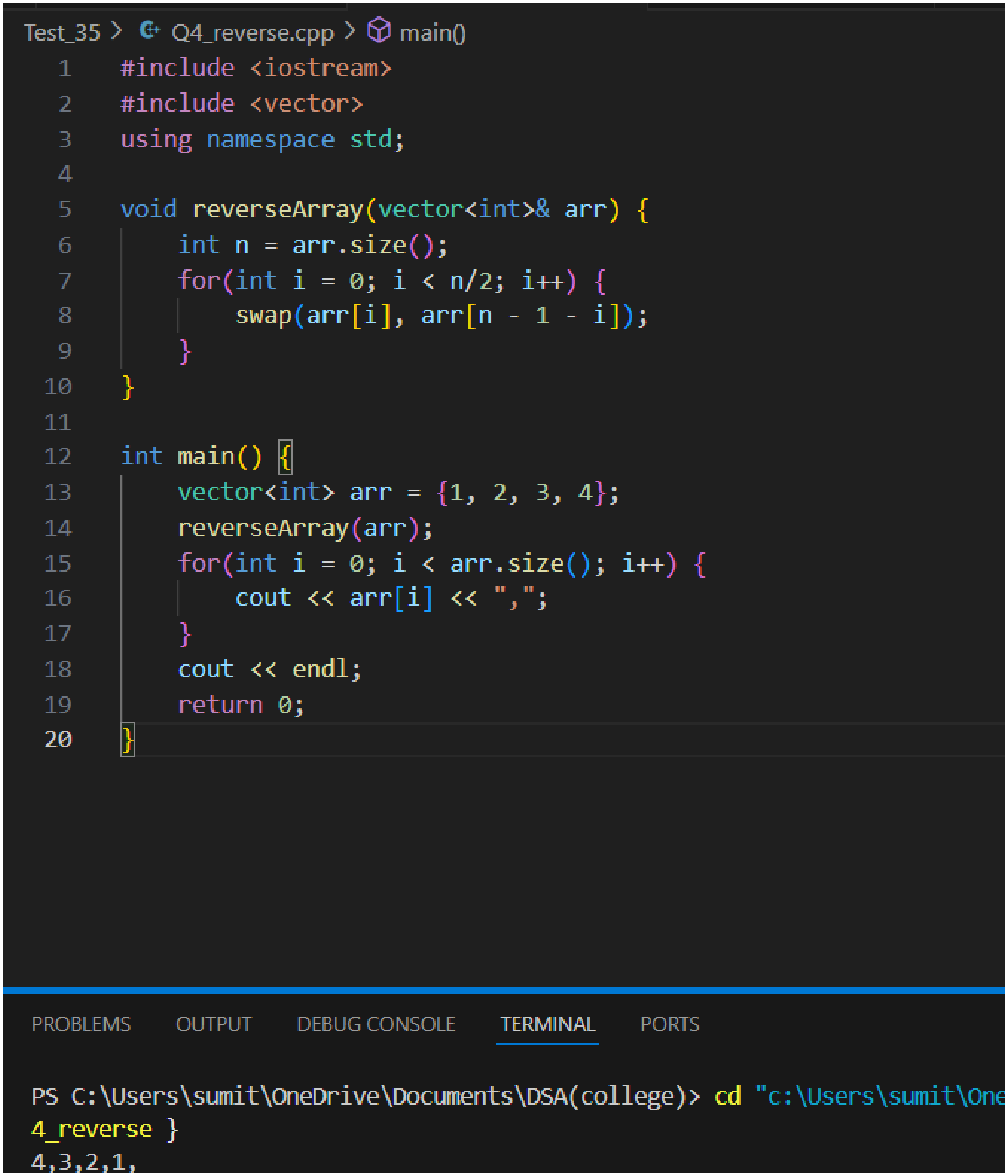
```cpp
#include <iostream>
using namespace std;

int missingNo(int arr[], int n) {
    int sum = n * (n+1)/2;
    int arrSum = 0;
    for(int i=0; i<n; i++) {
        arrSum += arr[i];
    }

    return sum - arrSum;
}

int main() {
    int arr[] = {0, 1, 2, 4, 5};
    int n = 5;

    cout << missingNo(arr, n) << endl;

    return 0;
}
```

```
PS C:\Users\sumit\OneDrive\Documents\DSA(college)> cd "c:\Users\sumit\OneDrive\Doc
3
PS C:\Users\sumit\OneDrive\Documents\DSA(college)\Test_35>
```



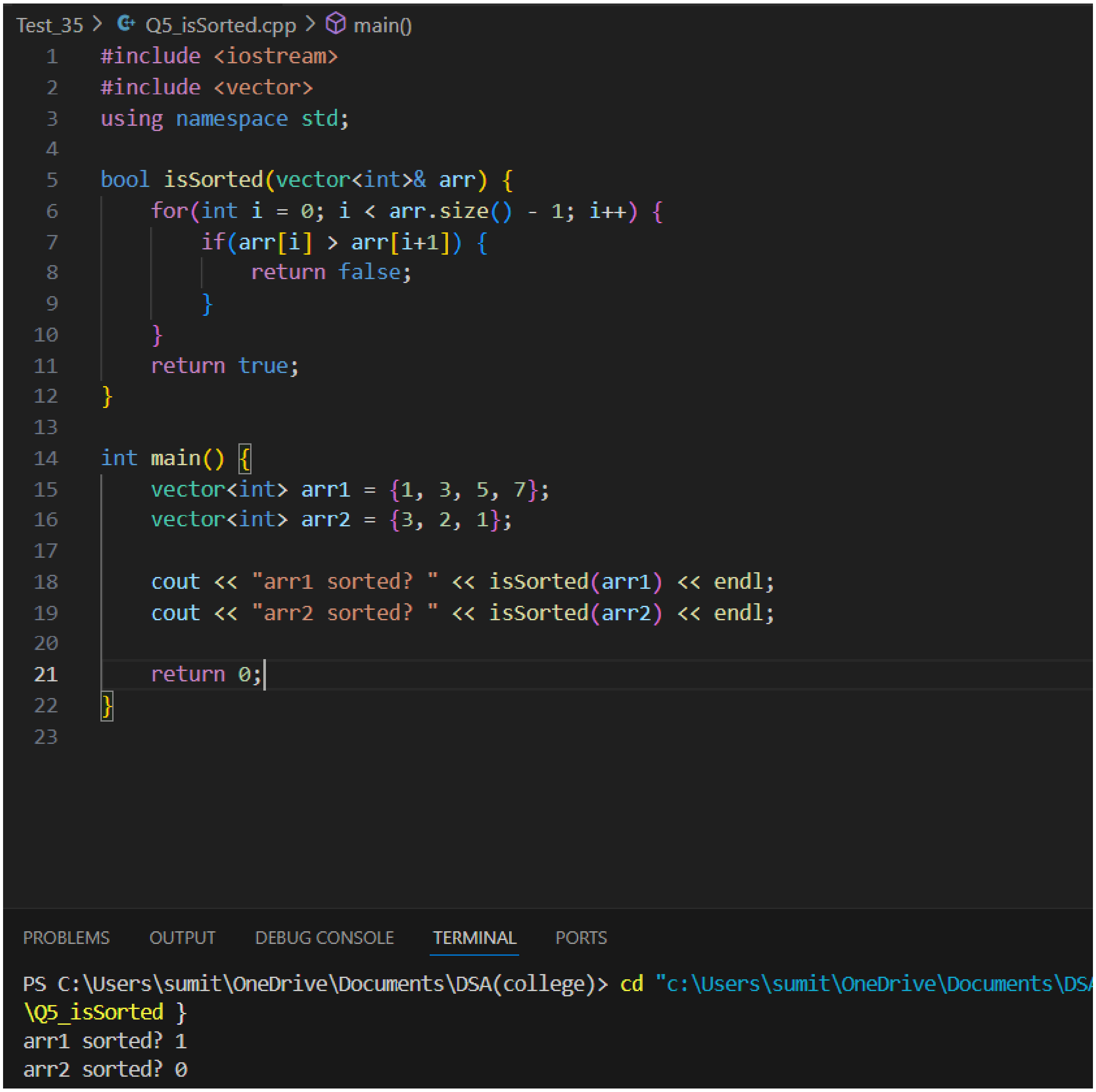
```cpp
#include <iostream>
#include <vector>
using namespace std;

void twoSum(const vector<int>& arr, int target, int &i1, int &i2) {
    i1 = i2 = -1;
    for(int i = 0; i < arr.size(); i++) {
        for(int j = i + 1; j < arr.size(); j++) {
            if(arr[i] + arr[j] == target) {
                i1 = i; i2 = j;
                return;
            }
        }
    }
}

int main() {
    vector<int> arr = {3, 2, 4, 7};
    int target = 7, idx1, idx2;
    twoSum(arr, target, idx1, idx2);
    if(idx1 != -1) {
        cout << "Indices: (" << idx1 << ", " << idx2 << ")\n";
    } else {
        cout << "No valid pair found.\n";
    }
    return 0;
}
```

```
PS C:\Users\sumit\OneDrive\Documents\DSA(college)> cd "c:\Users\sumit\OneDrive\Documen
twoSum }
No valid pair found.
PS C:\Users\sumit\OneDrive\Documents\DSA(college)\Test_35> cd "c:\Users\sumit\OneDrive
 { .\Q3_twoSum }
Indices: (0, 2)
```



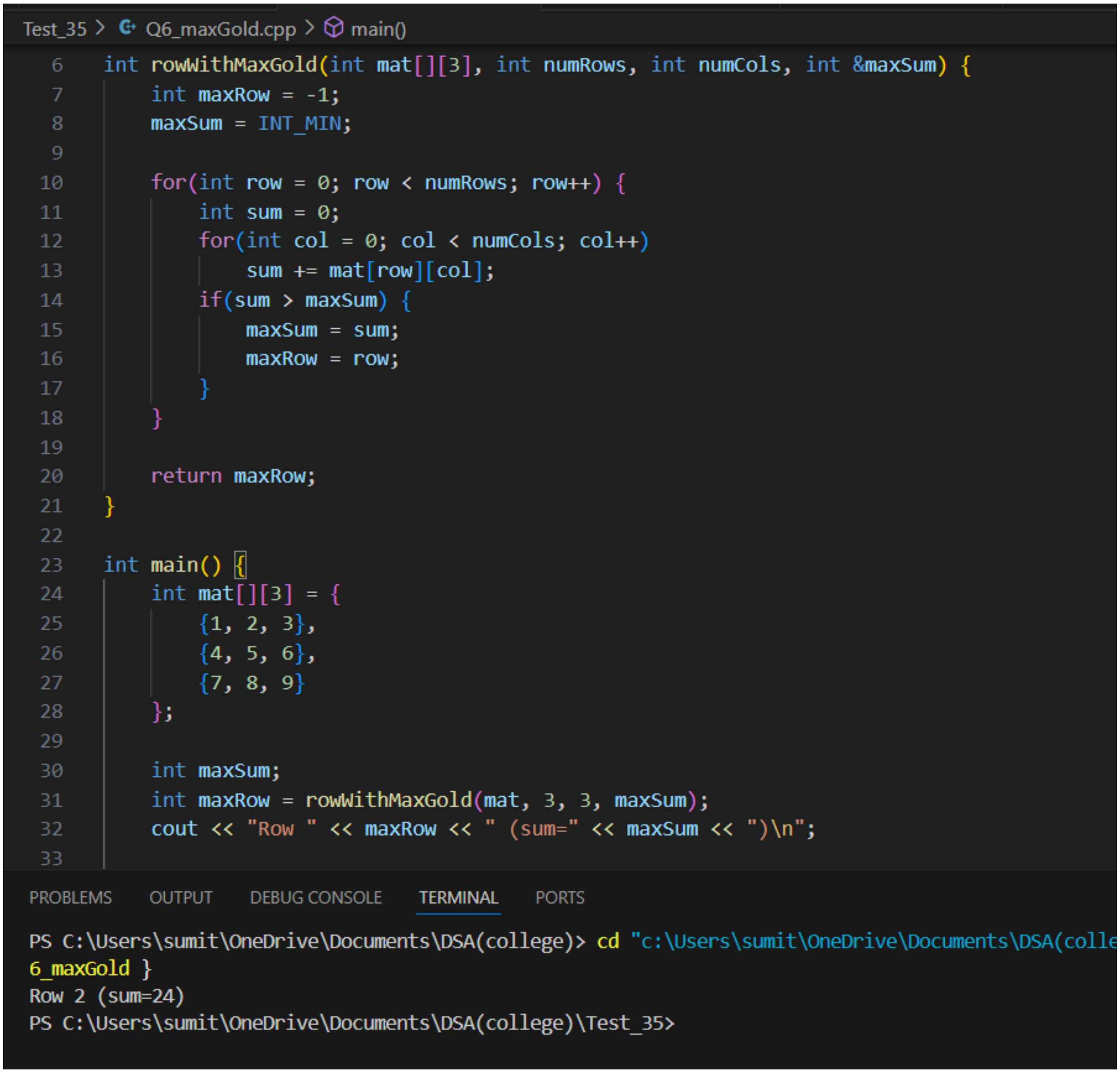
```cpp
#include <iostream>
#include <vector>
using namespace std;

void reverseArray(vector<int>& arr) {
    int n = arr.size();
    for(int i = 0; i < n/2; i++) {
        swap(arr[i], arr[n - 1 - i]);
    }
}

int main() {
    vector<int> arr = {1, 2, 3, 4};
    reverseArray(arr);
    for(int i = 0; i < arr.size(); i++) {
        cout << arr[i] << ",";
    }
    cout << endl;
    return 0;
}
```

```
PS C:\Users\sumit\OneDrive\Documents\DSA(college)> cd "c:\Users\sumit\One
4_reverse }
4,3,2,1,
```



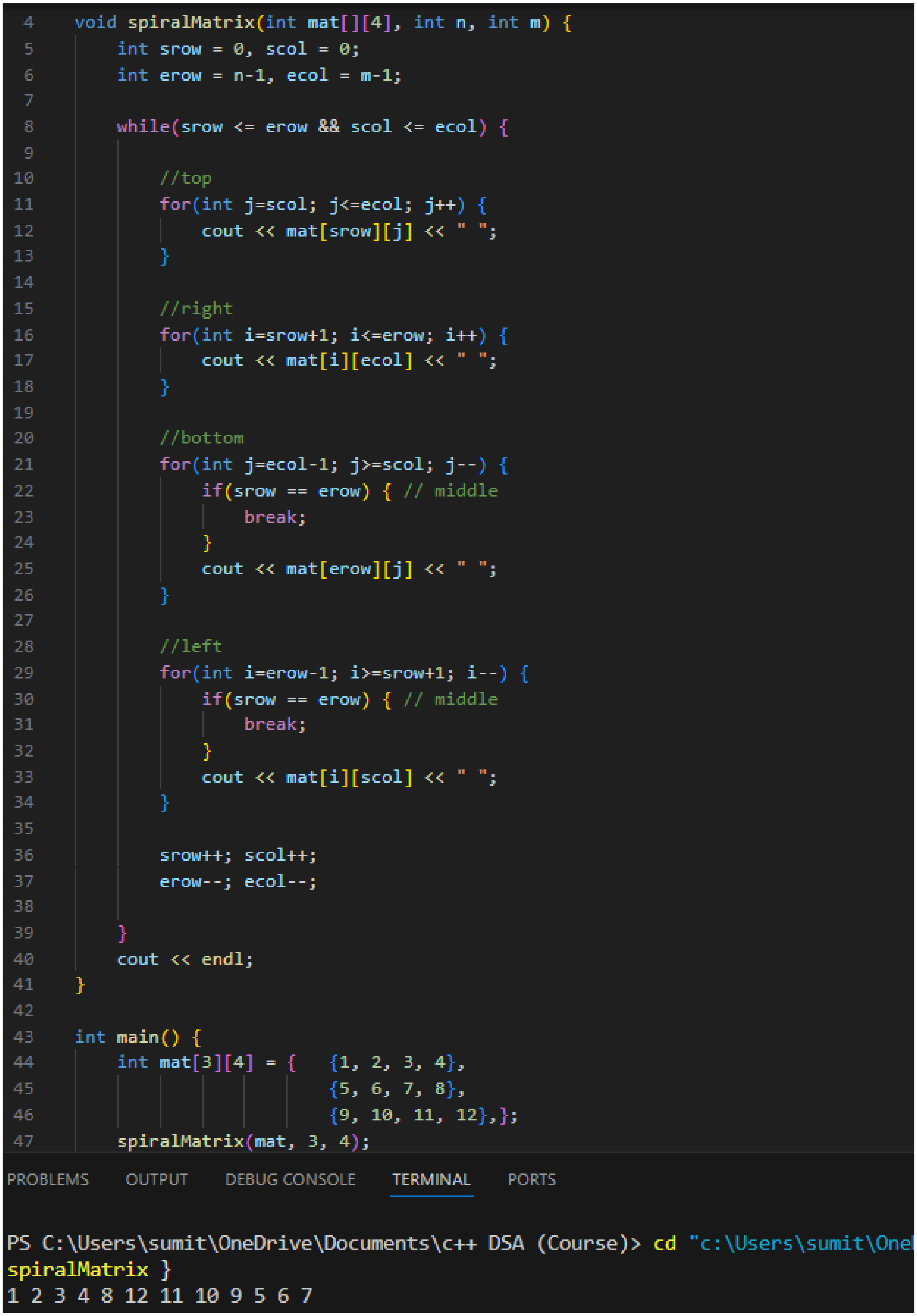
```cpp
#include <iostream>
#include <vector>
using namespace std;

bool isSorted(vector<int>& arr) {
    for(int i = 0; i < arr.size() - 1; i++) {
        if(arr[i] > arr[i+1]) {
            return false;
        }
    }
    return true;
}

int main() {
    vector<int> arr1 = {1, 3, 5, 7};
    vector<int> arr2 = {3, 2, 1};

    cout << "arr1 sorted? " << isSorted(arr1) << endl;
    cout << "arr2 sorted? " << isSorted(arr2) << endl;

    return 0;
}

```

```
PS C:\Users\sumit\OneDrive\Documents\DSA(college)> cd "c:\Users\sumit\OneDrive\Documents\DSA
\Q5_isSorted }
arr1 sorted? 1
arr2 sorted? 0
```



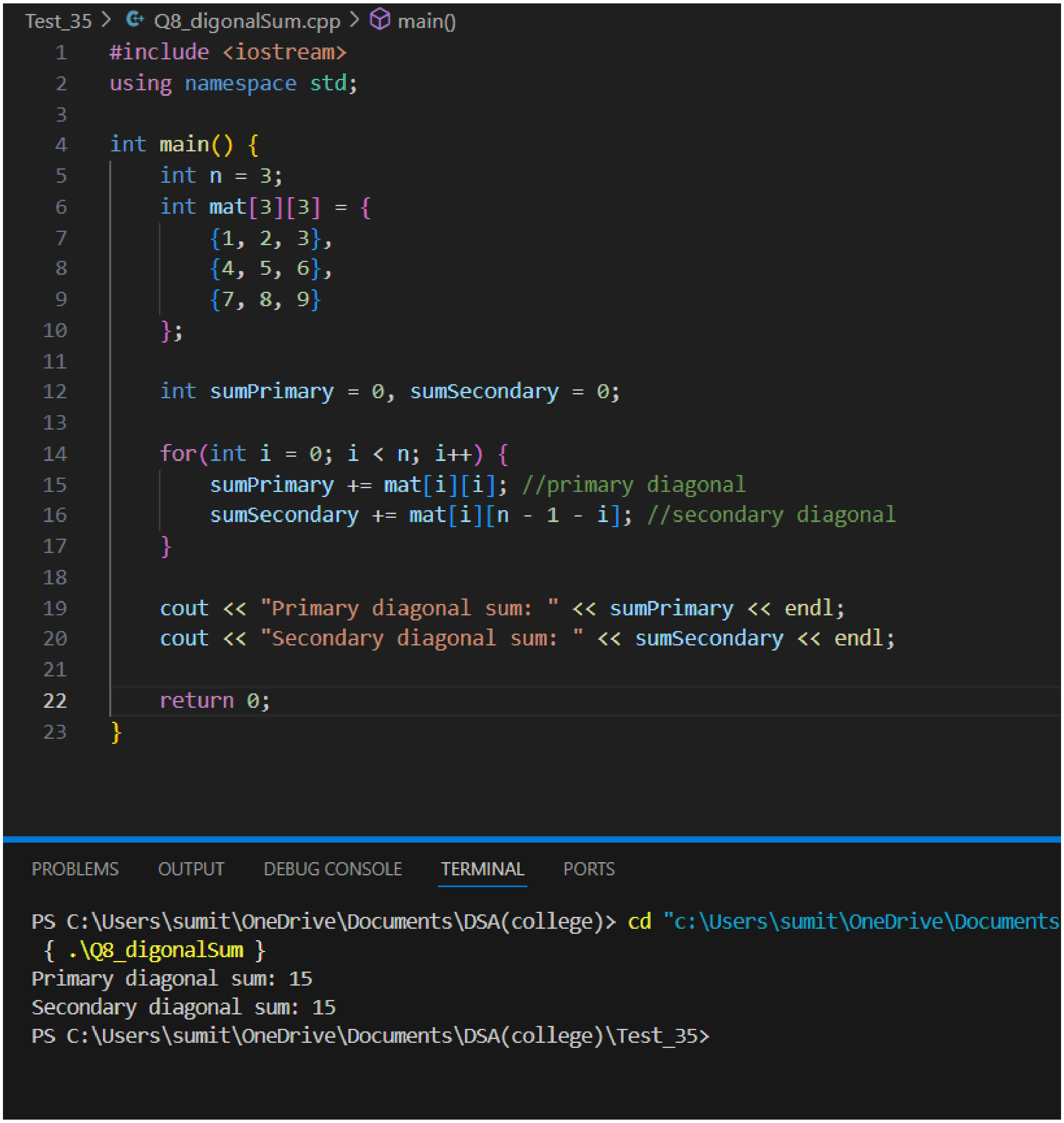
```cpp
int rowWithMaxGold(int mat[][3], int numRows, int numCols, int &maxSum) {
    int maxRow = -1;
    maxSum = INT_MIN;

    for(int row = 0; row < numRows; row++) {
        int sum = 0;
        for(int col = 0; col < numCols; col++)
            sum += mat[row][col];
        if(sum > maxSum) {
            maxSum = sum;
            maxRow = row;
        }
    }

    return maxRow;
}

int main() {
    int mat[][3] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

    int maxSum;
    int maxRow = rowWithMaxGold(mat, 3, 3, maxSum);
    cout << "Row " << maxRow << " (sum=" << maxSum << ")\n";

```

```
PS C:\Users\sumit\OneDrive\Documents\DSA(college)> cd "c:\Users\sumit\OneDrive\Documents\DSA(colle
6_maxGold }
Row 2 (sum=24)
PS C:\Users\sumit\OneDrive\Documents\DSA(college)\Test_35>
```



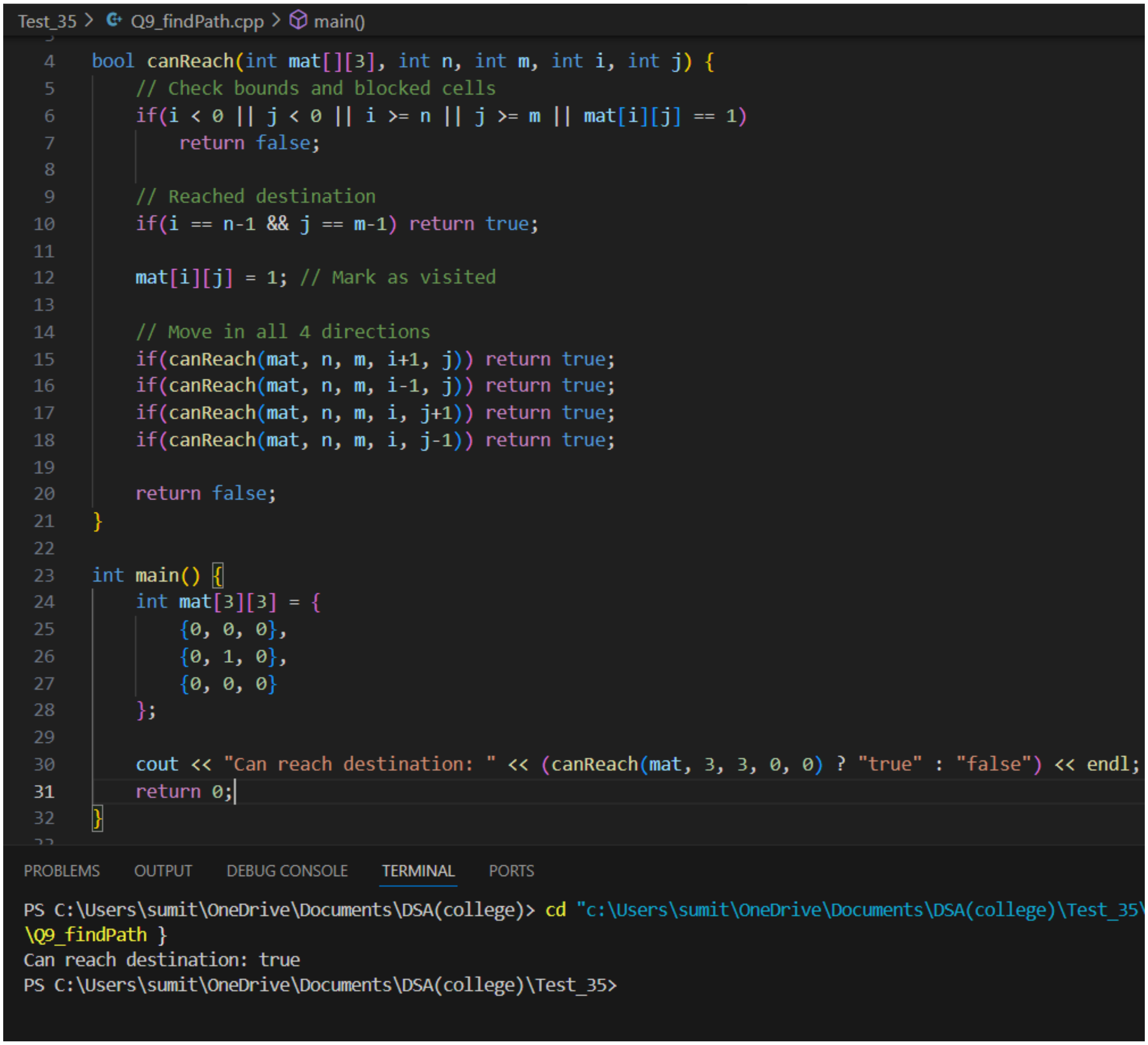
```cpp
void spiralMatrix(int mat[][4], int n, int m) {
    int srow = 0, scol = 0;
    int erow = n-1, ecol = m-1;

    while(srow <= erow && scol <= ecol) {

        //top
        for(int j=scol; j<=ecol; j++) {
            cout << mat[srow][j] << " ";
        }

        //right
        for(int i=srow+1; i<=erow; i++) {
            cout << mat[i][ecol] << " ";
        }

        //bottom
        for(int j=ecol-1; j>=scol; j--) {
            if(srow == erow) { // middle
                break;
            }
            cout << mat[erow][j] << " ";
        }

        //left
        for(int i=erow-1; i>=srow+1; i--) {
            if(srow == erow) { // middle
                break;
            }
            cout << mat[i][scol] << " ";
        }

        srow++; scol++;
        erow--; ecol--;

    }
    cout << endl;
}

int main() {
    int mat[3][4] = {   {1, 2, 3, 4},
                        {5, 6, 7, 8},
                        {9, 10, 11, 12},};
    spiralMatrix(mat, 3, 4);
```

```
PS C:\Users\sumit\OneDrive\Documents\c++ DSA (Course)> cd "c:\Users\sumit\One
spiralMatrix }
1 2 3 4 8 12 11 10 9 5 6 7
```



```cpp
#include <iostream>
using namespace std;

int main() {
    int n = 3;
    int mat[3][3] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

    int sumPrimary = 0, sumSecondary = 0;

    for(int i = 0; i < n; i++) {
        sumPrimary += mat[i][i]; //primary diagonal
        sumSecondary += mat[i][n - 1 - i]; //secondary diagonal
    }

    cout << "Primary diagonal sum: " << sumPrimary << endl;
    cout << "Secondary diagonal sum: " << sumSecondary << endl;

    return 0;
}
```

```
PS C:\Users\sumit\OneDrive\Documents\DSA(college)> cd "c:\Users\sumit\OneDrive\Documents
 { .\Q8_digonalSum }
Primary diagonal sum: 15
Secondary diagonal sum: 15
PS C:\Users\sumit\OneDrive\Documents\DSA(college)\Test_35>
```



```cpp
bool canReach(int mat[][3], int n, int m, int i, int j) {
    // Check bounds and blocked cells
    if(i < 0 || j < 0 || i >= n || j >= m || mat[i][j] == 1)
        return false;

    // Reached destination
    if(i == n-1 && j == m-1) return true;

    mat[i][j] = 1; // Mark as visited

    // Move in all 4 directions
    if(canReach(mat, n, m, i+1, j)) return true;
    if(canReach(mat, n, m, i-1, j)) return true;
    if(canReach(mat, n, m, i, j+1)) return true;
    if(canReach(mat, n, m, i, j-1)) return true;

    return false;
}

int main() {
    int mat[3][3] = {
        {0, 0, 0},
        {0, 1, 0},
        {0, 0, 0}
    };

    cout << "Can reach destination: " << (canReach(mat, 3, 3, 0, 0) ? "true" : "false") << endl;
    return 0;
}
```

```
PS C:\Users\sumit\OneDrive\Documents\DSA(college)> cd "c:\Users\sumit\OneDrive\Documents\DSA(college)\Test_35\
\Q9_findPath }
Can reach destination: true
PS C:\Users\sumit\OneDrive\Documents\DSA(college)\Test_35>
```