Xi'an Jiaotong-Liverpool University
西交利物浦大學

XJTLU Entrepreneur College (Taicang) Cover Sheet

Module code and Title	**DTS102TC Programming with C++**
School Title	**School of Artificial Intelligence and Advanced Computing**
Assignment Title	**Coursework 1 (CW 1)**
Submission Deadline	**5 pm China time (UTC+8 Beijing) on** Fri. 20th. Oct. 2023
Final Word Count	**N/A**
If you agree to let the university use your work anonymously for teaching and learning purposes, please type **"yes"** here.	**Yes**

I certify that I have read and understood the University's Policy for dealing with Plagiarism, Collusion and the Fabrication of Data (available on Learning Mall Online). With reference to this policy, I certify that:

- My work does not contain any instances of plagiarism and/or collusion.
- My work does not contain any fabricated data.

By uploading my assignment onto Learning Mall Online, I formally declare that all of the above information is true to the best of my knowledge and belief.

Scoring – For Tutor Use	
Student ID	**2254164**

Stage of Marking	**Marker Code**	**Learning Outcomes Achieved （F/P/M/D） (please modify as appropriate)**			**Final Score**
		A	**B**	**C**	
1st Marker – red pen					
Moderation – green pen	**IM Initials**	The original mark has been accepted by the moderator (please circle as appropriate):			Y / N
		Data entry and score calculation have been checked by another tutor (please circle):			Y
2nd Marker if needed – green pen					

<table>
<tr><td colspan="3">For Academic Office Use</td><td colspan="2">Possible Academic Infringement (please tick as appropriate)</td></tr>
<tr><td>Date Received</td><td>Days late</td><td>Late Penalty</td><td>☐ Category A</td><td rowspan="5">Total Academic Infringement Penalty (A,B, C, D, E, Please modify where necessary)
_____</td></tr>
<tr><td rowspan="4"></td><td rowspan="4"></td><td rowspan="4"></td><td>☐ Category B</td></tr>
<tr><td>☐ Category C</td></tr>
<tr><td>☐ Category D</td></tr>
<tr><td>☐ Category E</td></tr>
</table>

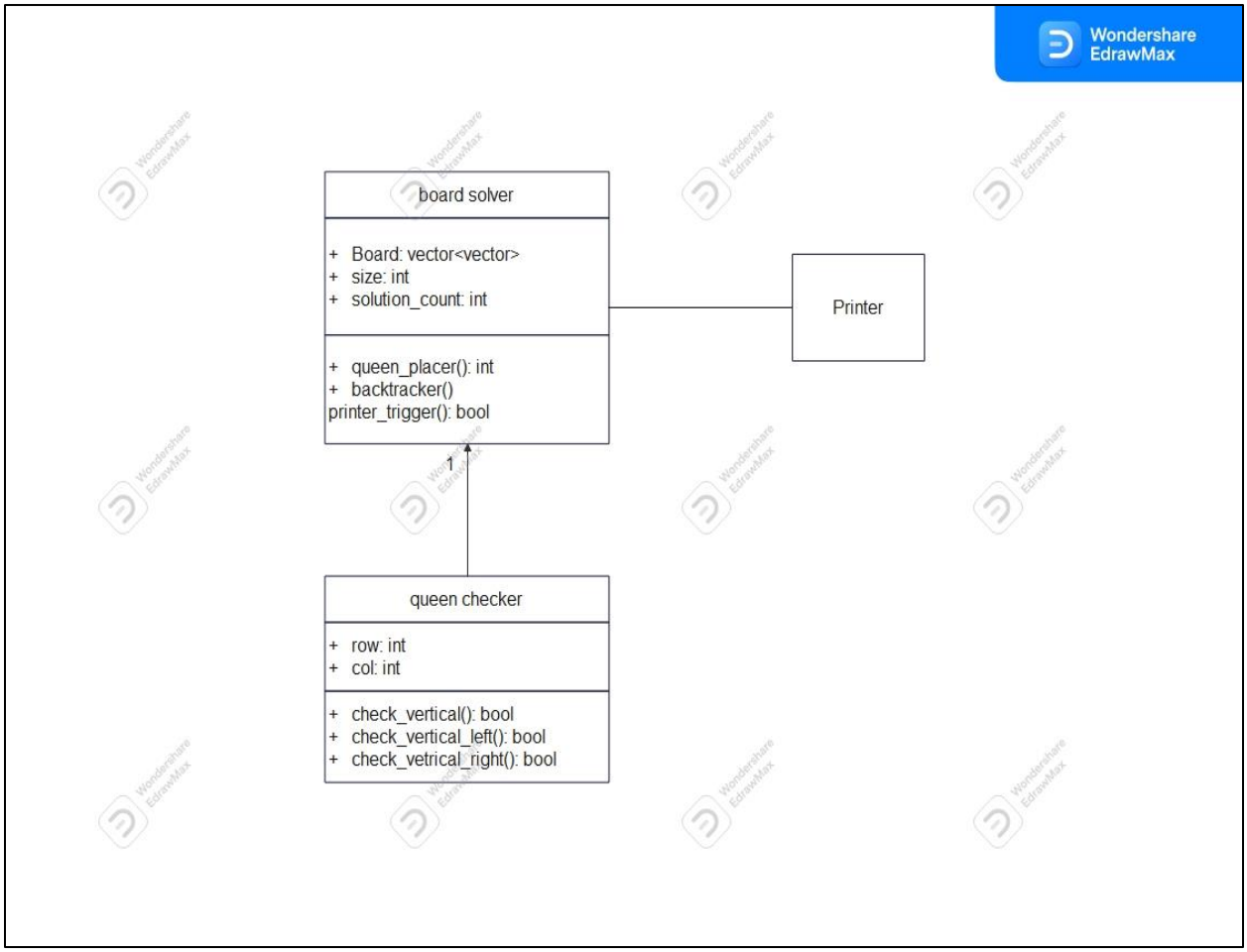
Students

The assignment must be typed in MS Word and converted to a PDF document. The document must be submitted via Learning Mall Online to the correct drop box. Only electronic submission is accepted and no hard copy submission.

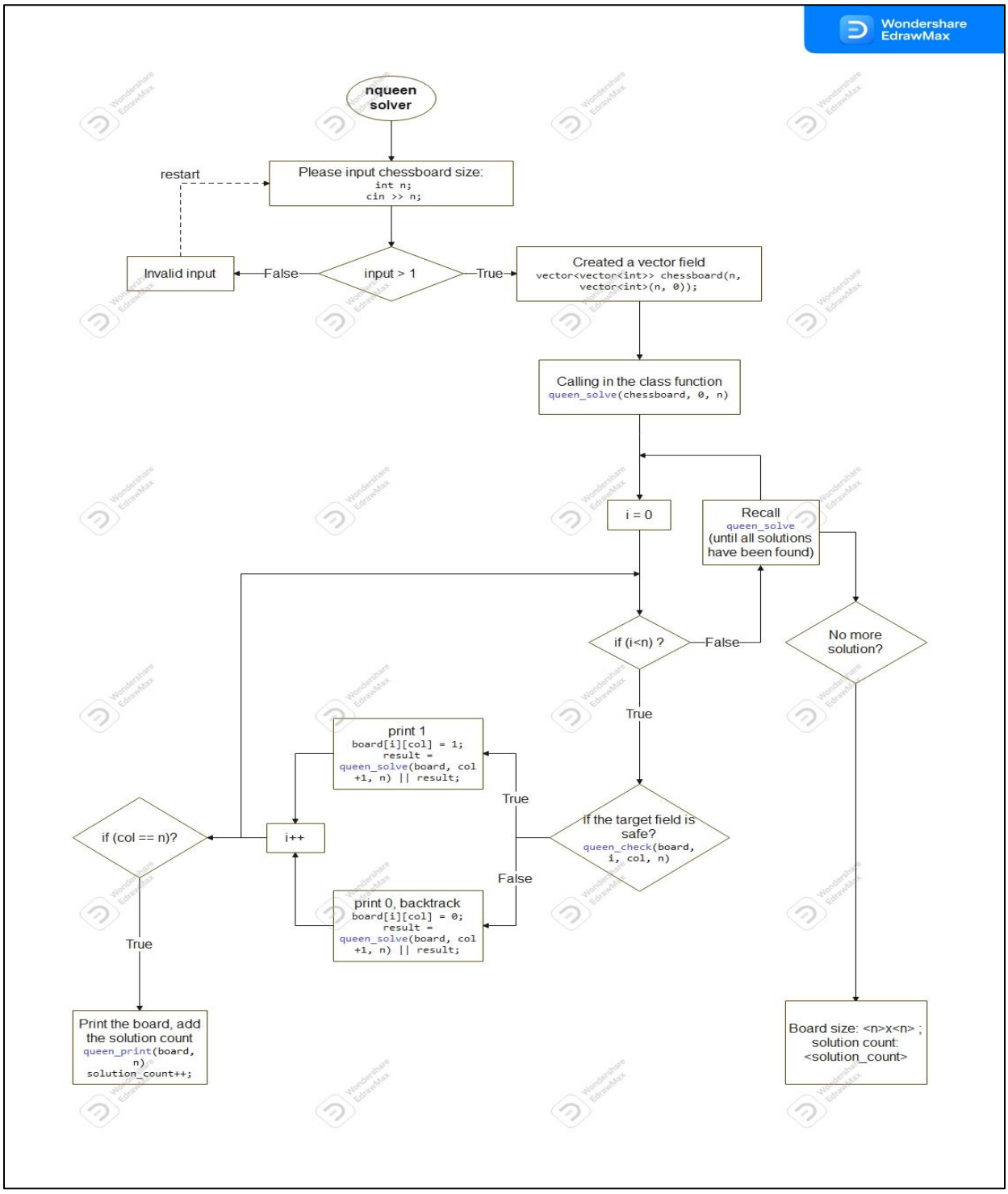
All students must download their file and check that it is viewable after submission. Documents may become corrupted during uploading (e.g., due to slow internet connections). However, students themselves are responsible for submitting a functional and correct file for assessments.

I. Program Design

1.1 Data Structure



1.2 System Process



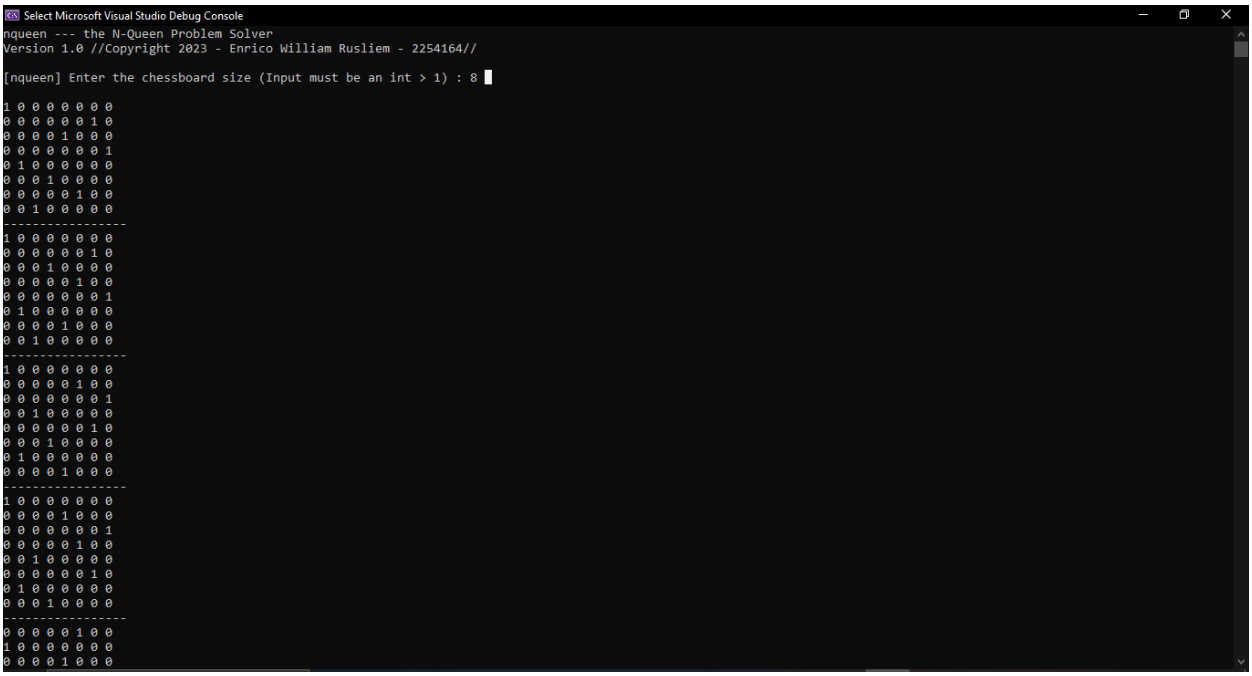
II. Program Implementation

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

int solution_count = 0;//notice solution_count is unbounded, purpose: we can call
solution_count on 2 different functions

void queen_print(vector<vector<int>> board, int n)
//function to print out the board
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            cout << board[i][j] << " ";//taking the vector 'board' content out
        }
        cout << '\n';
    }
    cout << "-----------------\n";
}

bool queen_check(vector<vector<int>> board, int row, int col, int n)
//set of loops to implement the queen function (checks run along the y-axis)
{
    //vertical check
    for (int j = 0; j < col; j++)
    {
        if (board[row][j])
        {
            return false;
        }
    }
    //vertical-left diagonal check
    for (int i = row, j = col; i >= 0 && j >= 0; i--, j--)
    {
        if (board[i][j])
        {
            return false;
        }
    }
    //vertical-right diagonal check
    for (int i = row, j = col; i < n && j >= 0; i++, j--)
    {
        if (board[i][j])
        {
            return false;
        }
    }
    //if the given position passed all the checks,
    return true;
}
```

```
bool queen_solve(vector<vector<int>> board, int col, int n)
//this function will place the queens while applying backtracking algorithm
{
    bool result = true;
    if (col == n)
    {
        queen_print(board, n);//prints the solution board every time the tracer
reaches the end
        solution_count++;
        return result;
    }
    for (int i = 0; i < n; i++)//iterator for placing the queens
    {
        if (queen_check(board, i, col, n))//backtracker
        {
            board[i][col] = 1;
            result = queen_solve(board, col+1, n) || result;
            board[i][col] = 0;
        }
    }
    return result;
}

int main()
{
    cout << "nqueen --- the N-Queen Problem Solver\nVersion 1.0 //Copyright 2023 -
Enrico William Rusliem - 2254164//\n\n";
    cout << "[nqueen] Enter the chessboard size (Input must be an int > 1) : ";
    int n;
    cin >> n;
    while (true)//filtering user input, preventing errors caused by invalid input
        //note: all doubles input will be rounded down
    {
        if (n > 1)
            break;
        else
            cout << "[nqueen] Invalid input, please enter an integer > 1\n";
        return 0;
    }
    cout << '\n';
    vector<vector<int>> chessboard(n, vector<int>(n, 0)); //formulating the nxn
chessboard
    if (queen_solve(chessboard, 0, n)) //print out the iteration summary
        cout << "\n[nqueen] Board size: " << n << "x" << n << " ; solution count: "
<< solution_count << '\n';
      return 0;
}
```

III. Program Test

3.1 8-queens problem console interface



3.2 4-queens problem console interface

IV. Program Analysis

4.1 Complexity Analysis of the Eight Queens Problem

- Background: The Eight Queen Problem is a classic puzzle of how to successfully place eight queen pieces on an 8 x 8 chessboard (simplification of n-queens problem) with a condition such that every queen must avoid all possible horizontal, vertical, and diagonal intersections with each other.
- Based on the length of the source code (close to 100) and the difficulty of the required research to complete the task, the complexity of the Eight Queens problem could be categorized as moderate-high.
- There are multiple ways to solve the Eight Queen Problems. The 'Brute Force' approach, the algorithm that would run to check every possible board combination, is considered to be the easiest solution. However, the technique is time-consuming for wider boards (inefficient). Therefore, an alternative solution is to be considered
- Given the difficulties posed by other algorithms, the backtracking approach becomes a viable basic choice. In general, the algorithm will place queens orderly in quick succession with the help of a filter. This filter is a set of loops that implements the queen function which in return will decide whether the placement of the next queen is valid or not.
- The backtracking system appears to significantly reduce the number of explored possibilities, leading to a greater runtime efficiency.

4.2 Analysis of the Solution for N-Queen Problems

- Based on the checks run by the code, the problem of n queens on the nxn chessboard will yield results only if $n > 3$. Specifically, there are 2 solutions for a 4x4 chessboard and 92 solutions for an 8x8 chessboard. No possible solution was found at $1 < n \leq 3$. Important to note that $n = 1$ does not counted as a solution since one solitary cell defies the definition of a chessboard.