```
#include <bits/stdc++.h>
using namespace std;

vector<double> weights, values;

bool cmp(int a, int b) {
    return (values[a] / weights[a]) > (values[b] / weights[b]);
}

int main() {
    int n;
    double W;
    cin >> W >> n;

    weights.resize(n);
    values.resize(n);

    for (int i = 0; i < n; i++) {
        cin >> weights[i] >> values[i];
    }

    vector<int> idx(n);
    for (int i = 0; i < n; i++) idx[i] = i;

    sort(idx.begin(), idx.end(), cmp);

    double totalValue = 0.0, remaining = W;
    vector<double> taken(n, 0.0);

    for (int j = 0; j < n; j++) {
        int i = idx[j];
        if (remaining <= 0) break;

        if (weights[i] <= remaining) {
            taken[i] = 1.0;
            totalValue += values[i];
            remaining -= weights[i];
        } else {
            taken[i] = remaining / weights[i];
            totalValue += values[i] * taken[i];
            remaining = 0;
        }
    }

    for (int i = 0; i < n; i++) {
        cout << (i + 1) << "\t" << weights[i] << "\t" << values[i]
             << "\t" << taken[i] << "\n";
    }

    cout << totalValue << "\n";
}
//nqueen
#include <bits/stdc++.h>
using namespace std;



bool isSafe(vector<string> &board, int row, int col, int n) {

    for (int j = 0; j < n; j++) {
        if (board[row][j] == 'Q') return false;
    }
    for (int i = 0; i < n; i++) {
        if (board[i][col] == 'Q') return false;
    }

    for (int i = row - 1, j = col - 1; i >= 0 && j >= 0; i--, j--) {
        if (board[i][j] == 'Q') return false;
    }

    for (int i = row - 1, j = col + 1; i >= 0 && j < n; i--, j++) {
        if (board[i][j] == 'Q') return false;
    }

    for (int i = row + 1, j = col - 1; i < n && j >= 0; i++, j--) {
        if (board[i][j] == 'Q') return false;
    }

    for (int i = row + 1, j = col + 1; i < n && j < n; i++, j++) {
        if (board[i][j] == 'Q') return false;
    }
    return true;
}

void nQueens(vector<string> &board, int row, int n,
             vector<vector<int>> &ansIdx,
             vector<vector<string>> &ansBoard,
             vector<int> &current) {
    if (row == n) {
        ansIdx.push_back(current);
        ansBoard.push_back(board);
        return;
    }
    for (int j = 0; j < n; j++) {
        if (isSafe(board, row, j, n)) {
            board[row][j] = 'Q';
            current[row] = j + 1;
            nQueens(board, row + 1, n, ansIdx, ansBoard, current);
            board[row][j] = '.';
            current[row] = 0;
        }
    }
}

int main() {
    int n;
    cout << "Enter number of Queens: ";
    cin >> n;

    vector<string> board(n, string(n, '.'));
    vector<vector<int>> solutionsIdx;
    vector<vector<string>> solutionsBoard;
    vector<int> current(n, 0);

    nQueens(board, 0, n, solutionsIdx, solutionsBoard, current);

    if (solutionsIdx.empty()) {
        cout << "No possible Solutions\n";
        return 0;
    }

    for (int k = 0; k < (int)solutionsIdx.size(); k++) {

        cout << "Solution " << (k + 1) << " :\t[ ";
        for (int i = 0; i < n; i++) {
            cout << solutionsIdx[k][i] << " ";
        }
        cout << "]\n";



        for (int i = 0; i < n; i++) {
            cout << solutionsBoard[k][i] << "\n";
        }
        cout << "\n";
    }
    return 0;
}
```