```
//Implement application of array in sparse matrix to
//perform simple and fast transpose.

#include <iostream>
using namespace std;

struct SparseMatrix {
    int row;
    int col;
    int value;
};

SparseMatrix* transpose(SparseMatrix* mat, int size) {
    SparseMatrix* transposed = new SparseMatrix[size];
    for (int i = 0; i < size; i++) {
        transposed[i].col = mat[i].row;
        transposed[i].row = mat[i].col;
        transposed[i].value = mat[i].value;
    }
    return transposed;
}

int main() {
    SparseMatrix mat[5] = {{0, 0, 15}, {0, 3, 22}, {1, 1, 11}, {2, 2, 27}, {3, 1, 
17}};
    int size = sizeof(mat)/sizeof(mat[0]);
    SparseMatrix* transposed = transpose(mat, size);

    cout << "Original Matrix:" << endl;
    for (int i = 0; i < size; i++) {
        cout << "(" << mat[i].row << ", " << mat[i].col << ", " << mat[i].value << 
")" << endl;
    }

    cout << "Transposed Matrix:" << endl;
    for (int i = 0; i < size; i++) {
        cout << "(" << transposed[i].row << ", " << transposed[i].col << ", " << 
transposed[i].value << ")" << endl;
    }

    return 0;
}


output:-
Original Matrix:
(0, 0, 15)
(0, 3, 22)
(1, 1, 11)
(2, 2, 27)
(3, 1, 17)
Transposed Matrix:
(0, 0, 15)
(3, 0, 22)
(1, 1, 11)
(2, 2, 27)
(1, 3, 17)
```