P4 – Matrices and Vectors

Final

```
double getRandom(void) {
    //default_random_engine rndeng;                       //generator seed engine
    //uniform_real_distribution<double> distribution(1, 10);  //generator distribution

    return distribution(rndeng);

}

vector<double> fillArray(int arrSize) {
    vector<double> v;

    for(int j=0; j<arrSize; j++) {
        v.push_back(getRandom() * 1);

    }
    return v;
}
```

```
vector<double> multipMatrix(double E, vector<double> D, vector<double> L) {
    vector<double> v;

    for(int i=0; i<MAT_S; i++) {
        v.push_back(D[i] * L[i]);
    }

    for(int i=0; i<MAT_S - 1; i++) {
        v[i] += (E * L[i+1]);
        v[i+1] += (E * L[i]);
    }

    return v;
}
```

```cpp
void printMatrix(double E, vector<double> D) {

    for(int i=0; i<MAT_S; i++) {

        for(int j=0; j<MAT_S; j++) {

            if(i == j) {
                cout << D[i];
            }
            else if(((i - j) == 1) || ((j - i) == 1)) {
                cout << E;
            }
            else {
                cout << "0";
            }

            cout << "\t";
        }
        cout << "\n";
    }
}
```

```cpp
vector<double> TridiagonalSolve(double E, vector<double> D, vector<double> R) {
    vector<double> c(MAT_S);
    double id;

    int i;

    vector<double> L(MAT_S);

    for(int i=0; i<MAT_S; i++) {
        c[i] = E;
    }

    c[0] /= D[0];
    R[0] /= D[0];

    for(int i=1; i<MAT_S; i++) {
        id = D[i] - c[i - 1] * E;
        c[i] /= id;
        R[i] = (R[i] - R[i - 1] * E) / id;
    }

    L[MAT_S - 1] = R[MAT_S - 1];

    for(int i=MAT_S - 2; i>=0; i--) {
        L[i] = R[i] - c[i] * L[i + 1];
    }

    return L;
}
```

```cpp
int main() {
    double E = 0.0;
    vector<double> D = {};
    vector<double> R = {};
    vector<double> L = {};
    vector<double> Lr(MAT_S);



    //generate T matrix
    E = getRandom();
    D = fillArray(MAT_S);

    //generate matrix L
    L = fillArray(MAT_S);

    //calculate matrix R
    R = multipMatrix(E, D, L);


    /*
    cout << "T:\n";
    cout << "\tE: " << E << endl;

    cout << "\tD: ";
    printArray(D);*/
    cout << "T: \n";
    printMatrix(E, D);

    cout << "\nL: ";
    printArray(L);

    cout << "\n= R: ";
    printArray(R);


    //reverse solve
    Lr = TridiagonalSolve(E, D, R);

    cout << "\n\nL: ";
    printArray(Lr);

    return 0;
}
```

```
D:\2020+21\ELEC1204 - Advanced Programming\P4\src\Final>main.exe
T:
5.12785 2.18384 0       0
2.18384 2.97063 2.18384 0
0       2.18384 7.10978 2.18384
0       0       2.18384 9.41224

L: 5.67475 1.31115 5.7673 1.06928

= R: 31.9626 28.8825 46.2027 22.6592


L: 5.67475 1.31115 5.7673 1.06928

D:\2020+21\ELEC1204 - Advanced Programming\P4\src\Final>
```