

Soft copy: 4/5/2020

Hard copy: 4/5/2020

*****Main*****

Step 0: pointSet \leftarrow given, K \leftarrow given
changes \leftarrow 0

Step 1: randomly select k points from pointSet.
(The selected k points will be the initial K centroids.)

Step 2: For each point, p, in pointSet, compute distance from p to each centroid
(K of them). So, we have K distances: d₁, d₂, ..., d_K, associate with each d_i is the
label of the centroid.

Step 3: minLabel \leftarrow the label of min(d₂, ..., d_K)

Step 4: if p's label != minLabel
 P's label \leftarrow minLabel
 changes ++

Step 5: repeat step 2 to step 4 until all points in pointSet are processed

Step 6: if changes > 2
 Compute centroid for each group,...
 changes \leftarrow 0

Step 7: repeat step 2 to step 6 until changeFlag <= 2

```

SOURCE CODE:
#include <iostream>
#include <string>
#include <fstream>
#include <cstdlib>
#include <cmath>
using namespace std;

// Data structures
class Point{
public:
    double Xcoord;
    double Ycoord;
    int Label = 0;
    double Distance = 99999.00;
};

// Prototypes
void loadPointSet(ifstream& inFile, Point* pointSet);
void PlotDisplayAry(Point* pointSet, int** displayAry, int numPts);
void PrettyPrint(int** displayAry, ofstream& outFile, int iteration, int rows, int cols);
void kMeansClustering(Point* pointSet, int K, Point* KcentroidAry, int** displayAry, ofstream& outFile1, ofstream& outFile2, int numPts, int numRows, int numCols, int changes);

void selectKcentroids(Point* pointSet, int K, Point* KcentroidAry, int numPts);
int DistanceMinLabel(Point& pt, Point* KcentroidAry, double& minDist, int K);
double computeDist(Point pt, Point centroid);
void computeCentroids(Point* pointSet, Point* KcentroidAry, int K, int numPts);
void PrintResult(Point* pointSet, ofstream& outFile1, int numRows, int numCols, int numPts);

int main(int argc, char* argv[]){
    ifstream inFile(argv[1]);
    int K = stoi(argv[2]);
    ofstream outFile1(argv[3]);
    ofstream outFile2(argv[4]);

    // Variables
    // 1
    int numRows, numCols, numPts, changes = 0;
    inFile >> numRows;
    inFile >> numCols;
    inFile >> numPts;

    int** displayAry = new int*[numRows];
    for(int i = 0; i < numRows; i++){
        displayAry[i] = new int[numCols];
    }

    Point* pointSet = new Point[numPts];
    Point* KcentroidAry = new Point[K+1];
    for(int i = 0; i < K+1; i++){
        KcentroidAry[i].Distance = 0.0;
    }

    loadPointSet(inFile, pointSet);
    kMeansClustering(pointSet, K, KcentroidAry, displayAry, outFile1, outFile2, numPts, numRows, numCols, changes);
    PrintResult(pointSet, outFile2, numRows, numCols, numPts);

    // Closing Files
    inFile.close();
    outFile1.close();
    outFile2.close();
}

```

```

// Functions

void PrintResult(Point* pointSet, ofstream& outFile2, int numRows, int numCols, int numPts) {
    outFile2 << numRows << " " << numCols << endl;
    outFile2 << numPts << endl;

    for(int i = 0; i < numPts; i++){
        outFile2 << pointSet[i].Xcoord << " " << pointSet[i].Ycoord << " " << pointSet[i].Label
    << endl;
    }
}

void computeCentroids(Point* pointSet, Point* KcentroidAry, int K, int numPts){
    double sumX[K+1];
    double sumY[K+1];
    int totalPt[K+1];
    for(int i = 0; i < K+1; i++){
        sumX[i] = 0.0;
        sumY[i] = 0.0;
        totalPt[i] = 0;
    }

    // 1
    int index = 0;
    int label = 0;

    while(index < numPts){
        //2
        label = pointSet[index].Label;
        sumX[label] += pointSet[index].Xcoord;
        sumY[label] += pointSet[index].Ycoord;
        totalPt[label]++;
        // 3
        index++;
    } // 4

    // 5
    label = 1;

    while(label <= K){ // <= K because we need K elements and we start label at 1
        // 6
        if(totalPt[label] > 0.0){
            KcentroidAry[label].Xcoord = (sumX[label]/totalPt[label]);
            KcentroidAry[label].Ycoord = (sumY[label]/totalPt[label]);
        }
        // 7
        label++;
    }
}

double computeDist(Point pt, Point centroid){
    double result = sqrt( pow( (pt.Xcoord-centroid.Xcoord), 2) + pow( (pt.Ycoord-
    centroid.Ycoord), 2) );
    return result;
}

int DistanceMinLabel(Point& pt, Point* KcentroidAry, double& minDist, int K){
    // minDist = 99999.0;
    int minLabel = 0;

    int label = 1;

```

```

while (label <= K){
    // Point whichCentroid = KcentroidAry[label];

    double dist = computeDist(pt, KcentroidAry[label]);

    if(dist < minDist){
        minLabel = label;
        minDist = dist;
    }
    label++;
}
return minLabel;
}

void selectKcentroids(Point* pointSet, int K, Point* KcentroidAry, int numPts){
    // creating data structure
    bool checkedPoints[numPts];
    for(int i = 0; i < numPts; i++){ checkedPoints[i] = false;}

    int index = rand() % numPts;
    // bool repeatYN = false;
    int Kcnt = 0;
    while(Kcnt < K){
        while(checkedPoints[index]){
            index = rand() % numPts;
            // repeatYN = checkedPoints[index];
        }
        Kcnt++;
        KcentroidAry[Kcnt].Xcoord = pointSet[index].Xcoord;
        KcentroidAry[Kcnt].Ycoord = pointSet[index].Ycoord;
        KcentroidAry[Kcnt].Label = Kcnt;
        KcentroidAry[Kcnt].Distance = 0.0;

        checkedPoints[index] = true;
    }
}

void kMeansClustering(Point* pointSet, int K, Point* KcentroidAry, int** displayAry, ofstream& outFile1, ofstream& outFile2, int numPts, int numRows, int numCols, int changes){
    // 0
    int iteration = 0;
    int index = 0;
    int doOnce = 1;

    // 1
    selectKcentroids(pointSet, K, KcentroidAry, numPts);

    while (iteration < 25 || doOnce > 0) { // 2 - 9
        doOnce--;
        // 2
        index = 0;
        iteration++;

        while (index < numPts){ // 4 - 7
            // 4
            Point pt = pointSet[index];
            double minDist = pointSet[index].Distance;

            // 5
            int minLabel = DistanceMinLabel(pt, KcentroidAry, minDist, K);
            // pointSet[index].Distance = minDist;

            // 6
            if(pointSet[index].Label != minLabel){

```

```

        pointSet[index].Label = minLabel;
        pointSet[index].Distance = minDist;
        changes++;
    }
    // 7
    index++;
} // 8 loop

// 9
if (changes > 2) {
    computeCentroids(pointSet, KcentroidAry, K, numPts);
    changes = 0;
}
// // 3

PlotDisplayAry(pointSet, displayAry, numPts);
PrettyPrint(displayAry, outFile1, iteration, numRows, numCols);

} // 10 loop
}

void PrettyPrint(int** displayAry, ofstream& outFile1, int iteration, int rows, int cols){
    outFile1 << "\t***** Result of iteration " << iteration << " *****" <<
endl;
    for(int i = 0; i < rows; i++){
        for(int j = 0; j < cols; j++){
            if(displayAry[i][j] > 0 ){
                outFile1 << displayAry[i][j];
            } else {
                outFile1 << " ";
            }
        }
        outFile1 << endl;
    }
}

void PlotDisplayAry(Point* pointSet, int** displayAry, int numPts){
    // X is the row
    // Y is the col
    for(int i = 0 ; i < numPts; i++){
        displayAry
        [(int) pointSet[i].Xcoord]
        [(int) pointSet[i].Ycoord] = pointSet[i].Label;
    }
}

void loadPointSet(ifstream& inFile, Point* pointSet){
    int index = 0;
    int x,y;
    while(!inFile.eof()){
        inFile >> x;
        inFile >> y;

        pointSet[index].Xcoord = (double) x;
        pointSet[index].Ycoord = (double) y;
        pointSet[index].Label = 0;
        pointSet[index].Distance = 99999.0;

        index++;
    }
}

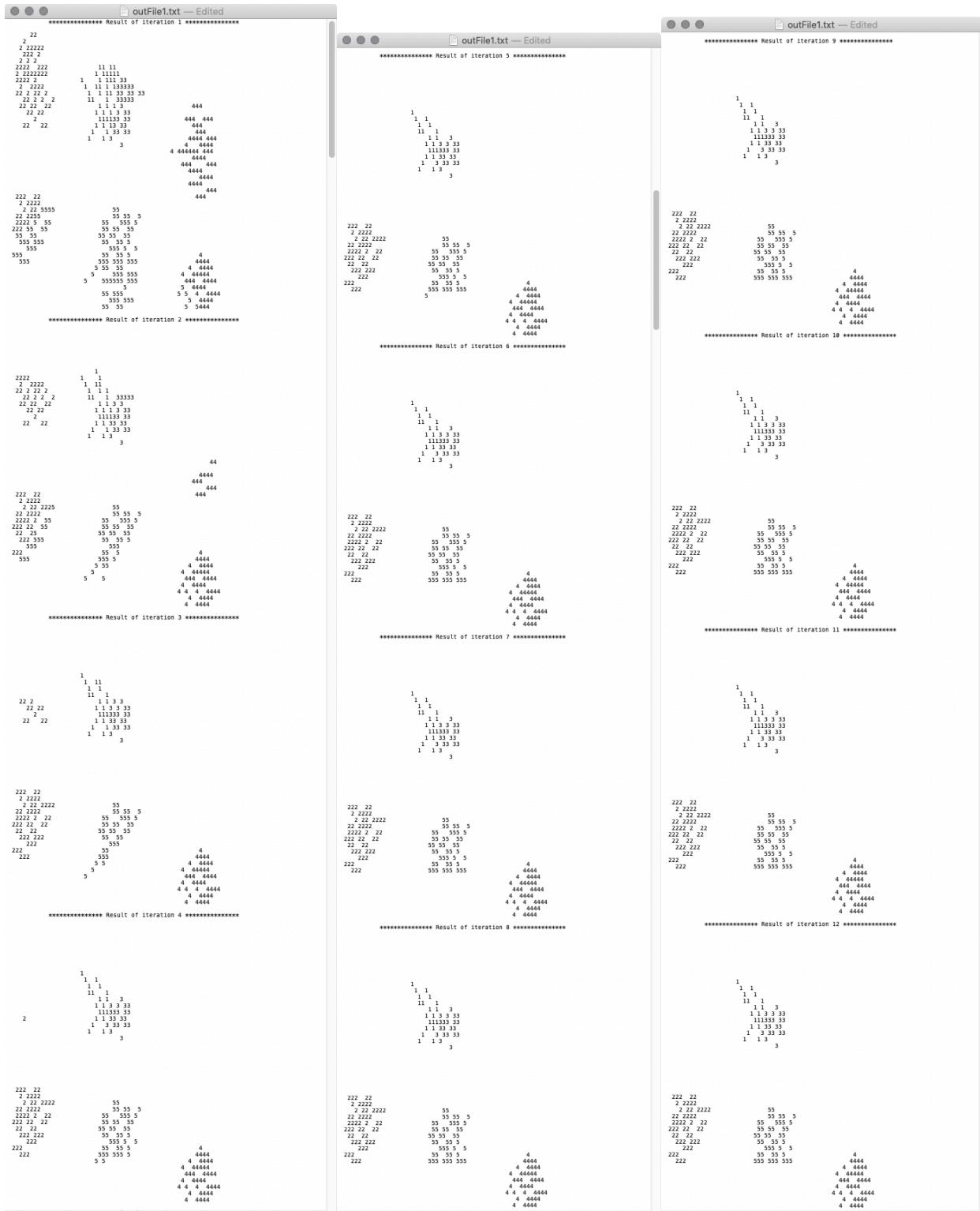
```

K=4

45 60	15 28 1	33 26 3
39 0	15 29 1	33 27 3
1 6 0	15 31 1	33 30 3
1 7 0	15 31 1	33 31 3
2 4 0	15 51 0	33 33 3
3 3 0	15 52 0	33 33 3
3 5 0	15 53 0	34 5 2
3 6 0	16 23 1	34 6 2
3 7 0	16 27 1	34 7 2
3 8 0	16 29 1	34 7 2
3 9 0	16 30 1	34 28 3
4 4 0	16 32 1	34 29 3
4 5 0	16 33 1	34 30 3
4 6 0	16 52 0	34 32 3
4 8 0	16 53 0	34 35 3
5 3 0	16 54 0	35 26 3
5 5 0	17 22 1	35 1 2
5 7 0	17 26 1	35 2 2
6 2 0	17 28 1	35 3 2
6 3 0	17 50 0	35 26 3
6 5 0	17 51 0	35 27 3
6 8 0	17 53 0	35 30 3
6 9 0	17 55 0	35 31 3
6 10 0	17 56 0	35 33 3
6 25 0	17 57 0	35 53 4
6 26 0	18 31 1	36 3 2
6 28 0	18 49 0	36 4 2
6 29 0	18 53 0	36 5 2
7 2 0	18 54 0	36 5 2
7 4 0	18 55 0	36 25 3
7 5 0	18 56 0	36 26 3
7 6 0	19 45 0	36 27 3
7 7 0	19 47 0	36 27 3
7 8 0	19 48 0	36 29 3
7 9 0	19 49 0	36 30 3
7 10 0	19 50 0	36 31 3
7 24 0	19 51 0	36 32 3
7 26 0	19 52 0	36 33 3
7 27 0	19 54 0	36 34 3
7 28 0	19 55 0	36 35 3
7 29 0	19 56 0	36 52 4
7 30 0	20 51 0	36 53 4
8 2 0	20 52 0	36 54 4
8 3 0	20 53 0	36 55 4
8 4 0	20 54 0	37 24 3
8 5 0	21 48 0	37 26 3
8 7 0	21 49 0	37 27 3
8 28 0	21 50 0	37 30 3
8 25 0	21 55 0	37 31 3
8 27 0	21 56 0	37 50 4
8 28 0	21 57 0	37 53 4
8 29 0	22 50 0	37 54 4
8 31 0	22 51 0	37 55 4
8 32 0	22 52 0	37 56 4
9 3 0	22 53 0	37 57 4
9 6 0	23 53 0	37 55 4
9 7 0	23 54 0	37 56 4
9 8 0	23 55 0	38 23 3
9 9 0	23 56 0	38 29 3
9 21 0	24 50 0	38 30 3
9 24 0	24 51 0	38 31 3
9 25 0	24 52 0	38 32 3
9 27 0	24 53 0	38 33 3
9 29 0	25 55 0	38 34 3
9 30 0	25 56 0	38 35 3
9 31 0	25 57 0	38 35 3
9 32 0	26 2 2	38 48 4
9 33 0	26 3 2	38 51 4
9 34 0	26 4 2	38 52 4
10 2 0	26 7 2	38 53 4
10 3 0	26 8 2	38 54 4
10 8 0	26 52 0	38 55 4
10 7 0	26 53 0	39 21 3
10 10 0	26 54 0	39 26 3
10 22 0	27 3 2	39 27 3
10 23 0	27 6 2	39 28 3
10 27 0	27 7 2	39 29 3
10 28 0	28 4 2	39 30 3
10 30 0	28 6 2	39 31 3
10 31 0	28 7 2	39 32 3
10 33 0	28 9 2	39 33 3
10 34 0	28 10 2	39 34 3
10 36 0	29 11 2	39 35 3
10 37 0	29 12 2	39 49 4
11 4 0	28 29 3	39 50 4
11 5 0	28 30 3	39 51 4
11 7 0	29 2 2	39 51 4
11 9 0	29 3 2	39 54 4
11 10 0	29 5 2	39 55 4
11 22 1	29 6 2	39 56 4
11 23 1	29 7 2	39 57 4
11 27 0	29 8 2	39 57 4
11 30 0	29 29 3	40 32 3
11 31 0	29 30 3	40 48 4
11 32 0	29 31 3	40 51 4
11 34 0	29 33 3	40 52 4
12 3 0	29 36 0	40 53 4
12 4 0	30 2 2	40 54 4
12 6 0	30 3 2	40 54 4
12 7 0	30 4 2	41 26 3
12 10 0	30 5 2	41 27 3
12 12 0	30 6 2	41 29 3
12 25 1	30 10 2	41 30 3
12 27 1	30 11 2	41 31 3
12 29 1	30 26 3	41 47 4
12 31 0	30 27 3	42 28 3
12 32 0	30 31 3	42 29 3
12 52 0	30 33 3	42 30 3
12 53 0	30 33 0	42 32 3
13 5 0	30 35 0	42 33 3
13 6 0	31 1 2	42 34 3
13 8 0	31 2 2	42 50 4
13 9 0	31 3 2	42 53 4
13 24 1	31 5 2	42 54 4
13 26 1	31 6 2	42 55 4
13 28 1	31 9 2	42 56 4
13 30 1	31 10 2	43 26 3
13 32 0	31 26 3	43 27 3
13 33 0	31 27 3	43 30 3
14 1 0	31 29 3	43 32 3
14 25 1	31 30 3	43 34 4
14 26 1	31 33 0	43 35 4
14 27 1	31 34 0	43 36 4
14 28 1	32 2 2	43 37 4
14 29 1	32 3 2	43 38 4
14 31 1	32 6 2	43 39 4
14 32 1	32 7 2	43 40 4
14 33 0	32 25 3	43 41 4
14 49 0	32 26 3	43 42 4
14 50 0	32 28 3	43 43 4
14 51 0	32 29 3	43 44 4
14 52 0	32 32 3	43 45 4
14 53 0	32 33 3	43 46 4
14 56 0	33 2 2	43 47 4
15 4 0	33 4 2	43 48 4
15 5 0	33 5 2	43 49 4
15 9 0	33 7 2	43 50 4
15 10 0	33 8 2	43 51 4
15 24 1	33 9 2	43 52 4
15 26 1	33 26 3	43 55 4

END OF K = 4

K = 5



outFile2.txt	outFile2.txt	outFile2.txt
45 60	15 26 1	33 7 2
39 0	15 28 3	33 8 2
1 6 0	15 29 3	33 9 2
1 7 0	15 31 3	33 26 5
2 4 0	15 32 3	33 27 5
3 3 0	15 51 0	33 30 5
3 5 0	15 52 0	33 31 5
3 6 0	15 53 0	33 33 5
3 7 0	16 23 1	34 5 2
3 8 0	16 23 3	34 6 2
3 9 0	16 29 3	34 7 2
4 4 0	16 30 3	34 28 5
4 5 0	16 32 3	34 29 5
4 6 0	16 33 3	34 30 5
4 8 0	16 52 0	34 32 5
5 3 0	16 53 0	34 35 5
5 5 0	16 54 0	35 1 2
5 7 0	17 22 1	35 2 2
6 2 0	17 26 1	35 3 2
6 3 0	17 28 3	35 26 5
6 8 0	17 50 0	35 27 5
6 5 0	17 51 0	35 30 5
6 8 0	17 53 0	35 31 5
6 9 0	17 55 0	35 33 5
6 10 0	17 56 0	35 35 4
6 25 0	17 57 0	36 25 5
6 26 0	18 31 3	36 26 5
6 28 0	18 31 5	36 27 5
6 29 0	18 52 0	36 29 5
7 2 0	18 53 0	36 30 5
7 4 0	18 54 0	36 31 5
7 5 0	18 55 0	36 33 5
7 6 0	18 56 0	36 4 2
7 7 0	19 45 0	36 5 2
7 8 0	19 47 0	36 25 5
7 9 0	19 48 0	36 26 5
7 10 0	19 49 0	36 27 5
7 24 0	19 50 0	36 29 5
7 26 0	19 51 0	36 30 5
7 27 0	19 52 0	36 31 5
7 28 0	19 54 0	36 33 5
7 29 0	19 55 0	36 34 5
7 30 0	19 56 0	36 35 5
8 2 0	20 51 0	36 52 4
8 3 0	20 52 0	36 53 4
8 4 0	20 53 0	36 54 4
8 5 0	20 54 0	36 55 4
8 7 0	21 49 0	37 24 0
8 28 1	21 50 0	37 26 0
8 25 0	21 55 0	37 27 0
8 27 0	21 56 0	37 30 0
8 28 0	21 57 0	37 31 0
8 29 0	22 58 0	37 50 4
8 31 0	22 59 0	37 53 4
8 32 0	22 52 0	38 34 0
9 3 0	22 53 0	38 35 4
9 6 0	23 53 0	38 36 4
9 7 0	23 54 0	38 37 4
9 8 0	23 55 0	38 38 4
9 9 0	23 56 0	38 39 4
9 21 1	24 50 0	38 40 4
9 24 1	24 51 0	38 41 4
9 25 0	24 52 0	38 42 4
9 27 0	24 53 0	38 43 4
9 29 0	25 55 0	38 44 4
9 30 0	25 56 0	38 45 4
9 31 0	25 57 0	38 46 4
9 32 0	26 2 2	38 47 4
9 33 0	26 3 2	38 48 4
10 2 0	26 4 2	38 51 4
10 3 0	26 7 2	38 52 4
10 7 0	26 8 2	38 53 4
10 8 0	28 4 2	38 54 4
10 10 0	28 6 2	38 55 4
10 22 1	28 7 2	39 21 0
10 25 1	28 9 2	39 26 0
10 27 0	28 11 2	39 27 0
10 28 0	28 12 2	39 29 0
10 30 0	28 29 5	39 30 0
10 31 0	28 30 5	39 31 0
10 33 0	29 2 2	39 33 0
10 34 0	29 2 5	39 34 0
10 36 0	29 5 2	39 35 0
10 37 0	29 6 2	39 36 0
11 4 0	29 8 2	39 37 0
11 5 0	29 29 5	39 49 4
11 7 0	29 30 5	39 50 4
11 9 0	29 2 2	39 51 4
11 11 0	29 2 5	39 54 4
11 22 1	29 5 2	39 55 4
11 23 1	29 6 2	39 56 4
11 27 1	29 7 2	39 57 4
11 30 0	29 8 2	40 32 0
11 31 0	29 29 5	40 48 4
11 34 0	29 30 5	40 51 4
12 3 0	29 33 5	40 52 4
12 4 0	29 36 5	40 53 4
12 6 0	30 2 2	40 54 4
12 7 0	30 3 2	41 26 0
12 10 0	30 4 2	41 27 0
12 11 0	30 5 2	41 29 0
12 25 1	30 7 2	41 30 0
12 27 1	30 10 2	41 31 0
12 29 0	30 11 2	41 47 4
12 31 3	30 26 5	42 28 0
12 5 0	30 27 5	42 29 0
12 52 0	30 31 5	42 30 0
12 53 0	30 32 5	42 32 0
13 5 0	30 33 5	42 33 0
13 6 0	30 35 5	42 34 0
13 8 0	31 1 2	42 50 4
13 9 0	31 2 2	42 53 4
13 24 1	31 2 5	42 54 4
13 26 1	31 6 2	42 55 4
13 28 3	31 9 2	42 56 4
13 30 3	31 10 2	43 26 0
13 32 3	31 26 5	43 27 0
13 33 3	31 27 5	43 28 0
14 1 0	31 5 5	43 29 0
14 25 1	31 30 5	43 30 0
14 26 1	31 33 5	43 31 0
14 27 1	31 34 5	43 49 4
14 28 3	32 2 2	43 52 4
14 29 3	32 3 2	43 53 4
14 31 3	32 5 2	43 54 4
14 32 3	32 7 2	43 55 4
14 33 3	32 7 5	43 56 4
14 49 0	32 25 5	43 57 4
14 50 0	32 26 5	43 58 4
14 51 0	32 28 5	44 30 0
14 52 0	32 29 5	44 49 4
14 53 0	32 32 5	44 50 4
14 56 0	32 33 5	44 53 4
15 4 0	33 3 2	44 54 4
15 5 0	33 4 2	44 55 4
15 9 0	33 5 2	44 56 4
15 10 0	33 7 2	44 57 4
15 24 1	33 8 2	44 58 4
15 26 1	33 9 2	44 59 4

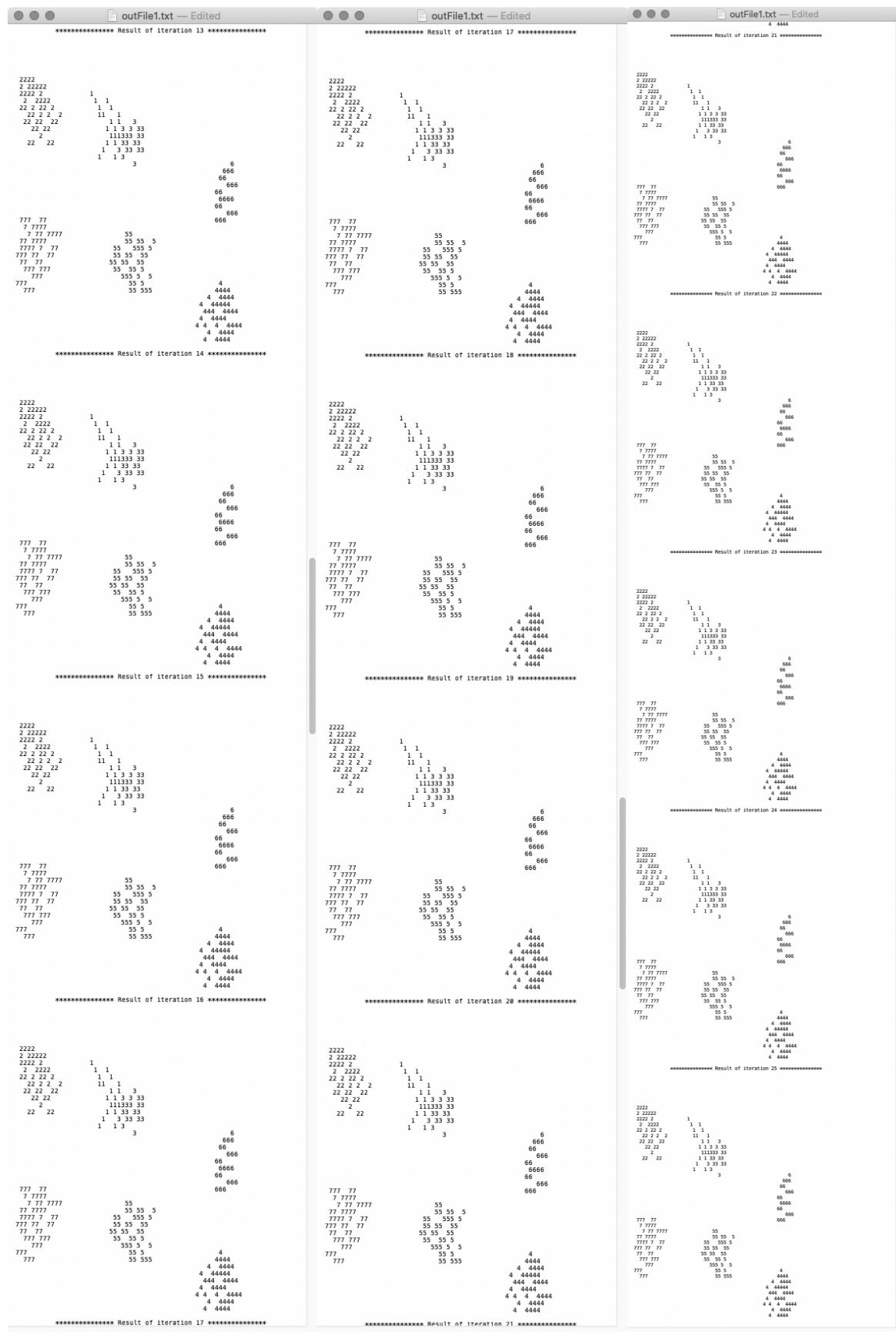
End of K = 5

K = 6

outFile2.txt	outFile2.txt	outFile2.txt
45 60	15 5 0	31 5 2
399	15 10 0	31 6 2
1 5 0	15 12 1	31 12 2
1 7 0	15 26 1	31 10 2
2 4 0	15 28 3	31 26 5
3 3 0	15 29 3	31 27 5
3 5 0	15 31 3	31 28 5
3 6 0	15 32 3	31 39 5
3 7 0	15 51 0	31 33 5
3 8 0	15 52 0	31 34 5
3 9 0	15 53 0	32 2 2
4 4 0	16 23 1	32 3 2
4 5 0	16 25 3	32 7 2
4 6 0	16 30 3	32 25 5
4 8 0	16 32 3	32 26 5
5 3 0	16 33 3	32 28 5
5 5 0	16 53 0	32 32 5
5 7 0	16 54 0	32 33 5
6 3 0	17 22 1	33 3 2
6 4 0	17 26 1	33 4 2
6 5 0	17 3 0	33 5 2
6 8 0	17 50 0	33 2
6 10 0	17 51 0	33 8 2
6 25 0	17 52 0	33 9 2
6 26 0	17 53 0	33 26 5
6 28 0	17 55 0	33 35
6 59 0	17 57 0	33 38 5
7 2 0	18 31 3	33 31 5
7 4 0	18 49 0	33 33 5
7 5 0	18 54 0	34 5 2
7 6 0	18 55 0	34 6 2
7 7 0	18 56 6	34 1 2
7 8 0	19 45 0	34 28 5
7 9 0	19 48 0	34 29 5
7 10 0	19 49 0	34 30 5
7 24 0	19 50 0	34 32 5
7 25 0	19 51 0	34 35 5
7 27 0	20 52 0	35 1 2
7 28 0	20 53 0	35 2 2
7 29 0	20 54 6	35 3 2
7 30 0	21 48 0	35 26 5
8 2 0	21 49 0	35 37 5
8 3 0	21 50 0	35 38 5
8 4 0	21 53 6	35 31 5
8 5 0	21 55 6	35 33 5
8 7 0	21 56 6	35 53 4
8 20 1	22 51 0	36 3 2
8 25 0	22 52 0	36 2
8 27 0	22 53 6	36 5 2
8 28 0	23 53 6	36 25 5
8 29 0	23 54 6	36 26 5
8 31 0	23 55 6	36 27 5
8 32 0	23 56 6	36 29 5
9 3 0	24 59 0	36 30 5
9 6 0	24 51 0	36 31 5
9 7 0	24 52 6	36 33 5
9 8 0	24 53 6	36 35 5
9 9 0	25 55 6	36 52 4
9 21 1	25 56 6	36 53 4
9 24 1	25 57 6	36 54 4
9 25 0	26 2 2	36 55 4
9 27 0	26 3 2	37 56 4
9 29 0	26 4 2	38 57 0
9 30 0	26 5 2	38 29 0
9 31 0	26 6 2	38 30 0
9 32 0	26 8 2	38 31 0
9 33 0	26 52 6	38 33 0
9 34 0	26 53 6	38 35 0
10 2 0	26 54 6	38 36 4
10 3 0	27 3 2	37 53 4
10 5 0	27 5 2	37 54 4
10 7 0	27 6 2	37 55 4
10 8 0	27 7 2	37 56 4
10 18 0	27 8 2	38 57 0
10 22 1	26 5 2	38 29 0
10 25 1	26 8 2	38 30 0
10 27 0	26 52 6	38 31 0
10 28 0	26 53 6	38 33 0
10 30 0	26 54 6	38 35 0
10 31 0	27 3 2	38 36 0
10 33 0	27 5 2	38 48 4
10 34 0	27 6 2	38 51 4
10 36 0	27 7 2	38 52 4
10 37 0	27 8 2	38 53 4
11 4 0	28 6 2	38 54 4
11 5 0	28 7 2	38 55 4
11 7 0	28 8 2	39 21 0
11 9 0	28 10 2	39 26 0
11 10 0	28 12 2	39 27 0
11 22 1	28 29 5	39 29 0
11 23 1	28 30 5	39 30 0
11 27 1	29 2 2	39 31 0
11 30 0	29 3 2	39 33 0
11 32 0	29 4 2	39 34 0
11 33 0	29 6 2	39 35 0
11 34 0	29 7 2	39 49 4
12 3 0	29 8 2	39 50 4
12 5 0	29 29 5	39 51 4
12 6 0	29 3 5	39 52 4
12 7 0	29 32 5	39 55 4
12 10 0	29 33 5	39 56 4
12 11 0	29 36 5	39 57 4
12 25 1	30 2 2	40 32 0
12 27 1	30 3 2	40 48 5
12 29 0	30 4 2	40 51 4
12 31 3	30 5 2	40 52 4
12 51 0	30 7 2	40 53 4
12 52 0	30 10 2	40 54 4
12 53 0	30 12 2	41 26 8
13 5 0	30 26 5	41 27 0
13 6 0	30 27 5	41 29 0
13 8 0	30 31 5	41 30 0
13 9 0	30 32 5	41 31 0
13 34 1	30 35 5	41 47 4
13 26 1	30 35 5	41 52 4
13 28 3	31 1 2	41 55 4
13 30 3	31 2 2	41 56 4
13 32 3	31 3 2	41 57 4
13 33 3	31 5 2	42 1 4
14 7 0	31 6 2	42 28 0
14 25 1	31 9 2	42 29 0
14 26 1	31 10 2	42 30 0
14 27 1	31 26 5	42 32 0
14 28 3	31 27 5	42 33 0
14 29 3	31 29 5	42 34 0
14 30 3	31 30 5	42 50 4
14 32 3	31 33 5	42 53 4
14 33 3	31 34 5	42 54 4
14 49 0	32 2 2	42 55 4
14 50 0	32 5 2	42 56 4
14 51 0	32 6 2	43 26 0
14 54 0	32 7 2	43 27 0
14 55 0	32 25 5	43 30 0
14 56 0	32 26 5	43 48 0
15 4 0	32 27 5	43 49 4
15 5 0	32 29 5	43 52 4
15 9 0	32 32 5	43 53 4
15 10 0	32 33 5	43 54 4
15 24 1	33 3 2	43 55 4
15 26 1	33 4 2	

END OF K = 6

K = 7



outFile2.txt	outFile2.txt	outFile2.txt
45 60	15 26 1	33 8 7 ,
39 0	15 28 3	33 9 7 ,
1 6 0	15 29 3	33 26 5
1 7 0	15 31 3	33 27 5
2 4 0	15 32 3	33 30 5
3 3 0	15 51 0	33 31 5
3 5 0	15 52 0	33 33 5
3 6 0	15 53 0	34 5 7
3 7 0	16 23 1	34 6 7
3 8 0	16 23 3	34 7 7
3 9 0	16 30 3	34 8 7
4 4 0	16 32 3	34 28 5
4 5 0	16 33 3	34 29 5
4 6 0	16 52 0	34 30 5
4 8 0	16 53 0	34 32 5
5 3 0	16 59 0	34 35 5
5 5 0	17 22 1	35 1 7
5 7 0	17 26 1	35 2 7
6 2 2	17 28 3	35 3 7
6 3 2	17 50 0	35 5 7
6 4 2	17 51 0	35 6 7
6 5 2	17 53 0	35 26 0
6 8 0	17 55 0	35 27 0
6 10 0	17 56 0	35 30 5
6 25 0	17 57 0	35 31 5
6 26 0	18 31 3	35 33 5
6 28 0	18 40 0	35 53 4
6 29 0	18 52 0	36 3 7
7 2 2	18 54 0	36 4 7
7 5 2	18 55 0	36 5 7
7 6 2	18 56 6	36 25 0
7 7 2	19 45 0	36 26 0
7 9 0	19 46 0	36 27 0
7 10 0	19 49 0	36 29 0
7 24 0	19 50 0	36 30 5
7 26 0	19 51 0	36 31 5
7 27 0	19 52 0	36 34 5
7 28 0	19 54 6	36 35 5
7 29 0	19 55 6	36 36 5
7 30 0	19 56 6	36 37 0
8 2 2	20 51 0	36 38 5
8 3 2	20 52 0	36 52 4
8 4 2	20 53 6	36 53 4
8 5 2	20 54 6	36 54 4
8 7 2	21 49 0	36 55 4
8 20 1	21 50 0	37 24 0
8 25 0	21 55 6	37 26 0
8 27 0	21 56 6	37 27 0
8 28 0	21 57 6	37 30 0
8 29 0	22 50 0	37 31 0
8 31 0	22 51 0	37 50 4
8 32 0	22 52 6	37 53 4
9 6 2	22 53 6	37 54 4
9 7 2	23 53 6	37 55 4
9 8 2	23 54 6	37 55 4
9 9 2	23 55 6	37 56 4
9 21 1	23 56 6	38 23 0
9 24 1	24 50 0	38 29 0
9 25 0	24 51 0	38 30 0
9 27 0	24 52 6	38 31 0
9 29 0	24 53 6	38 33 0
9 30 0	25 55 6	38 34 0
9 31 0	25 56 6	38 35 0
9 32 0	25 57 6	38 35 0
9 33 0	26 2 7	38 48 4
9 34 0	26 3 7	38 51 4
10 2 2	26 4 7	38 52 4
10 3 2	26 7 7	38 53 4
10 5 2	26 8 7	38 54 4
10 7 2	26 52 6	38 55 4
10 8 2	26 53 6	39 21 0
10 10 2	26 54 6	39 26 0
10 22 1	27 3 7	39 27 0
10 25 1	27 5 7	39 28 0
10 27 0	27 6 7	39 29 0
10 28 0	27 7 7	39 30 0
10 30 0	28 4 7	39 31 0
10 31 0	28 6 7	39 33 0
10 33 0	28 7 7	39 34 0
10 34 0	28 8 7	39 35 0
10 35 0	28 10 7	39 36 0
11 4 2	28 11 7	39 37 0
11 5 2	28 12 7	39 38 0
11 7 2	28 29 5	39 49 4
11 9 2	28 30 5	39 50 4
11 12 2	29 2 7	39 51 4
11 13 1	29 3 7	39 54 4
11 23 1	29 5 7	39 55 4
11 27 1	29 6 7	39 56 4
11 30 0	29 7 7	39 57 4
11 31 0	29 8 7	40 32 0
11 32 0	29 29 5	40 48 4
11 33 0	29 32 5	40 51 4
12 3 2	29 33 5	40 52 4
12 4 2	29 36 5	40 53 4
12 6 2	30 2 7	40 54 4
12 7 2	30 3 7	41 26 0
12 10 2	30 4 7	41 27 0
12 12 2	30 5 7	41 29 0
12 25 1	30 7 7	41 30 0
12 27 1	30 10 7	41 31 0
12 29 0	30 11 7	41 47 4
12 31 3	30 26 5	41 49 4
12 51 0	30 27 5	42 30 0
12 52 0	30 32 5	42 32 0
12 53 0	30 33 5	42 33 0
13 5 2	30 35 5	42 34 0
13 6 2	31 1 7	42 35 5
13 8 2	31 2 7	42 36 4
13 9 2	31 3 7	42 37 4
13 24 1	31 3 7	42 38 4
13 26 1	31 6 7	42 39 0
13 28 3	31 6 7	42 40 0
13 30 3	31 9 7	42 41 0
13 32 3	31 10 7	42 42 0
13 33 3	31 26 5	42 43 0
14 1 2	31 27 5	42 44 0
14 25 1	31 33 5	42 45 0
14 26 1	31 38 5	42 46 0
14 27 1	31 33 5	42 47 0
14 28 3	31 34 5	42 48 0
14 29 3	32 2 7	42 49 4
14 36 3	32 3 7	42 50 4
14 37 3	32 6 7	42 51 4
14 33 3	32 7 7	42 52 4
14 49 0	32 25 5	42 53 4
14 50 0	32 26 5	43 27 0
14 51 0	32 28 5	43 30 0
14 54 0	32 29 5	43 31 0
14 55 0	32 32 5	43 49 4
14 56 0	32 33 5	43 52 4
15 4 2	33 3 7	43 53 4
15 5 2	33 4 7	43 54 4
15 9 2	33 5 7	43 55 4
15 10 2	33 7 7	
15 24 1	33 8 7	
15 26 1	33 9 7	

END OF K = 7