```c
#include<stdio.h>

#define COL_COUNT 8
#define ROW_CAP 10

void getArray(FILE* inFile, double table[][COL_COUNT], int* nRow);
double getSum(double table[][COL_COUNT], int leftUpY, int leftUpX, int rightDownY, int rightDownX);
double maxSumConstPoint(double table[][COL_COUNT], int nRow, int leftUpY, int leftUpX, int* rightDownY, int* rightDownX);
double maxSumRec(double table[][COL_COUNT], int nRow, int* leftUpY, int* leftUpX, int* rightDownY, int* rightDownX);

int main(){
    double table[ROW_CAP][COL_COUNT];
    FILE* inFile;
    int nRow;
    double maxSum;
    int lUY, lUX, rDY, rDX;

    inFile=fopen("Table1.txt", "r");

    getArray(inFile, table, &nRow);

    maxSum=maxSumConstPoint(table, nRow, 0, 0, &rDY, &rDX);
    printf("MaxSum Rectangular starting from origin is %.2lf. Its right down coordinate (y,x) is %d, %d\n", maxSum, rDY, rDX);



    maxSum=maxSumRec(table, nRow, &lUY, &lUX, &rDY, &rDX);
    printf("MaxSum Rectangular is %.2lf. Its left upper coordinate (y,x) is %d, %d, right down coordinate is %d, %d\n", maxSum, lUY, lUX, rDY, rDX);

    fclose(inFile);
    return 0;
}

/*Reads the table from a file into a 2D array*/
void getArray(FILE* inFile, double table[][COL_COUNT], int* nRow){
    int row=0;
    int col;
    int status=EOF+1;/*Different from EOF*/

    /*one more row will be read but the values will not be recorded into the table
    therefore, it is safe to use a table having just enough capasity to hold the data*/
    while(status!=EOF){
        for(col=0; col<COL_COUNT; col++)
            status=fscanf(inFile, "%lf", &table[row][col]);
        ++row;
    }

    *nRow=row-1;/*one more row read*/
}

/*Returns the sum inside a given rectangular*/
double getSum(double table[][COL_COUNT], int leftUpY, int leftUpX, int rightDownY, int rightDownX){
    int row, col;
    double sum=0;

    for(row=leftUpY; row<=rightDownY; ++row)
        for(col=leftUpX; col<=rightDownX; ++col)
            sum+=table[row][col];

    return sum;
}

```

```
 63  /*Finds the rectangular left uppper point of which is specified having the max sum inside*/
 64  double maxSumConstPoint(double table[][COL_COUNT], int nRow, int leftUpY, int leftUpX, int* rightDownY, int
* rightDownX){
 65      int rDX;    /*x coordinate of the right down corner of the rec*/
 66      int rDY;    /*y coordinate of the right down corner of the rec*/
 67      double temp;
 68      /*initialize the rectangular with the one including only one point*/
 69      double sum=table[leftUpX][leftUpY];
 70      *rightDownY=leftUpY;
 71      *rightDownX=leftUpX;
 72
 73      /*Try all feasible rectangulars by changing the right down corner*/
 74      for(rDY=leftUpY; rDY<nRow; ++rDY){
 75          for(rDX=leftUpX; rDX<COL_COUNT; ++rDX){
 76              temp=getSum(table, leftUpY, leftUpX, rDY, rDX);
 77              printf("%d-",rDY);
 78              printf("%d\n",rDX);
 79              if(temp>sum){
 80                  /*a better rectangular is found, perform an update */
 81                  sum=temp;
 82                  *rightDownY=rDY;
 83                  *rightDownX=rDX;
 84              }
 85          }
 86      }
 87
 88      return sum;
 89  }
 90
 91
 92  double maxSumRec(double table[][COL_COUNT], int nRow, int* leftUpY, int* leftUpX, int* rightDownY, int*
rightDownX){
 93      double temp;
 94      int lUY, lUX;   /*coordinates of the left upper corner*/
 95      int rDY, rDX;   /*coordinates of the right down corner*/
 96      /*initialize the rectangular with the one including only origin point*/
 97      double maxSum=table[0][0];
 98      *leftUpY = *leftUpX = *rightDownY = *rightDownX = 0;
 99
100      /*For all feasible starting points call maxSumConstPoint*/
101      for(lUY=0; lUY<nRow; ++lUY){
102          for(lUX=0; lUX<COL_COUNT; ++lUX){
103              temp=maxSumConstPoint(table, nRow, lUY, lUX, &rDY, &rDX);
104              if(temp>maxSum){
105                  /*a better rectangular found, perform an update*/
106                  maxSum=temp;
107                  *leftUpY=lUY;
108                  *leftUpX=lUX;
109                  *rightDownY=rDY;
110                  *rightDownX=rDX;
111              }
112          }
113      }
114
115      return maxSum;
116  }
117
118
```