```c
#include<stdio.h>

#define COL_COUNT 8
#define ROW_CAP 10

typedef struct
{
    int x;
    int y;
}Point_t;

typedef struct
{
    Point_t left_up;
    Point_t right_down;
    double sum;
}Rectangle_t;

Point_t construct_point(int x,int y);
Rectangle_t construct_rectangle(Point_t left_up,Point_t right_down);
void print_rectangle(const Rectangle_t *rectangle);
void getArray(FILE* inFile, double table[][COL_COUNT], int* nRow);
void getSum(double table[][COL_COUNT],Rectangle_t *rectangle);
Rectangle_t maxSumConstPoint(double table[][COL_COUNT], int nRow, Point_t left_up);
Rectangle_t maxSumRec(double table[][COL_COUNT], int nRow);

int main(){
    double table[ROW_CAP][COL_COUNT];
    FILE* inFile;
    int nRow;

    Rectangle_t rectangle;
    Point_t start=construct_point(0,0);

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

    getArray(inFile, table, &nRow);

    rectangle = maxSumConstPoint(table, nRow,start);
    printf("MaxSum Rectangular starting from (y=%2d ,x=%2d) is %.2lf.\n",start.y,start.x,rectangle.sum);
    printf("Its right down coordinate (y,x) is %d, %d\n",rectangle.right_down.y, rectangle.right_down.x);

    rectangle = maxSumRec(table,nRow);
    printf("MaxSum Rectangular starting from (y=%2d ,x=%2d) is %.2lf.\n",rectangle.left_up.y,rectangle.
left_up.x,rectangle.sum);
    printf("Its right down coordinate (y,x) is %d, %d\n",rectangle.right_down.y, rectangle.right_down.x);

    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*/
```

```
66
67  }
68
69  /*Returns the sum inside a given rectangular*/
70  void getSum(double table[][COL_COUNT],Rectangle_t *rectangle)
71  {
72
73      int row, col;
74      (*rectangle).sum=0;
75      for(row=(*rectangle).left_up.y; row<=(*rectangle).right_down.y; ++row)
76          for(col=(*rectangle).left_up.x; col<=(*rectangle).right_down.x; ++col)
77                  ((*rectangle).sum)+=table[row][col];
78  }
79
80  /*Finds the rectangular left uppper point of which is specified having the max sum inside*/
81  Rectangle_t maxSumConstPoint(double table[][COL_COUNT], int nRow, Point_t left_up)
82  {
83      int temp=0;
84      Rectangle_t mSCP;
85      Point_t rDown=construct_point(left_up.x,left_up.y);
86      Point_t find;
87      mSCP.sum=table[left_up.x][left_up.y];
88      mSCP.left_up=rDown;
89      mSCP.right_down=rDown;
90
91      /*Try all feasible rectangulars by changing the right down corner*/
92      for(rDown.y=left_up.y; rDown.y<nRow; ++rDown.y){
93          for(rDown.x=left_up.x; rDown.x<COL_COUNT; ++rDown.x)
94              {
95                  mSCP.right_down=rDown;
96                  getSum(table,&mSCP);
97                  if(mSCP.sum>temp)
98                  {
99                      temp=mSCP.sum;
100                     find.x=rDown.x;
101                     find.y=rDown.y;
102                 }
103             }
104     }
105     mSCP.sum=temp;
106     mSCP.right_down=construct_point(find.x,find.y);
107
108     return mSCP;
109 }
110
111 Point_t construct_point(int x,int y)
112 {
113     Point_t points;
114
115     points.x=x;
116     points.y=y;
117
118     return points;
119 }
120
121 Rectangle_t construct_rectangle(Point_t left_up,Point_t right_down)
122 {
123     Rectangle_t c_rec;
124
125     c_rec.left_up=left_up;
126     c_rec.right_down=right_down;
127     return c_rec;
128 }
129 Rectangle_t maxSumRec(double table[][COL_COUNT], int nRow)
130 {
131     Point_t start;
```

```
    Rectangle_t new_rec;
    Rectangle_t found;
    double temp=0;
    int lUY, lUX;   /*coordinates of the left upper corner*/
    /*initialize the rectangular with the one including only origin point*/
    double maxSum=table[0][0];

    /*For all feasible starting points call maxSumConstPoint*/
    for(lUY=0; lUY<nRow; lUY++){
        for(lUX=0; lUX<COL_COUNT;lUX++)
        {

            start=construct_point(lUX,lUY);
            new_rec=maxSumConstPoint(table, nRow,start);

            if(new_rec.sum>temp)
            {
                temp=new_rec.sum;
                found = construct_rectangle(new_rec.left_up,new_rec.right_down);
                found.sum=temp;
            }
        }
    }
    return found;
}
```