Define **MAIN FUNCTION**:

1. Define global variables:
   1. L\*W: Dimensions of room
   2. N: number of LEDs
2. Call function Find\_Arrangements
   1. Store return values in count, possible\_arrangements[ ][ ]
3. Call function Best\_Arrangement
   1. Store values returned in idx, arr\_x [ ][ ], arr\_y [ ][ ].
4. Call function Best\_Led (idx, arr\_x [ ][ ], arr\_y [ ][ ])
5. EXIT

Define function **Find\_Arrangements( ):**

1. Fix max dimensions of each zone, width along x direction is maximum ‘p’ and width along y direction is maximum ‘q’. count=0;
2. Let lc\_n\_x define the location constraint along x axis for of zone\_n,

lc\_n\_y define the location constraint along y axis for of zone\_n,

zone\_n\_x\_min define the starting x coordinate of zone\_n,

zone\_n\_y\_min define the starting y coordinate of zone\_n,

zone\_n\_x define the ending x coordinate of zone\_n,

zone\_n\_y define the ending y coordinate of zone\_n,

n=1, 2, …..., N.

1. For zone\_1\_x=zone\_1\_x\_min to (zone\_1\_x< (zone\_1\_x\_min +p) and satisfies lc\_1\_x)
   1. For zone\_1\_y= zone\_1\_y\_min to (zone\_1\_y< (zone\_1\_y\_min +q) and satisfies lc\_1\_y)
      1. For zone\_2\_x= zone\_2\_x\_min to (zone\_2\_x< (zone\_2\_x\_min +p) and satisfies lc\_2\_x)
         1. For zone\_2\_y= zone\_2\_y\_min to (zone\_2\_y< (zone\_2\_y\_min +q) and satisfies lc\_2\_y)

.

.

.

.

For zone\_N\_x= zone\_N\_x\_min to (zone\_N\_x< (zone\_N\_x\_min +p) and satisfies lc\_N\_x)

For zone\_N\_y= zone\_N\_y\_min to (zone\_N\_y< (zone\_N\_y\_min +q) and satisfies lc\_N\_y)

count=count+1

arrangement= {(zones\_1\_x\_min, zone\_1\_x, zones\_1\_y\_min, zone\_1\_y), …, (zones\_N\_x\_min, zone\_N\_x, zones\_N\_y\_min, zone\_N\_y)}

Append arrangement to list ‘possible\_arrangements’

end

end

.

.

.

.

end

end

End

end

* 1. Return count, possible\_arrangements

Define function **Zone\_Led (possible\_arrangements, count)**

1. For a scenario in a room, fix priority for each cell based on attention points. Form L\*W matrix with each cell containing the weights based on priority, call it ‘weights\_matrix’\\

2. t=1, arr\_x= [ ],arr\_y= [ ]\\

3. Set window matrix as the photodiode analog values\\

4. For arrangement=possible\_arrangements(t)

b. For i:1 to N zones\\

i. zone\_matrix=weight\_matrix partitioned according to dimensions defined by possible\_arrangements (t, i)

ii. Perform correlation of zone matrix with window matrix and store in matrix ‘corr’\\

iii. Find corr\_max=max(corr)

iv. x\_best, y\_best= position of corr\_max in ‘corr’

v. Append x\_best to arr\_x[t], y\_best to arr\_y[t]

vi. Append corr\_max to all\_corr[t]

c. END FOR

d. t=t+1

5. END FOR

6. Repeat steps from 4,5 until t=count+1.

7. Return arr\_x [ ][ ], arr\_y [ ][ ], all\_corr [ ][ ].

Define function **Best\_Arrangement(count, possible\_arrangements)**

1. Call function Zone\_Led (count, possible\_arrangements)
   1. Store values returned in arr\_x [ ][ ], arr\_y [ ][ ], all\_corr [ ][ ].
2. for i=1 to count
   1. overall\_corr=0
   2. for j=1 to N
      1. overall\_corr= all\_corr[i][j] +overall\_corr
   3. Append overall\_corr to zones\_best [ ].
3. Find best\_zones=max (zones\_best [ ])
4. Find idx=position of best\_zones in zones\_best [ ]
5. Display “Best Zones’ arrangement is achieved in Arrangement ‘idx’ ”.
6. Return idx, arr\_x [ ][ ], arr\_y [ ][ ]

Define function **Best\_Led (idx, arr\_x [ ][ ], arr\_y [ ][ ]):**

1. for i=1: N
   1. display “Best LED position in zone ‘i’ is: (arr\_x[idx][i], arr\_y[idx][i])
2. END For