Amber Mishra
1BM18CS013

Write-up

Pseudo Code -

```
~~Count~~ count =0
int ar[n][n];
for(i=0; i<n; i++)
{
    for(j=0; j<n; j++)
    {
        if (ar[i][j]==1)
        {
            if(j+1<n && ar[j][j+1] ==1)
                union(i*(n)+j , (i)*(n)+(j+1))
            if(j-1>=0 && ar[i][j-1]==1)
                union(i*(n)+j , (i)*(n)+(j-1))
            if(i+1<n && ar[i+1][j]==1)
                union(i*n+j, (i+1)*n+j)
            if(i-1 >-1 && ar[i-1][j]==1)
                union(i*n+j , (i-1)*n+j)
            if(i+1<n && j+1 <n && ar[i+1][j+1]==1)
                union(i*n+j ,(i+1)*n+(j+1))
            if(i-1>-1 && j+1<n && ar[i-1][j+1]==1)
                union(i*n+j , (i-1)*n+j+1))
```

Checking all adjacent neighbours of a vertex.

```
    if (i-1 > -1 && j-1 > -1 && ar[i-1][j-1] == 1)
         union (i* n+k , (i-1) * n+(j-1))

    if ( i+1 < n && j-1 > -1 && ar[i+1] [j-1] == 1)
         union (i*n+k , (i+1) *n + (j-1) )

  int freq [] = new int [n *n];
freq. - it stores freq. of each set

for(i=0; i<n; i++)
 {
   for(j=0; j<n; j++)
   {
     if (ar[i] [j] == 1 )
       {
         x = find (i*n+k) // function to find connected distinct
                                        vertices.
         if ( freq[x] = = 0)
           {
             count++ ;
             freq[x]++;
           }
         else
         freq[x]++;
       }
   }
 }
```