**Count Inversions**

Let A[0 ... n-1] be an array of n distinct positive integers. If i < j and A[i] > A[j] then the pair (i, j) is called an inversion of A (where i and j are indexes of A). Given an integer array A, your task is to find the number of inversions in A.

Long long merge2(int A[],int s, int mid, int n)

{

int i=s,j=mid,k=0;

int temp[n-s+1];

long long count=0;

while(i<mid && j<=n)

{

if(A[i]<=A[j])

{

temp[k++]=A[i++];

}

else

{

temp[k++]=A[j++];

count += mid-i;

}

}

while(i<mid){

temp[k++]=A[i++];

}

while(j<=n)

{

temp[k++]=A[j++];

}

for(i=s,k=0;i<=n;i++,k++)

{

A[i]=temp[k];

}

return count;

}

long long merge1(int A[],int s, int n){

long long count=0;

if(n>s)

{

int mid= (s+n)/2;

long long left= merge1(A , s,mid);

long long right= merge1( A,mid+1,n );

long long merge= merge2( A, s, mid+1,n);

return merge+left+right;

}

return count;

}

long long solve(int A[], int n)

{

long k=merge1(A,0,n-1);

return k;

}