Exercise 2-4:

a. Answer:

Array:{2,3,8,6,1}

Inversions:{2,1},{3,1},{8,1},{6,1},{8,6}

b. Answer:

The permutation {n,n-1,...,1} has the most inversions, $\frac{n(n-1)}{2}$.

c. Answer:

The number of shifts performed during insertion sort equals its number of inversions.

Because every shift is equivalent to swapping adjacent pair, thus reducing the inversion by 1.

When the sequence is sorted, inversion is reduced to 0.

d. Answer:

Here is my code using C++.

```
// Exercise 2-4 Inversions
// This is my old code, from POJ 2299, that counts the inversions of a sequence.
#define _CRT_SECURE_NO_WARNINGS
#include <cstdio>
#include <cstdlib>
using namespace std;

int a[510000];
int t[510000];
int tc;
int n;
long long int inv;

void merge(int a[], int left, int mid, int right)
{
    int i, j;

    tc = left;
    i = left;
    j = mid + 1;
    while(true){
        if(i <= mid){
            if(j <= right){
                if(a[i] <= a[j]){
                    t[tc++] = a[i++];
                }else{
                    t[tc++] = a[j++];
                    inv += mid - i + 1;
                }
            }else{
                t[tc++] = a[i++];
            }
        }else{
            if(j <= right){
                t[tc++] = a[j++];
            }else{
                break;
            }
        }
    }

    for(i = left; i <= right; ++i){
        a[i] = t[i];
    }
}

void msort(int a[], int left, int right)
{
    int mid;

    if(left < right){
        mid = (left + right) / 2;
        msort(a, left, mid);
        msort(a, mid + 1, right);
        merge(a, left, mid, right);
    }
}

void MergeSort(int a[], int n)
{
    if(n < 2){
        return;
    }
    msort(a, 0, n - 1);
}
```

```
int main()
{
    int i;

    while(scanf("%d", &n) == 1 && n){
        for(i = 0; i < n; ++i){
            scanf("%d", &a[i]);
        }
        inv = 0;
        MergeSort(a, n);

        printf("%lld\n", inv);
    }

    return 0;
}
```