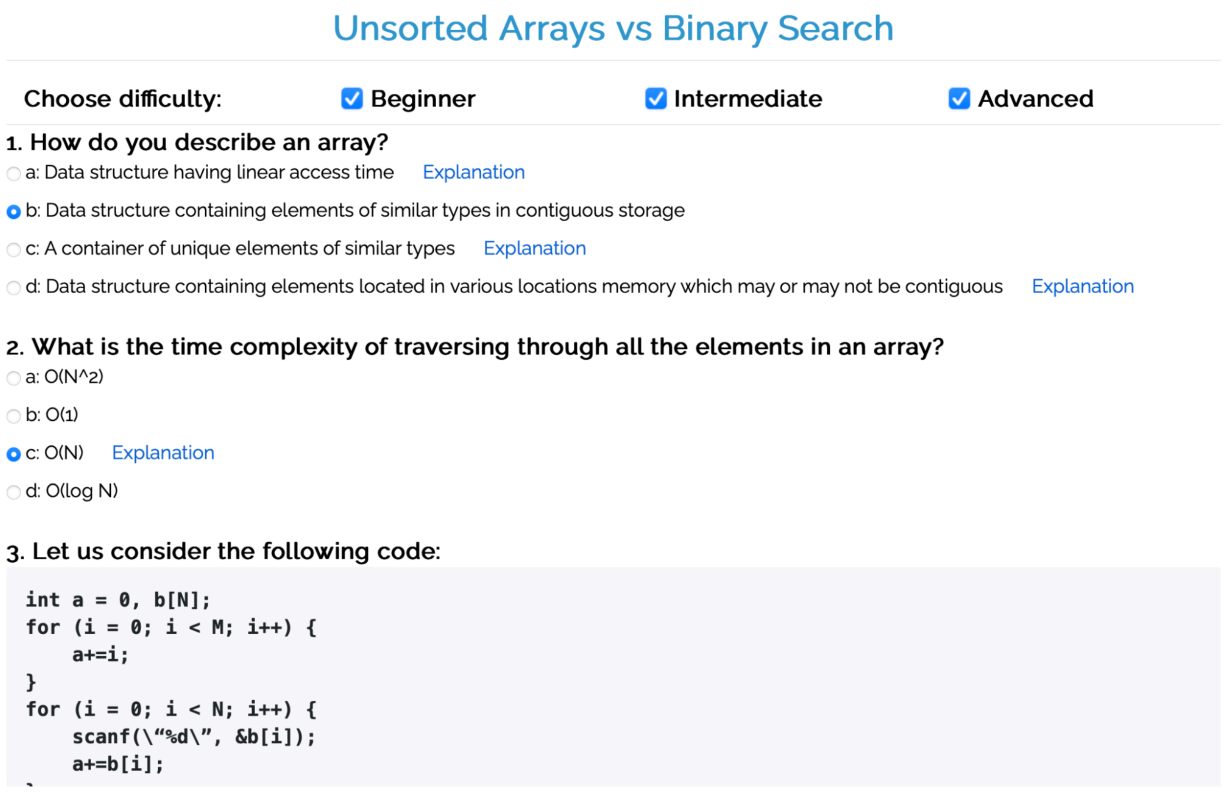
15B17CI371 – Data Structures Lab

ODD 2024
Week 3-LAB B
Practice Lab

Virtual Lab



Unsorted Arrays vs Binary Search

Choose difficulty: [✓ Beginner] [✓ Intermediate] [✓ Advanced]

1. How do you describe an array?

a: Data structure having linear access time Explanation

[✓ b: Data structure containing elements of similar types in contiguous storage]

c: A container of unique elements of similar types Explanation

d: Data structure containing elements located in various locations memory which may or may not be contiguous Explanation

2. What is the time complexity of traversing through all the elements in an array?

a: O(N^2)

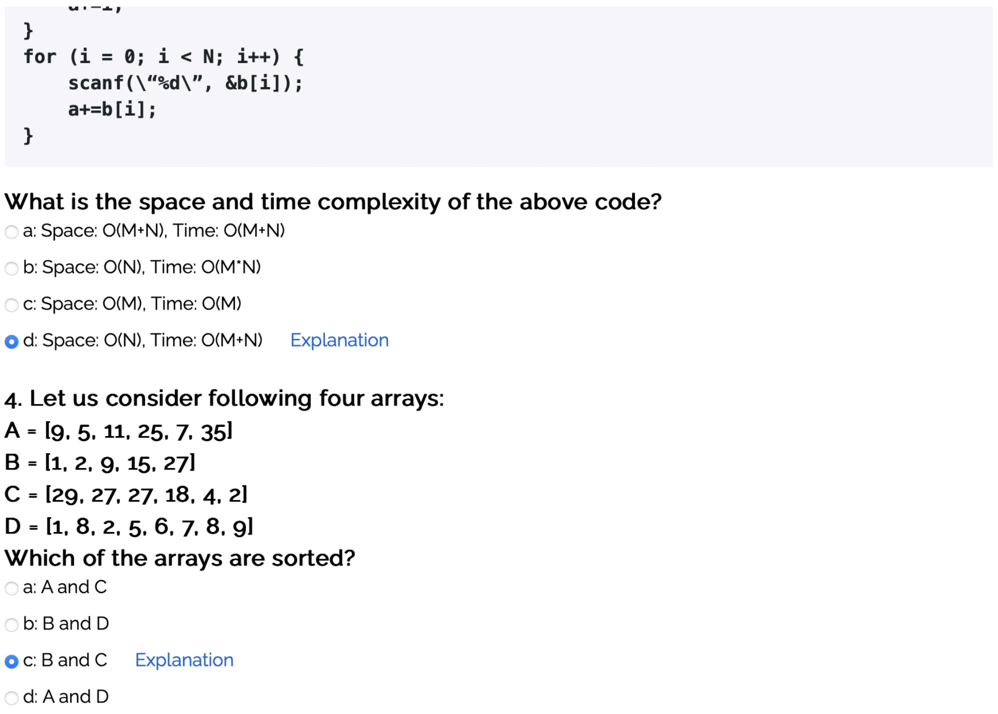
b: O(1)

[✓ c: O(N)] Explanation

d: O(log N)

3. Let us consider the following code:

```
int a = 0, b[N];
for (i = 0; i < M; i++) {
    a+=i;
}
for (i = 0; i < N; i++) {
    scanf(\“%d\”, &b[i]);
    a+=b[i];
```



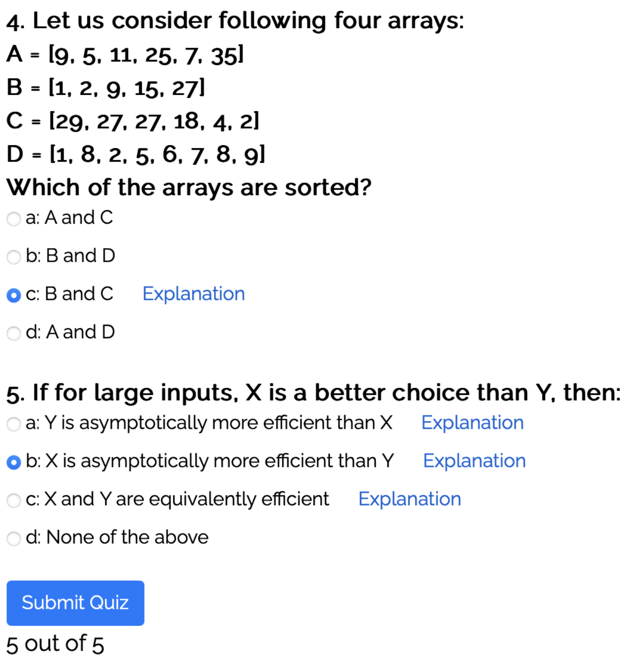
```
}
for (i = 0; i < N; i++) {
    scanf(\"%d\", &b[i]);
    a+=b[i];
}
```

What is the space and time complexity of the above code?

- a: Space: O(M+N), Time: O(M+N)
- b: Space: O(N), Time: O(M*N)
- c: Space: O(M), Time: O(M)
- [✓ d: Space: O(N), Time: O(M+N)] Explanation

4. Let us consider following four arrays:
A = [9, 5, 11, 25, 7, 35]
B = [1, 2, 9, 15, 27]
C = [29, 27, 27, 18, 4, 2]
D = [1, 8, 2, 5, 6, 7, 8, 9]
Which of the arrays are sorted?

- a: A and C
- b: B and D
- [✓ c: B and C] Explanation
- d: A and D



4. Let us consider following four arrays:
A = [9, 5, 11, 25, 7, 35]
B = [1, 2, 9, 15, 27]
C = [29, 27, 27, 18, 4, 2]
D = [1, 8, 2, 5, 6, 7, 8, 9]
Which of the arrays are sorted?

- a: A and C
- b: B and D
- [✓ c: B and C] Explanation
- d: A and D

5. If for large inputs, X is a better choice than Y, then:

- a: Y is asymptotically more efficient than X Explanation
- [✓ b: X is asymptotically more efficient than Y] Explanation
- c: X and Y are equivalently efficient Explanation
- d: None of the above

Submit Quiz

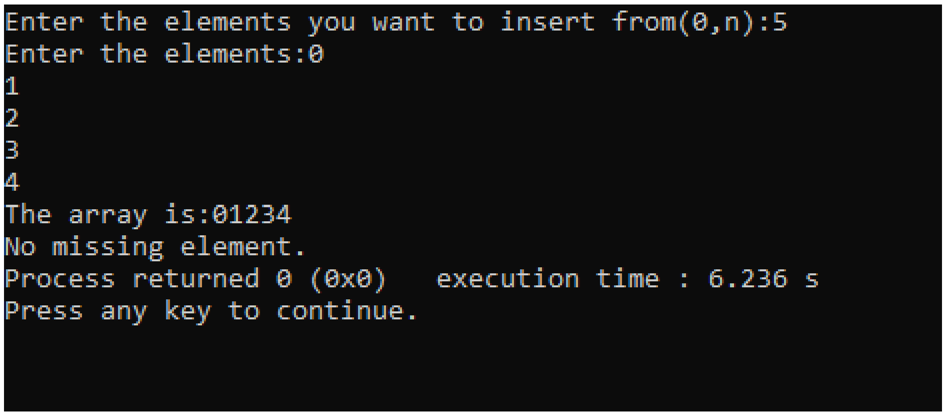
5 out of 5

1.

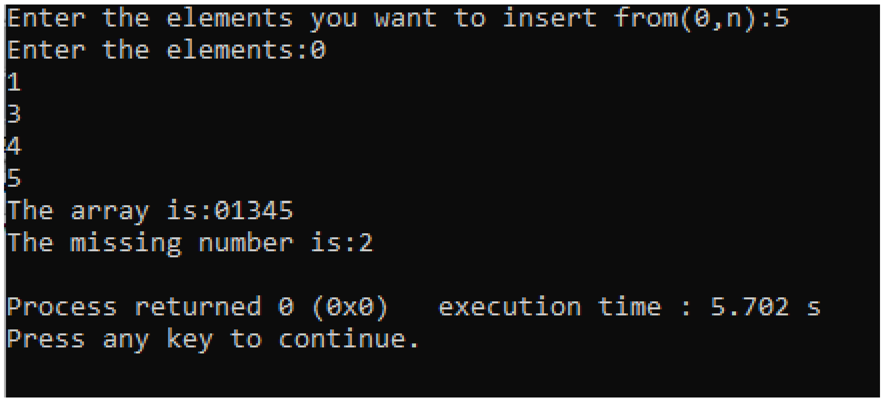
#include<iostream>

```
using namespace std;

int search(int arr[],int n,int x)

{

   for(int i=0;i<n;i++)

   {

      if(arr[i]==x)

      {



         return 1;

      }


   }

      return x;


}

int main()

{

   int n;

   cout<<"Enter the elements you want to insert from(0,n):";

   cin>>n;

   int arr[n];

   cout<<"Enter the elements:";

   int a;
```

```
 for(int i=0;i<n;i++)
{
    cin>>a;
    arr[i]=a;
}
cout<<"The array is:";
 for(int i=0;i<n;i++)
{
    cout<<arr[i];
}
cout<<endl;
int y,c=0;
for(int i=0;i<n;i++)
{
    y=search(arr,n,i);
 if(y!=1)
 {
    cout<<"The missing number is:"<<y;
    cout<<endl;
    c++;

 }
}
if(c==0)
{
        cout<<"No missing element.";

    }



}
```



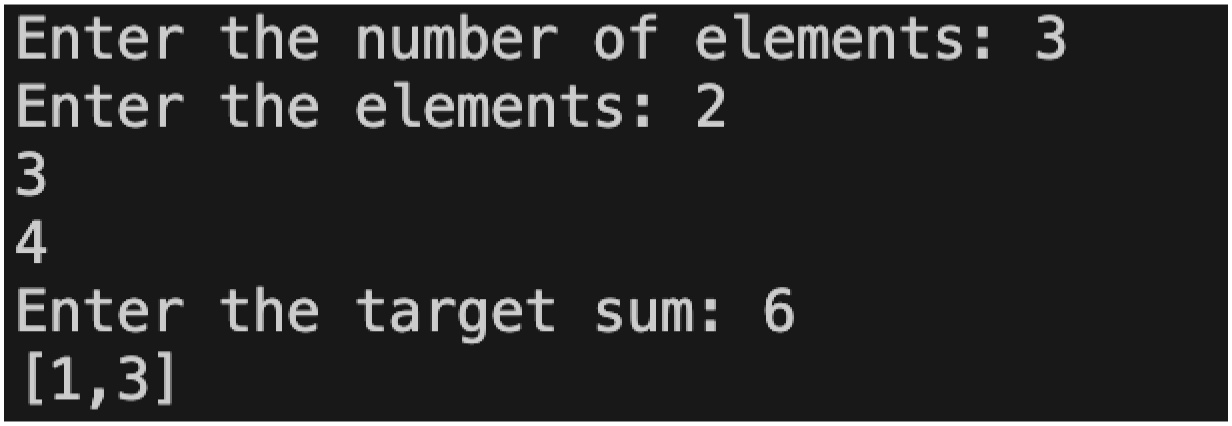
```
Enter the elements you want to insert from(0,n):5
Enter the elements:0
1
2
3
4
The array is:01234
No missing element.
Process returned 0 (0x0)   execution time : 6.236 s
Press any key to continue.
```



```
Enter the elements you want to insert from(0,n):5
Enter the elements:0
1
3
4
5
The array is:01345
The missing number is:2

Process returned 0 (0x0)   execution time : 5.702 s
Press any key to continue.
```

2.

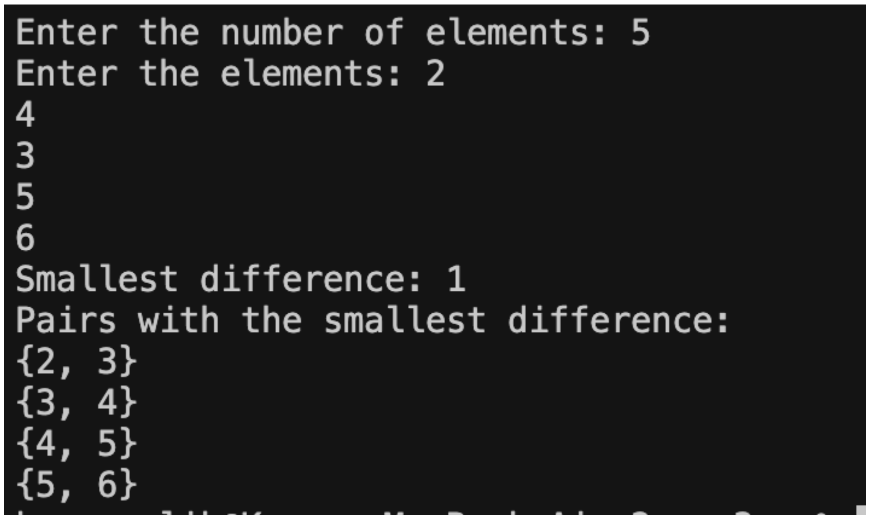
```
#include <iostream>

using namespace std;

#define MAX_SIZE 100

void bubbleSort(int arr[], int size) {
   for (int i = 0; i < size - 1; ++i) {
      for (int j = 0; j < size - i - 1; ++j) {
         if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

void findPairWithSum(int arr[], int size, int target) {
    if (size < 2) {
        cout << "No pairs found" << endl;
        return;
    }

    bubbleSort(arr, size);

    int left = 0;
    int right = size - 1;

    while (left < right) {
        int currentSum = arr[left] + arr[right];

        if (currentSum == target) {
            cout << "[" << left + 1 << "," << right + 1 << "]" << endl;
            return;
        } else if (currentSum < target) {
            ++left;
        } else {
            --right;
        }
    }

    cout << "No pairs found" << endl;
}

int main() {
    int size;
    cout << "Enter the number of elements: ";
    cin >> size;

    if (size < 2 || size > MAX_SIZE) {
        cout << "Invalid size. Size must be between 2 and " << MAX_SIZE << "." << endl;
        return 1;
    }
    int array[MAX_SIZE];
    cout << "Enter the elements: ";
    for (int i = 0; i < size; ++i) {
        cin >> array[i];
    }

    int target;
    cout << "Enter the target sum: ";
    cin >> target;

    findPairWithSum(array, size, target);

    return 0;
}
```



```
Enter the number of elements: 3
Enter the elements: 2
3
4
Enter the target sum: 6
[1,3]
```

3.

```
#include <iostream>
#include <climits>
#include<math.h>
using namespace std;
void bubbleSort(int arr[], int n) {
bool swapped;
for (int i = 0; i < n - 1; ++i)
 {swapped = false;
for (int j = 0; j < n - i - 1; ++j) {
if (arr[j] > arr[j + 1]) {
int temp = arr[j];
arr[j] = arr[j + 1];
arr[j + 1] = temp;
swapped = true;
}
}
if (!swapped)
break;
}
}
void findPairsWithSmallestDifference(int arr[], int n) {
if (n < 2) {
cout << "Not enough elements to form pairs." << std::endl;
return;
}
bubbleSort(arr, n);
int minDiff=abs(arr[0]-arr[1]);
for (int i = 1; i < n; ++i) {
int diff = arr[i] - arr[i - 1];
if (abs(diff) < minDiff) {
minDiff = diff;
}
}
cout << "Smallest difference: " << minDiff << endl;
cout << "Pairs with the smallest difference: " << endl;
for (int i = 1; i < n; ++i) {
if (abs(arr[i] - arr[i - 1]) == minDiff) {
cout << "{" << arr[i - 1] << ", " << arr[i] << "}" << endl;
}
}
}
int main() {
int n;
cout << "Enter the number of elements: ";
cin >> n;
if (n <= 0) {
cout << "Number of elements must be positive." << std::endl;
return 1;
}
int* arr = new int[n];
cout << "Enter the elements: ";
for (int i = 0; i < n; ++i) {
cin >> arr[i];
}
findPairsWithSmallestDifference(arr, n);
delete[] arr;
return 0;
}
```

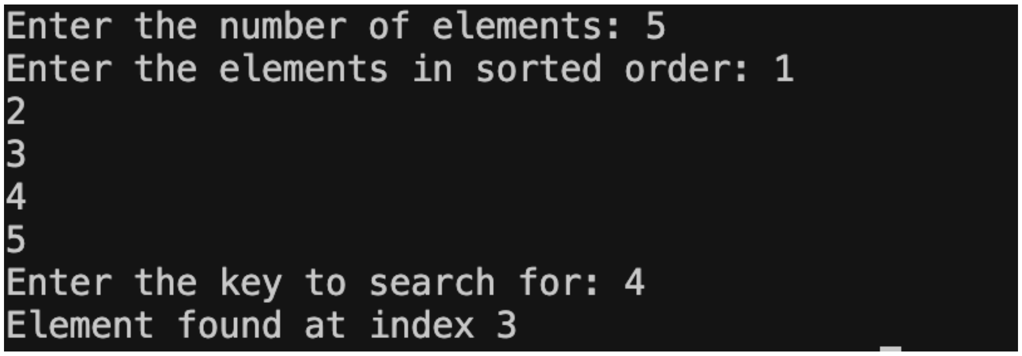


```
Enter the number of elements: 5
Enter the elements: 2
4
3
5
6
Smallest difference: 1
Pairs with the smallest difference:
{2, 3}
{3, 4}
{4, 5}
{5, 6}
```

4.

```
#include <iostream>

using namespace std;

int interpolationSearch(int arr[], int size, int key) {
    int low = 0;
    int high = size - 1;

    while (low <= high && key >= arr[low] && key <= arr[high]) {
        if (low == high) {
            if (arr[low] == key) return low;
            return -1;
        }

        int pos = low + ((key - arr[low]) * (high - low) / (arr[high] - arr[low]));

        if (arr[pos] == key) {
            return pos;
        } else if (arr[pos] < key) {
            low = pos + 1;
        } else {
            high = pos - 1;
        }
    }

    return -1;
}
```

```
int main() {
    int size;
    cout << "Enter the number of elements: ";
    cin >> size;

    int arr[size];
    cout << "Enter the elements in sorted order: ";
    for (int i = 0; i < size; ++i) {
        cin >> arr[i];
    }

    int key;
    cout << "Enter the key to search for: ";
    cin >> key;

    int result = interpolationSearch(arr, size, key);

    if (result != -1) {
        cout << "Element found at index " << result << endl;
    } else {
        cout << "Element not found in the array" << endl;
    }

    return 0;
}
```

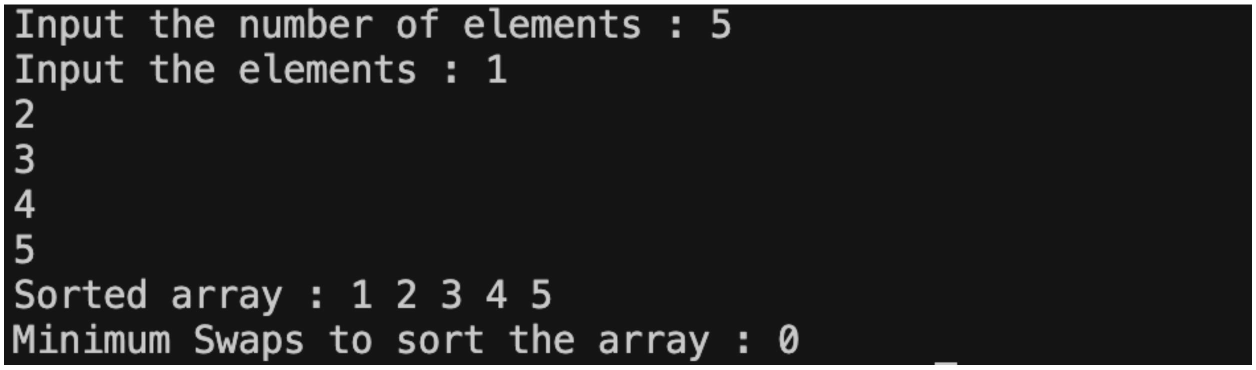


```
Enter the number of elements: 5
Enter the elements in sorted order: 1
2
3
4
5
Enter the key to search for: 4
Element found at index 3
```

5.

```
#include <iostream>
```

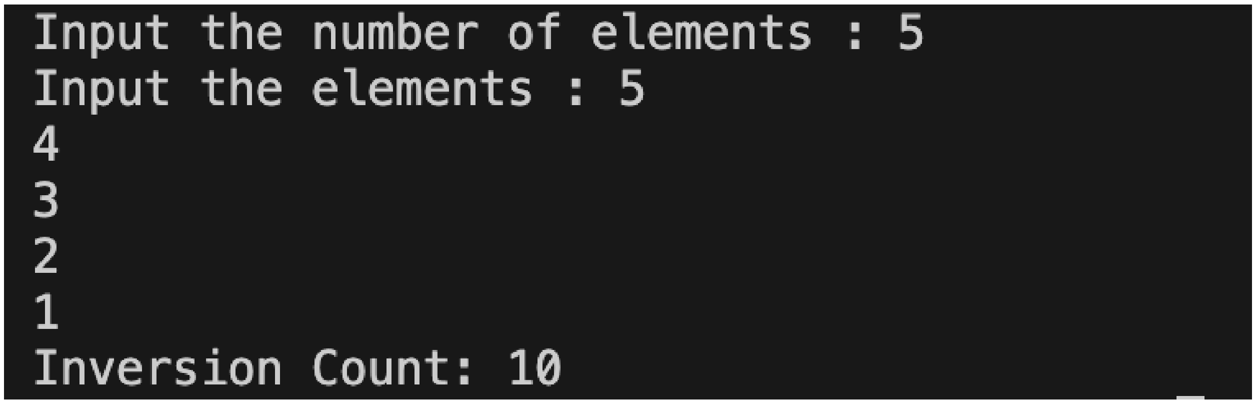
```
using namespace std;
int minSwaps(int *arr,int n) {
int swaps=0;
for (int i=0; i<n; i++)
{
int min=arr[i];
int index=i;
for(int j=i+1;j<n;j++)
if(arr[i]>arr[j])
{
int temp=arr[i];
arr[i]=arr[j];
arr[j]=temp;
swaps++;
}
}
return swaps;
}
int main()
{
int n;
cout<<"Input the number of elements : ";
cin>>n;
cout<<"Input the elements : ";
int *arr=new int[n];
for(int i=0;i<n;i++)
cin>>arr[i];
int a=minSwaps(arr,n);
cout<<"Sorted array : ";
for(int i=0; i<n; i++)
cout<<arr[i]<<" ";
cout<<"\nMinimum Swaps to sort the array : "<<a<<endl;
return 0;
}
```



```
Input the number of elements : 5
Input the elements : 1
2
3
4
5
Sorted array : 1 2 3 4 5
Minimum Swaps to sort the array : 0
```

6.

```
#include <iostream>
#include <vector>
using namespace std;
int mergeAndCount(int *arr,int left,int mid,int right)
{
    int n1=mid-left+1;
    int n2=right-mid;
    vector<int> leftArr(n1);
    vector<int> rightArr(n2);
    for (int i=0; i<n1; i++)
leftArr[i]=arr[left+i];
    for (int i=0; i<n2; i++)
 rightArr[i]=arr[mid+1+i];
    int i=0,j=0,k=left,swaps=0;
    while (i<n1 && j<n2)
    {
        if (leftArr[i]<=rightArr[j])
            arr[k++]=leftArr[i++];
        else
        {
            arr[k++]=rightArr[j++];
            swaps+=(n1-i);
        }
    }
    while (i<n1) arr[k++]=leftArr[i++];
    while (j<n2) arr[k++]=rightArr[j++];
    return swaps;
}
int mergeSortAndCount(int* arr,int left,int right)
{
    int count=0;
    if (left<right)
    {
        int mid=left+(right-left) / 2;
        count+=mergeSortAndCount(arr,left,mid);
        count+=mergeSortAndCount(arr,mid+1,right);
        count+=mergeAndCount(arr,left,mid,right);
    }
    return count;
}
int main()
{
    int n;
    cout<<"Input the number of elements : ";
    cin>>n;
    cout<<"Input the elements : ";
    int *arr=new int[n];
    for(int i=0;i<n;i++)
        cin>>arr[i];
    int result=mergeSortAndCount(arr,0,n-1);
    cout<<"Inversion Count: "<<result<<endl;
    return 0;
}
```



```
Input the number of elements : 5
Input the elements : 5
4
3
2
1
Inversion Count: 10
```