**Design and Analysis of Algorithms**

**Lab1(16-07-2021)**

**Assignment - Sorting**

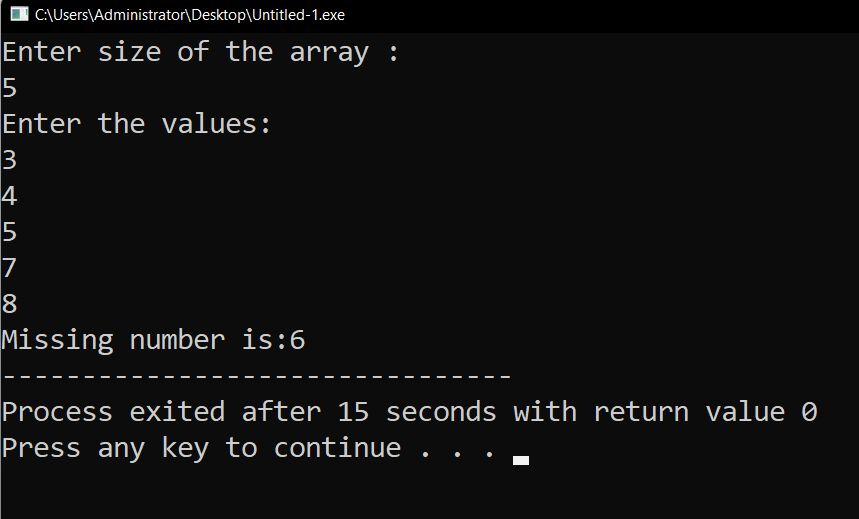
* **R.Abhinav**
* **CB.EN.U4CSE19453**

1. An array of n elements contains all but one of the integers from 1 to n + 1.
   1. Give the best algorithm you can for determining which number is missing if the array is sorted, and analyze its asymptotic worst-case running time.

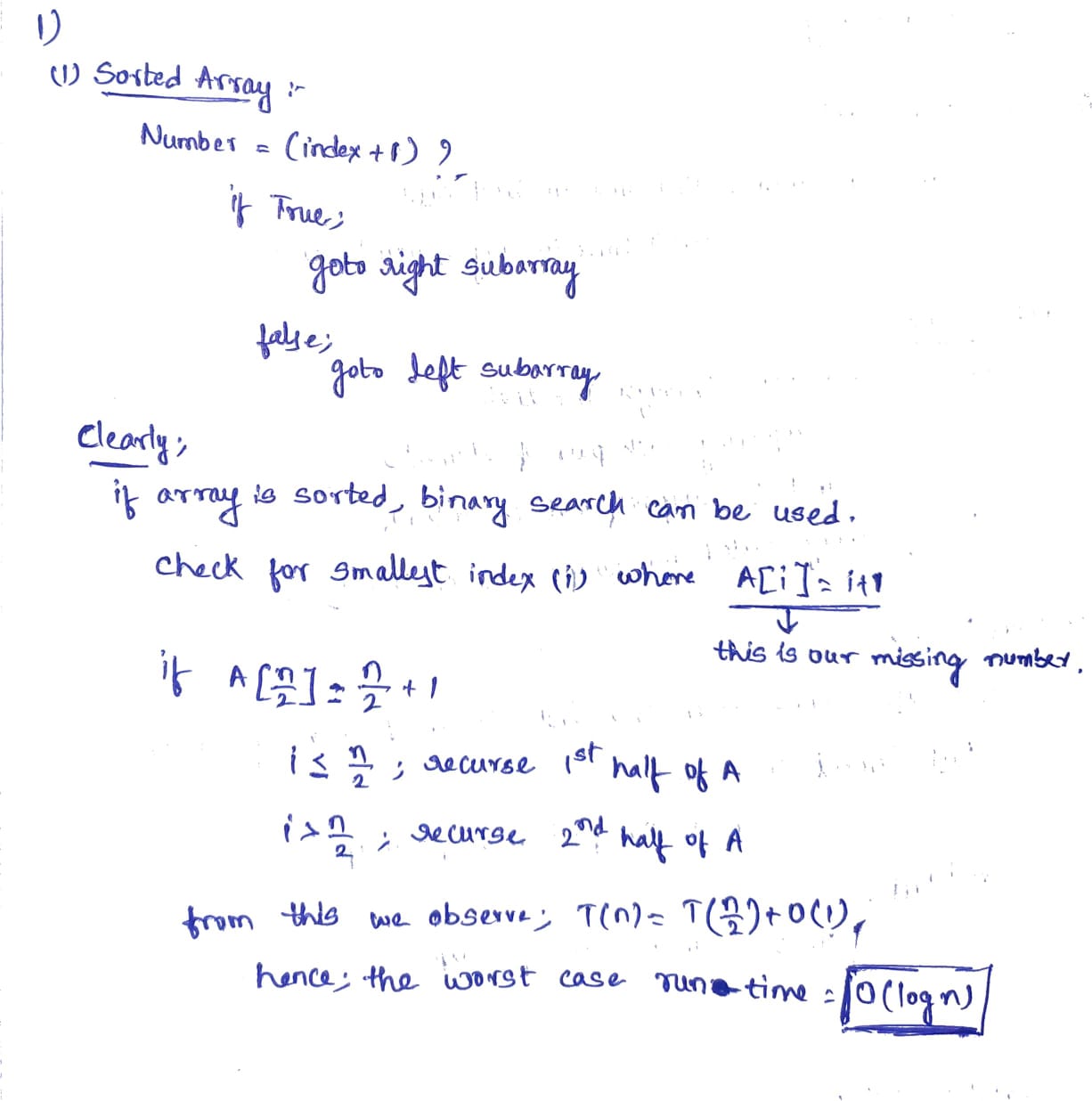
**Code:**

#include <iostream>  
using namespace std;  
  
int check(int array[], int n)  
{  
 int i = 0, j = n - 1;  
 int middle;  
 while ((j - i) > 1)  
 {  
 middle = (i + j) / 2;  
 if ((array[i] - i) != (array[middle] - middle))  
 j = middle;  
 else if ((array[j] - j) != (array[middle] - middle))  
 i = middle;  
 }  
 return (array[i] + 1);  
}  
  
int main()  
{  
 int num;  
 cout<<"Enter size of the array :"<<endl;  
 cin >> num;  
 cout<<"Enter the values:"<<endl;  
 int arr[num];  
 for (int i = 0; i < num; i++)  
 {  
 cin >> arr[i];  
 }  
 cout << "Missing number is:" << check(arr, num);  
}

**Output:**

****

**Explaination:**

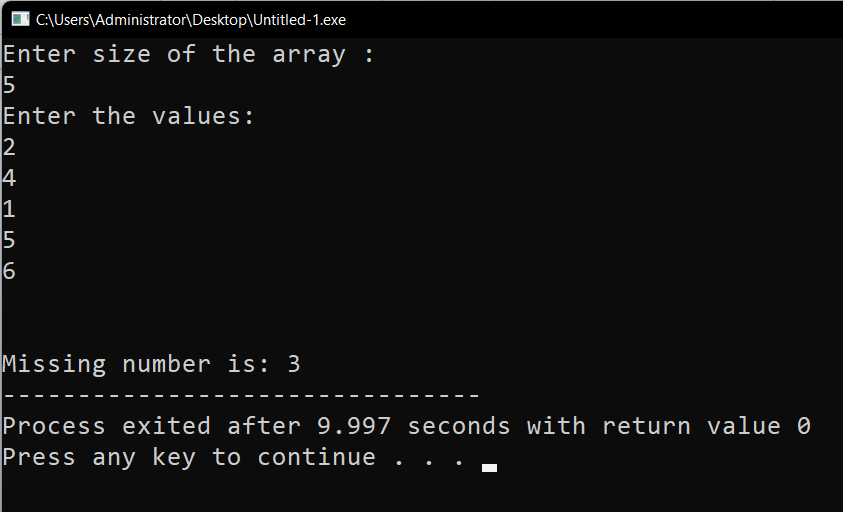


* 1. Give the best algorithm you can for determining which number is missing if the array is not sorted, and analyze its asymptotic worst-case running time.

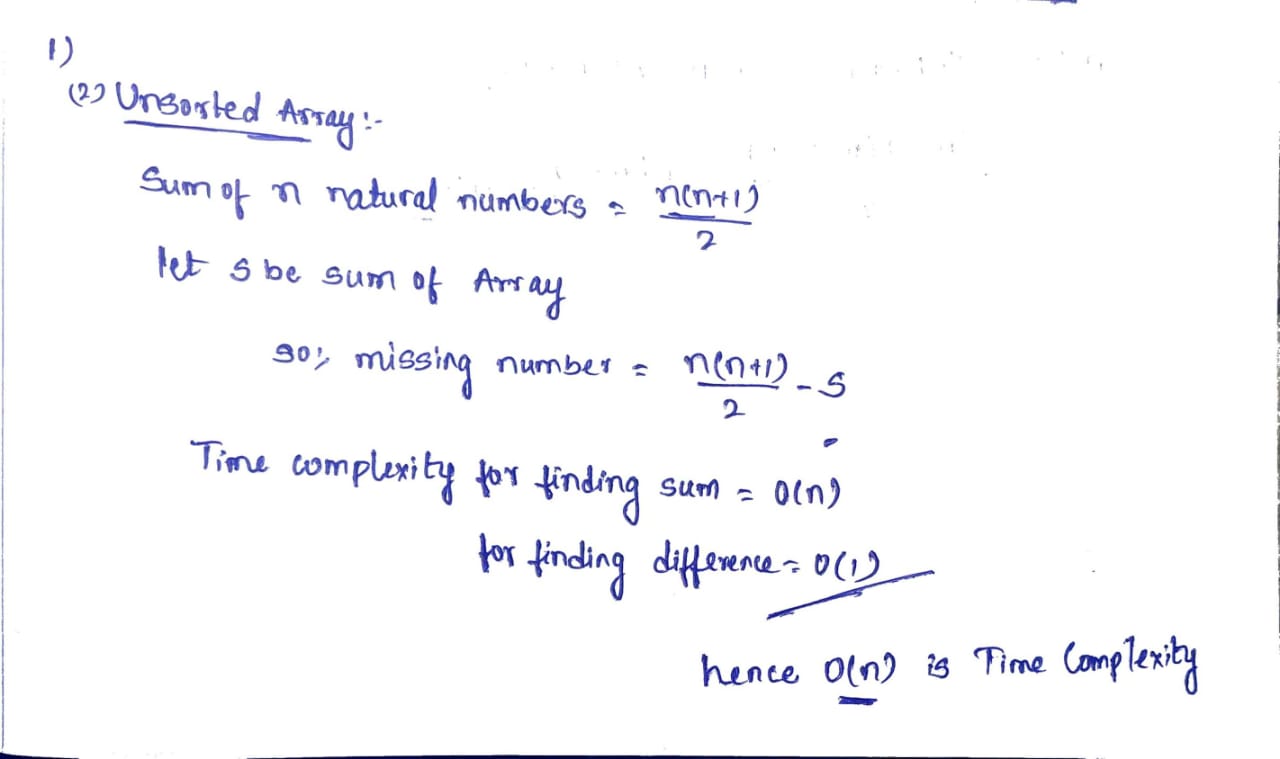
**Code:**

#include <iostream>  
using namespace std;  
int missed(int array[], int n)  
{  
 int a;  
 a = (n + 1) \* (n + 2) / 2;  
 for (int i = 0; i < n; i++)  
 a -= array[i];  
 return a;  
}  
int main()  
{  
 int num;  
 cout << "Enter size of the array :" << endl;  
 cin >> num;  
 int arr[num];  
 cout << "Enter the values:" << endl;  
 for (int i = 0; i < num; i++)  
 {  
 cin >> arr[i];  
 }  
 cout << "\n"  
 << endl;  
 int m = missed(arr, num);  
 cout << "Missing number is: " << m;  
}

**Output:**

****

**Explaination:**

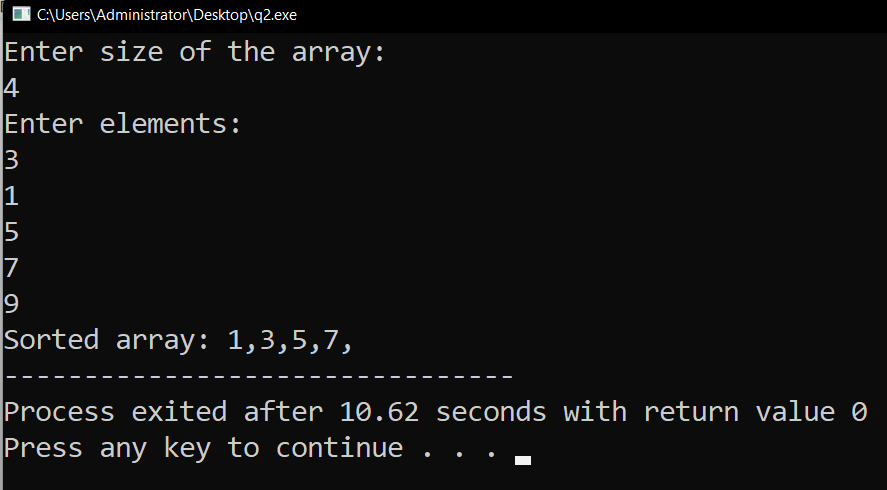


1. **Insertion Sort :**

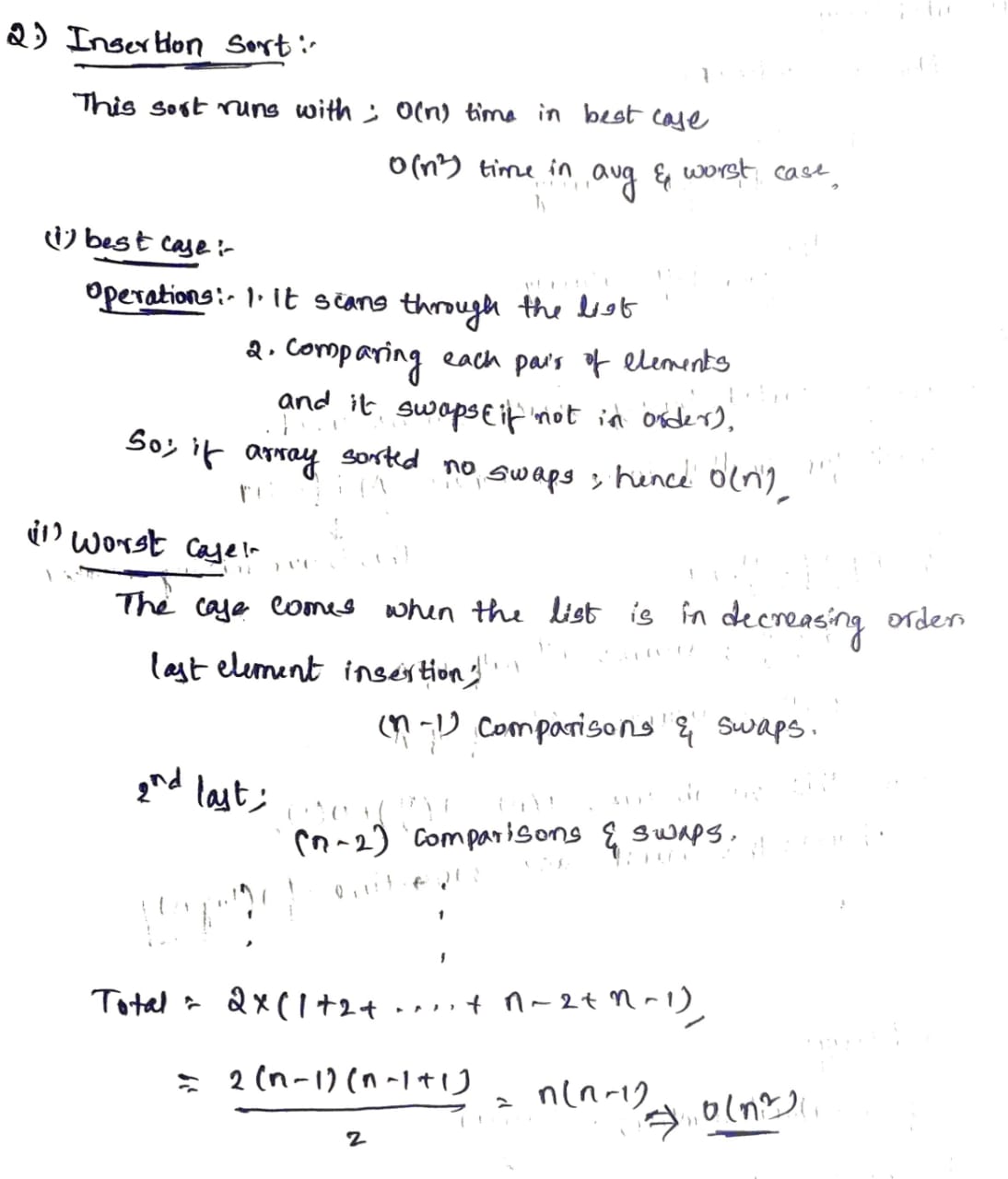
**Code:**

#include <stdio.h>  
int binary(int a[], int e, int h, int l)  
{  
 if (h <= l)  
 return (e > a[l]) ? (l + 1) : l;  
  
 int middleelement = (l + h) / 2;  
  
 if (e == a[middleelement])  
 return middleelement + 1;  
  
 if (e > a[middleelement])  
 return binary(a, e, middleelement + 1, h);  
 return binary(a, e, l, middleelement - 1);  
}  
void insertion(int a[], int n)  
{  
 int i, b, j, k, s;  
  
 for (i = 1; i < n; ++i)  
 {  
 j = i - 1;  
 s = a[i];  
 b = binary(a, s, 0, j);  
 while (j >= b)  
 {  
 a[j + 1] = a[j];  
 j--;  
 }  
 a[j + 1] = s;  
 }  
}  
  
int main()  
{  
 int n, i;  
 printf("enter size \n");  
 scanf("%d", &n);  
 int a[n];  
 for (int i = 0; i < n; i++)  
 {  
 scanf("%d ", &a[i]);  
 }  
  
 insertion(a, n);  
  
 printf("Sorted array: ");  
 for (i = 0; i < n; i++)  
 {  
 printf("%d,", a[i]);  
 }  
 return 0;  
}

**Output:**

****

**Explaination:**



**Improvement:**

