PROPOSED ENHANCED DIVISION SELECTION SORT ALGORITHM

* by ALOKAM NIKHITHA

nikhithaalokam@gmail.com

PROBLEM STATEMENT :







OBJECTIVE:



In this algorithm, sorting is performed in three phase

Phase-1:









(Smallest+largest)/2

Phase-2:







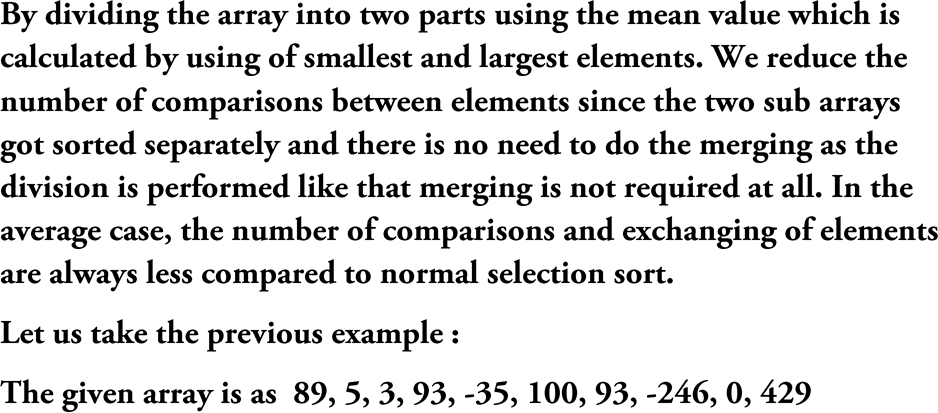


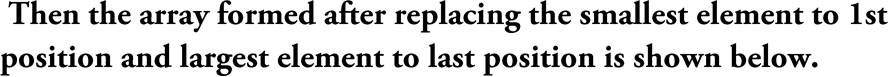
Phase-3:

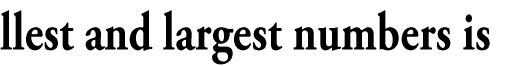








PHASE – 1



|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| -246 | 5 | 3 | 93 | -35 | 100 | 93 | 89 | 0 | 429 |

### PHASE-2







|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |

**PHASE-3**





|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |





|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |





# Normal Selection Sorting:

**CODE:**

**#include<stdio.h>**

**int main()**

**{**

**int n=1000; int nArr[n]; int nMin=0; int a; for(a=0;a<n;a++)**

**{**

**nArr[a]=n-a;**

**}**

**int t=0, i=0, j=0; for(i=0; i<n-1 ;i++)**

**{**

**nMin=i; for(j=i+1; j<n;j++)**

**{**

**if(nArr[j]<nArr[nMin]) nMin = j;**

**}**

**t=nArr[i]; nArr[i]=nArr[nMin]; nArr[nMin]=t;**

**}**

**printf("\n\nSorted Array:");**

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

**{**

**printf(" %d", nArr[i]);**

**}**

**printf("\n"); return 0;**

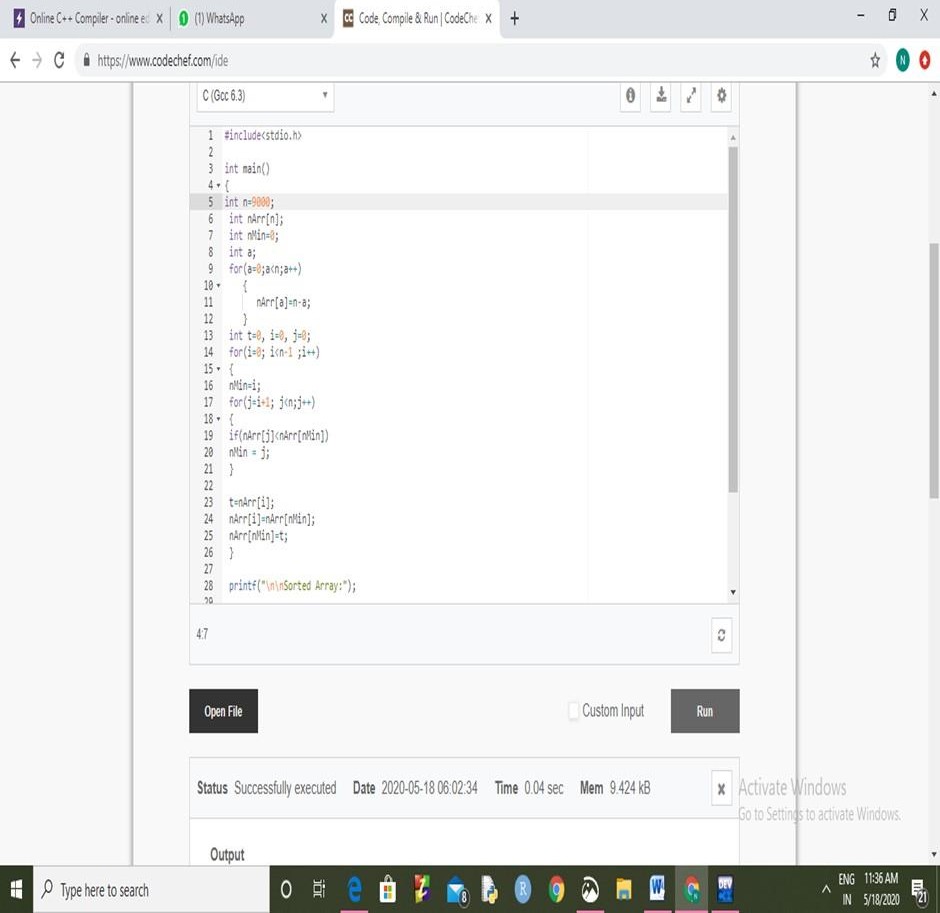
**}**

**RESULTS:**

**FIGURE :1**

**n=9000**

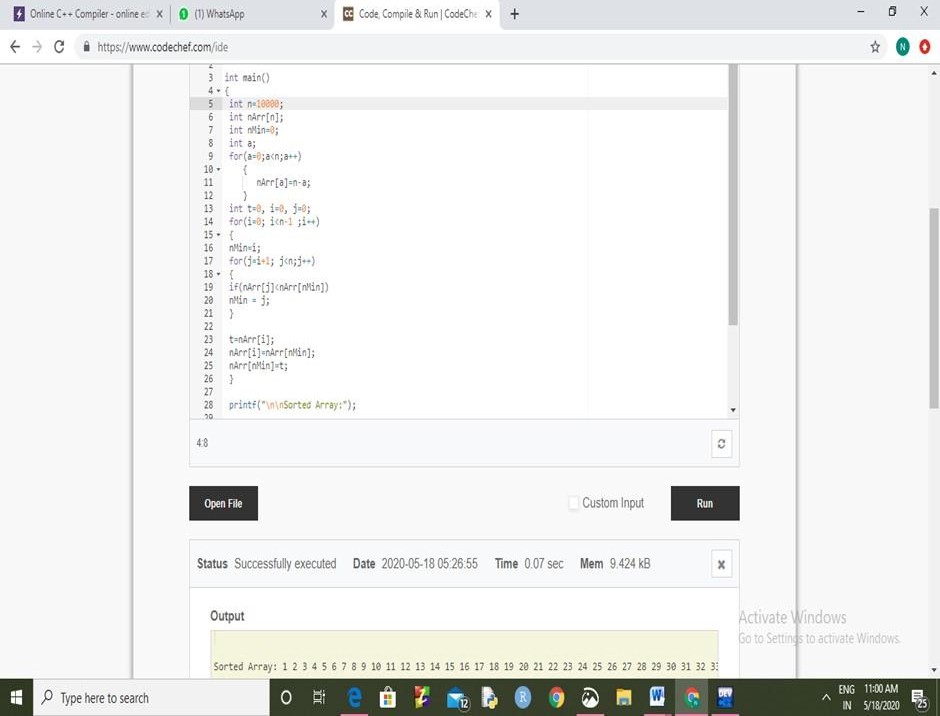
**Time complexity:0.04 sec**



# FIGURE:2

**n=10000**

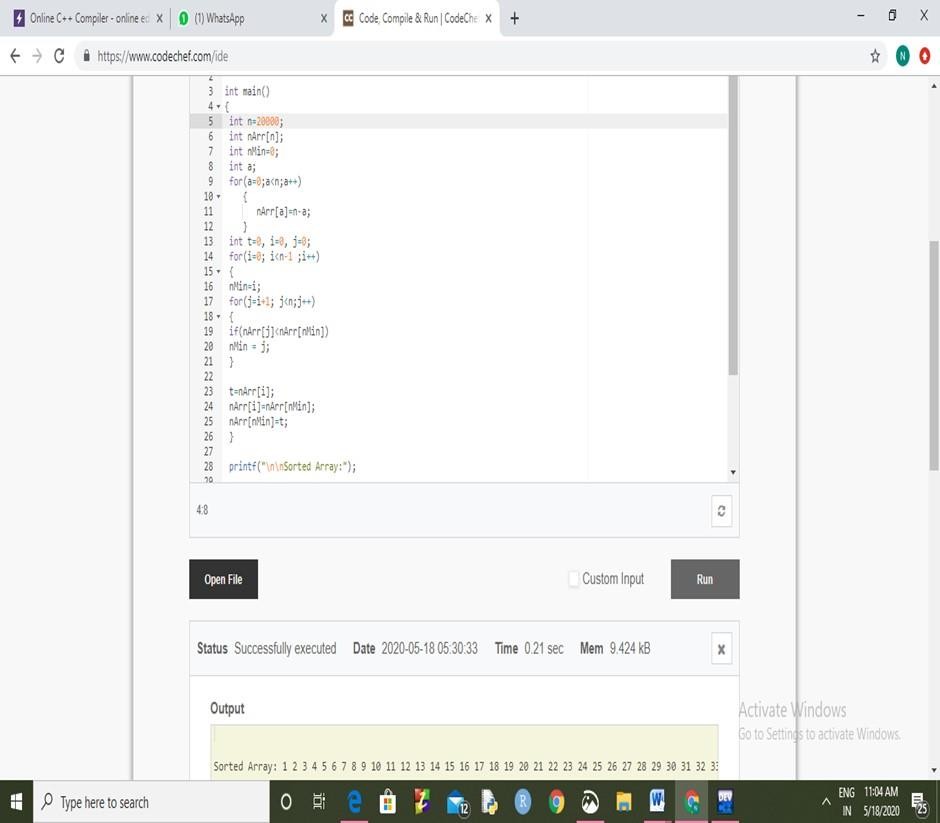
**Time Complexity:0.07**



# FIGURE:3

**n=20000**

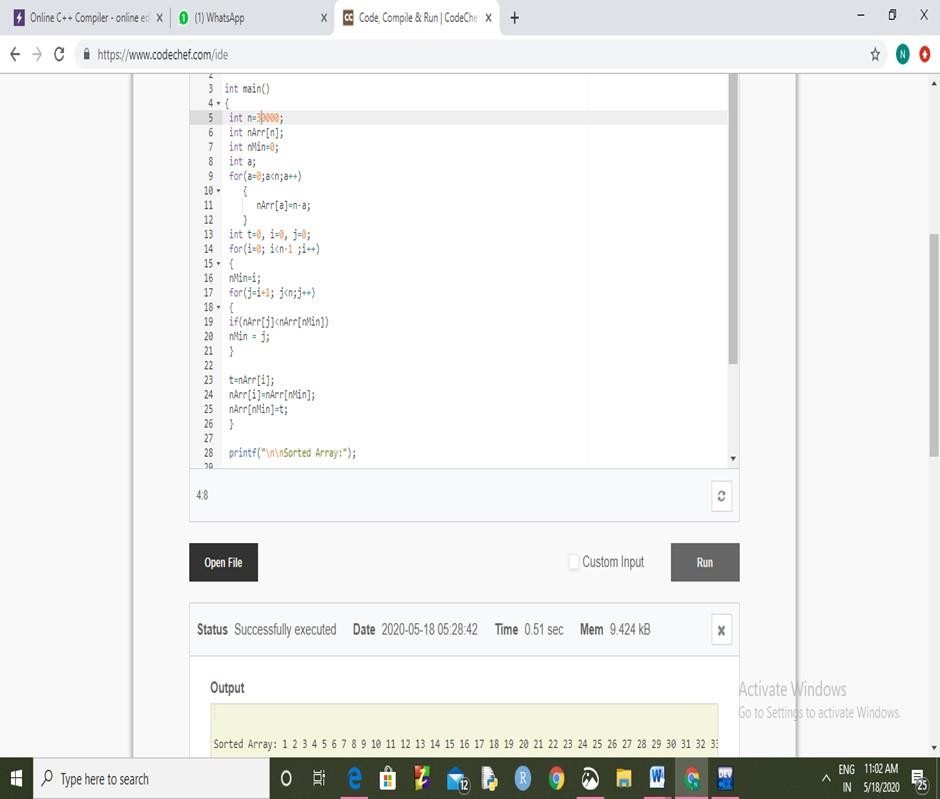
**Time complexity: 0.21 sec**



# FIGURE:4

**n=30000**

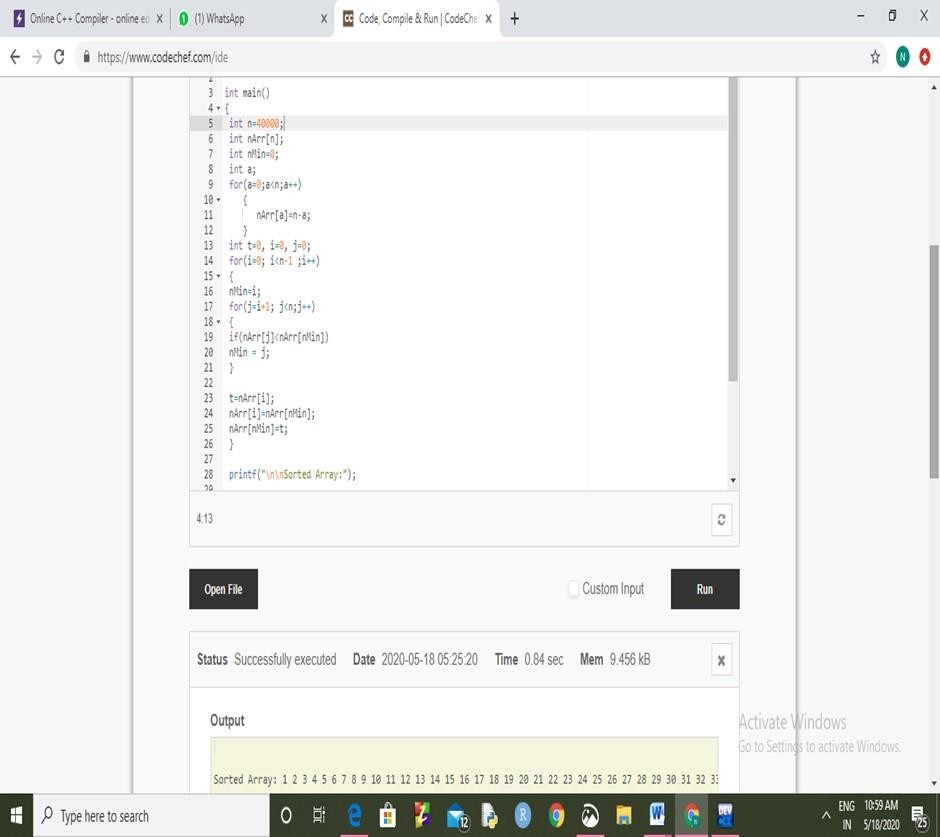
**Time Complexity: 0.51 sec**



# FIGURE :5

**n=40000**

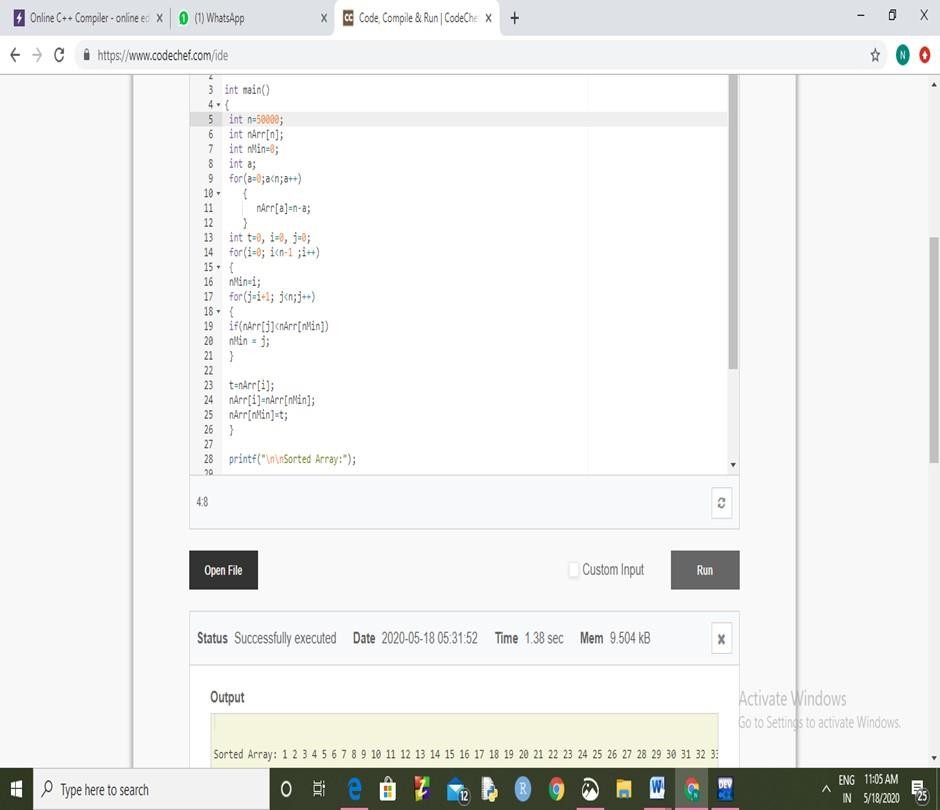
**Time Complexity: 0.84 sec**



# FIGURE:6

**n=50000**

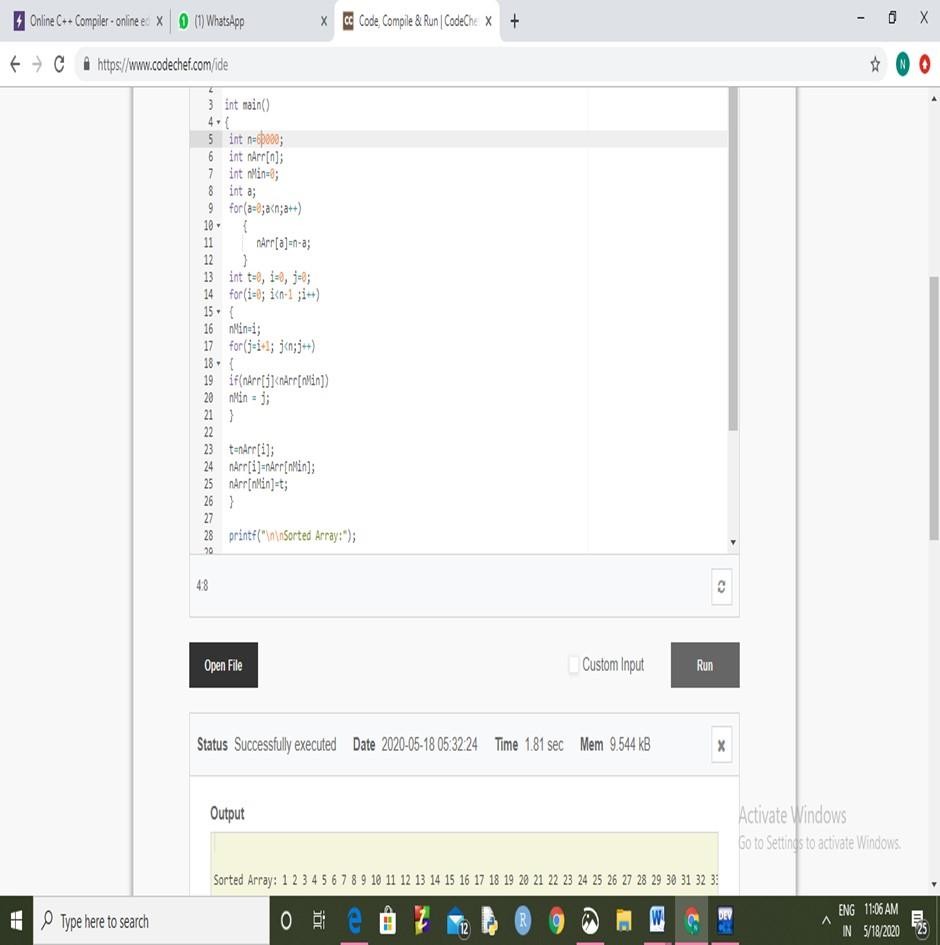
**Time complexity: 1.38 sec**



# FIGURE:7

**n=60000**

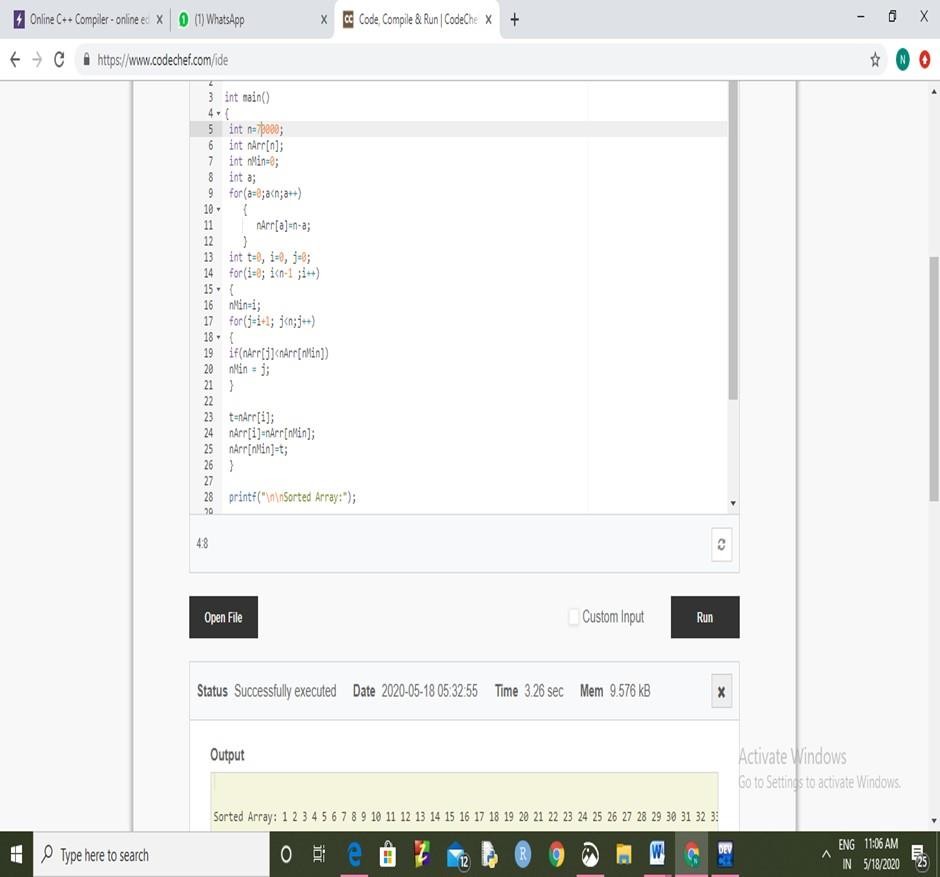
**Time Complexity: 1.81 sec**



# FIGURE:8

**n=70000**

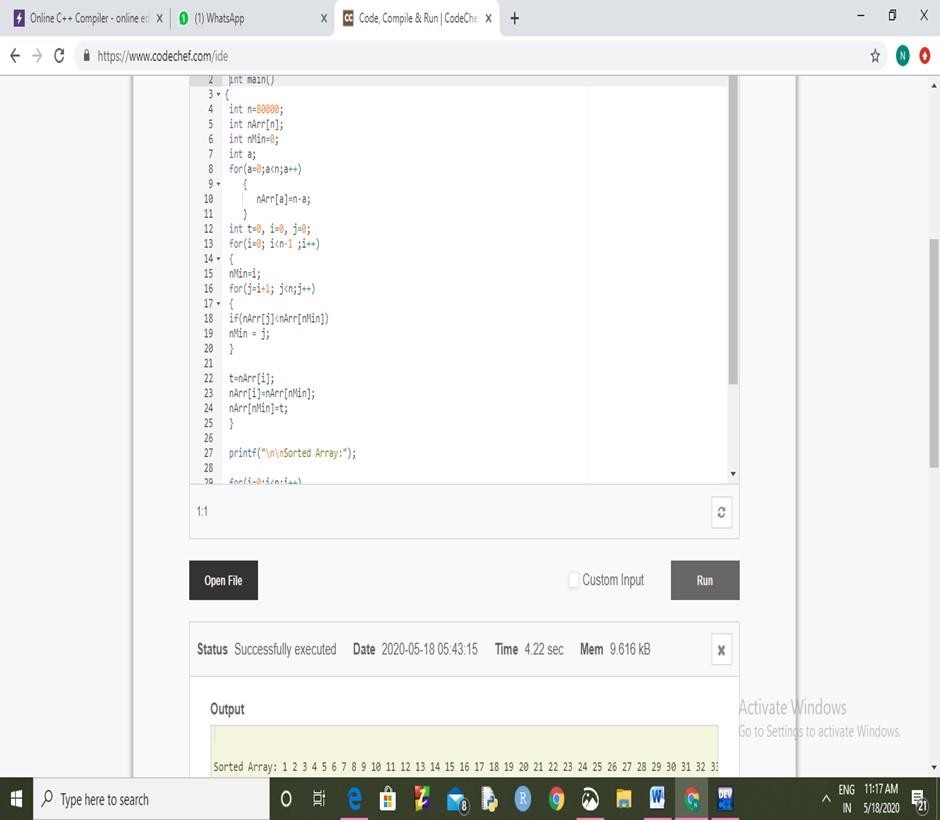
**Time Complexity:3.26 sec**



# FIGURE:9

**n=80000**

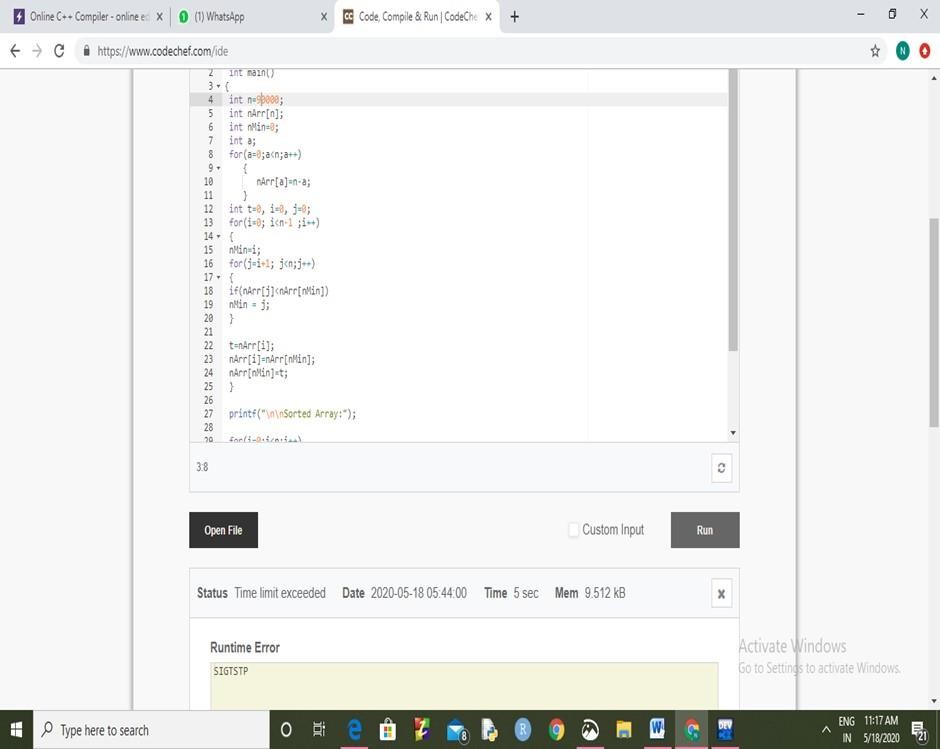
**Time Complexity: 4.22 sec**



# FIGURE:10

**n=90000**

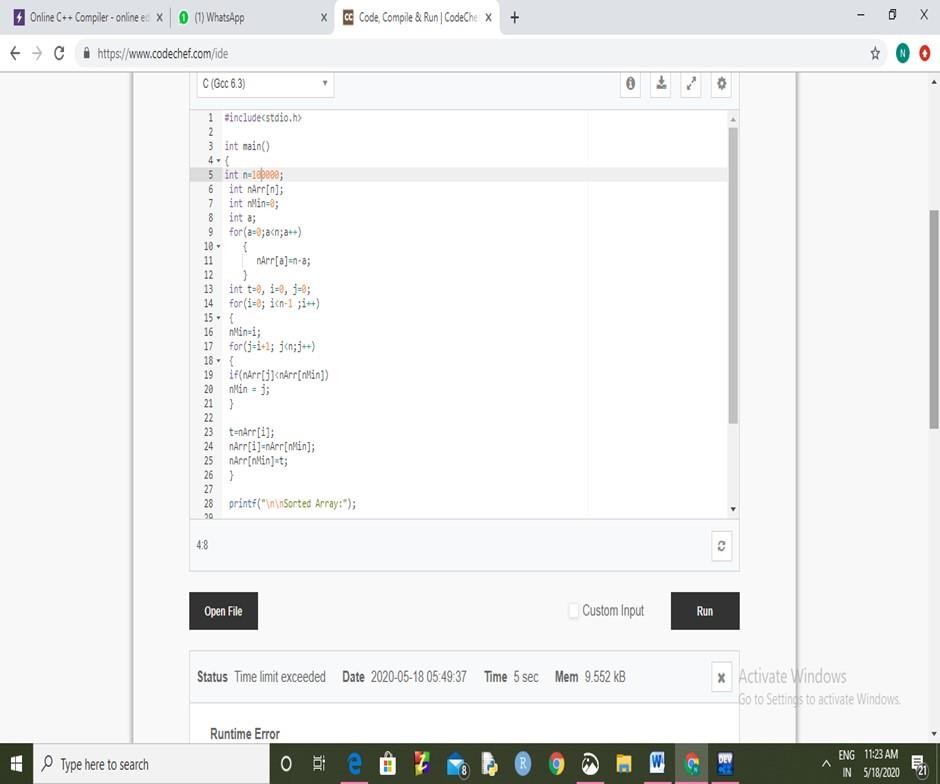
**Time Complexity: >5 sec**



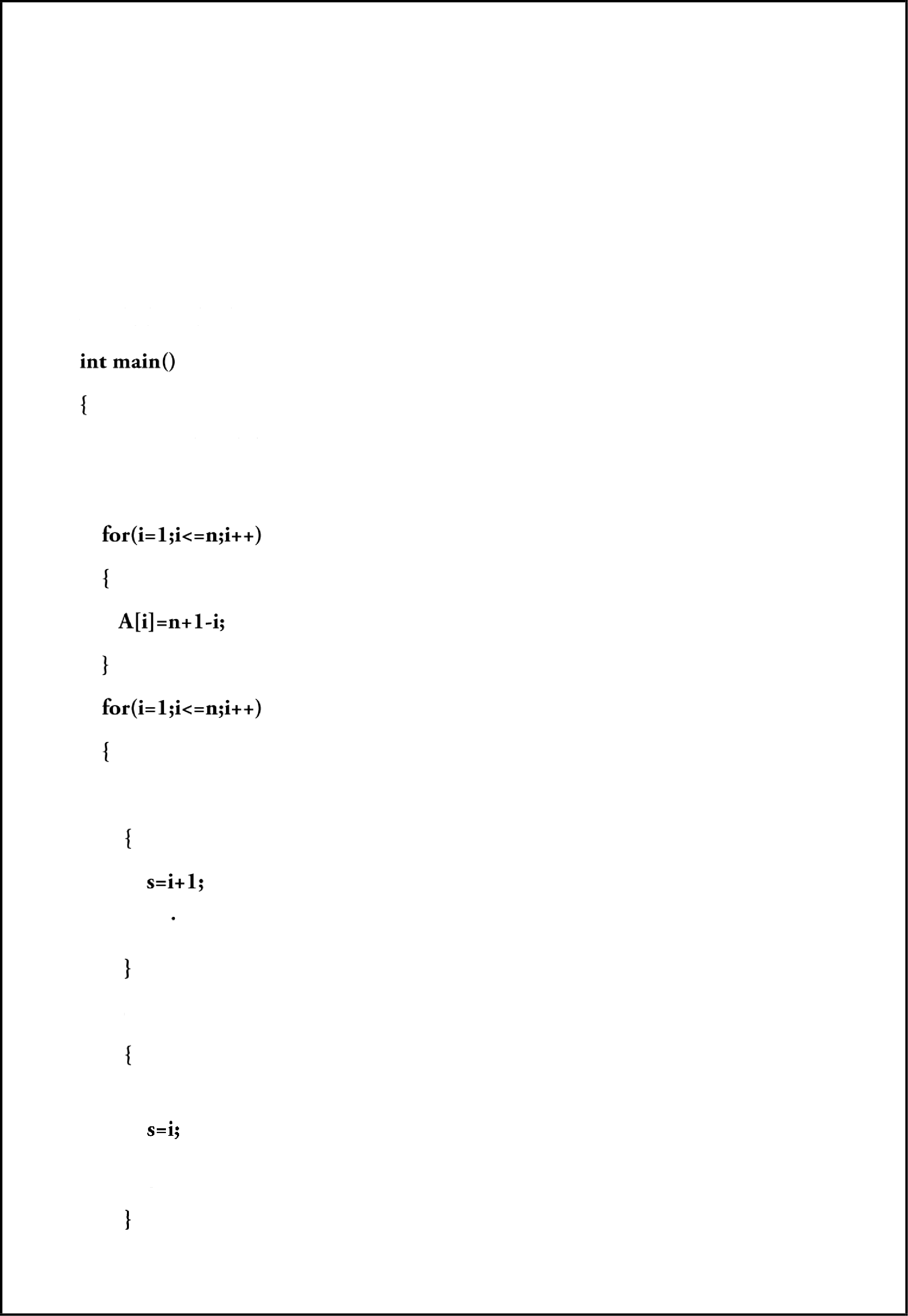
# FIGURE:11

**n=100000**

**Time Complexity:>5sec**







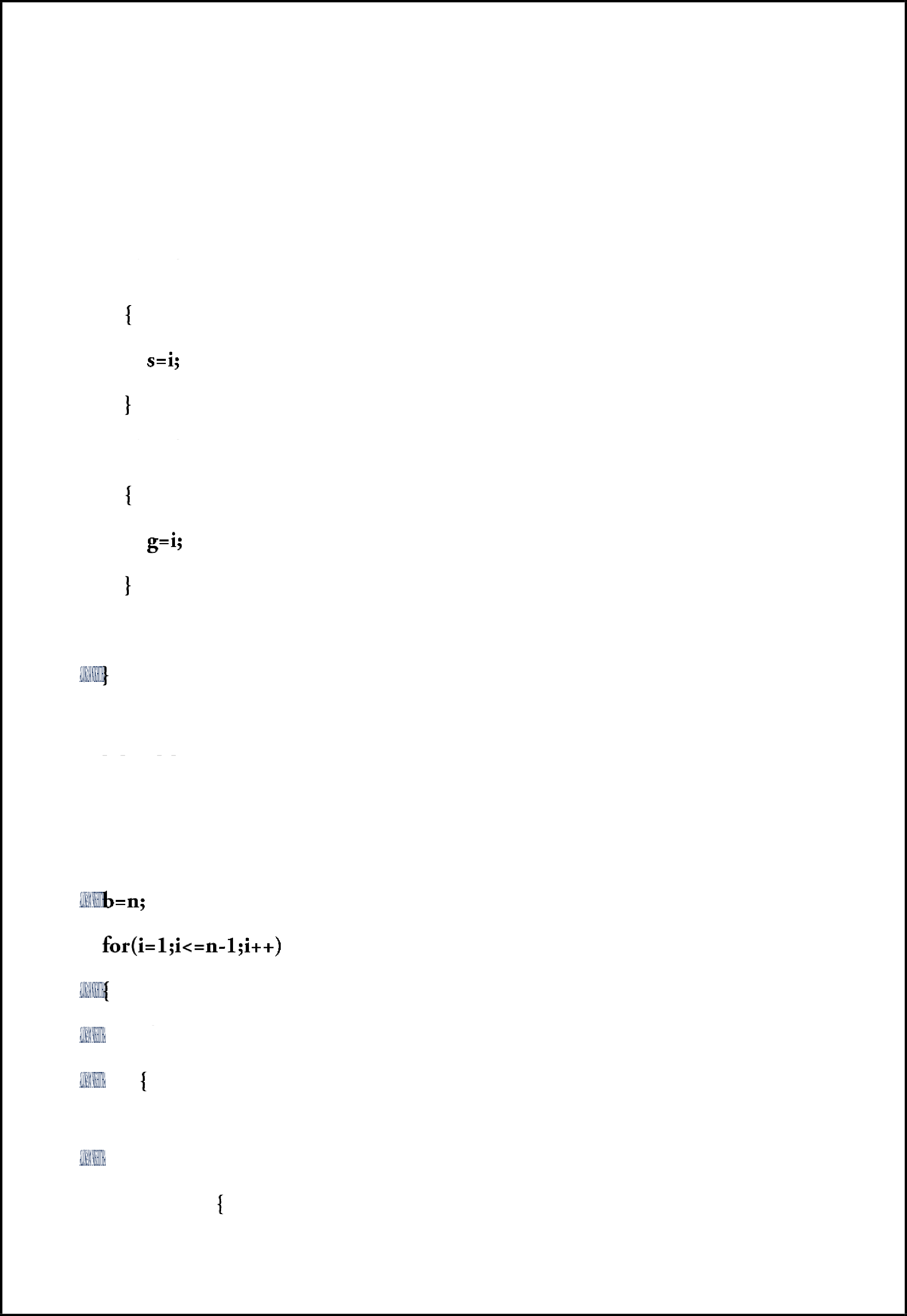
CODE:

#inc1ude<stdio.h>

int i,j,s,g,b,m,k,h,A[1000]; int n=1000;

i£((i==1)&&(A[i]>A[i+1]))

else if((i==1)&&(A[i] <A[i+1]))



else if(A[i] <=A[s])

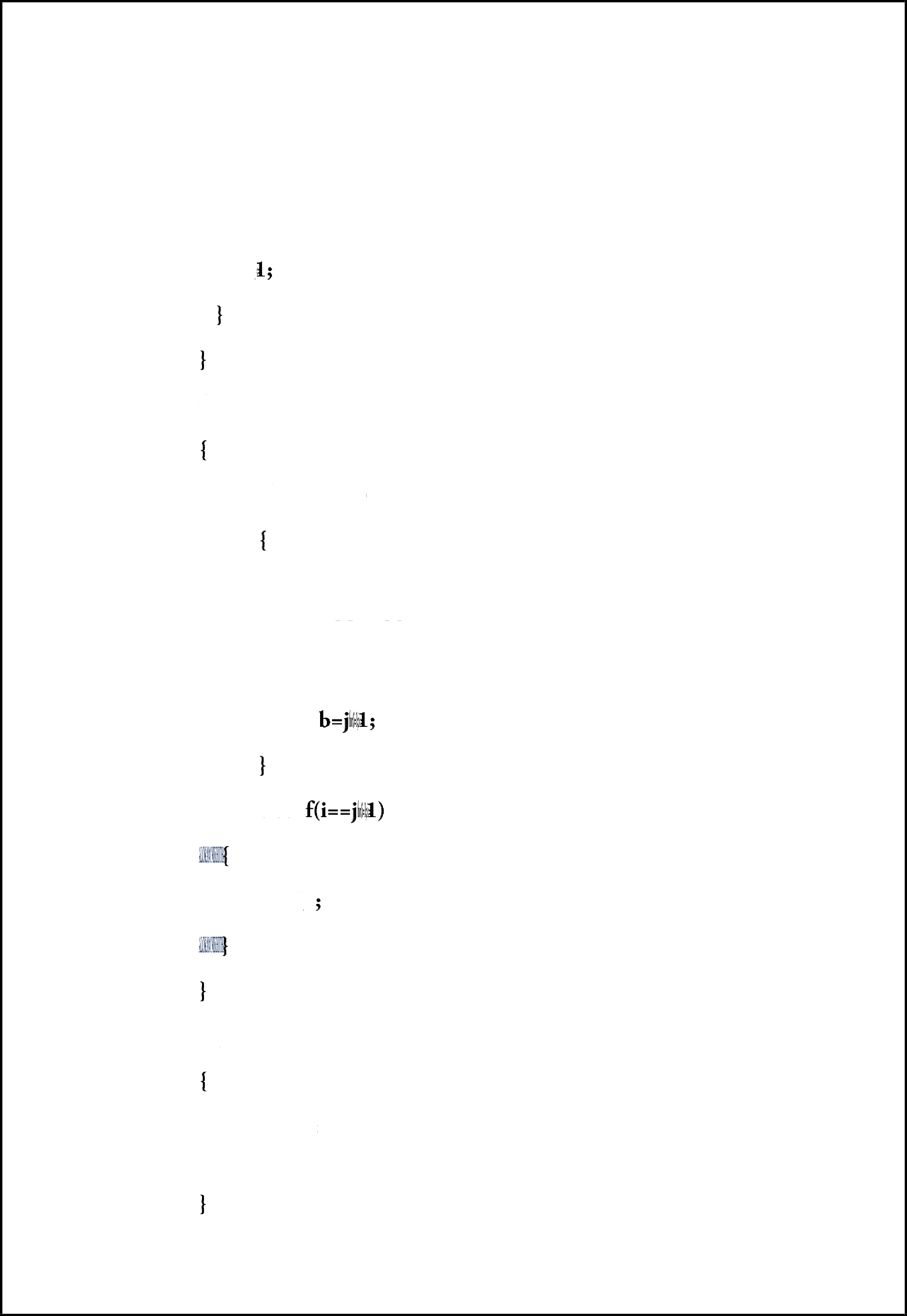
else if(A[i] >=A[g])

k-A[1]; A[1]-A[s]; A[s] =k;

m=(A[1]+A[g])/2j

if(A[i] >m)

if(i--b)



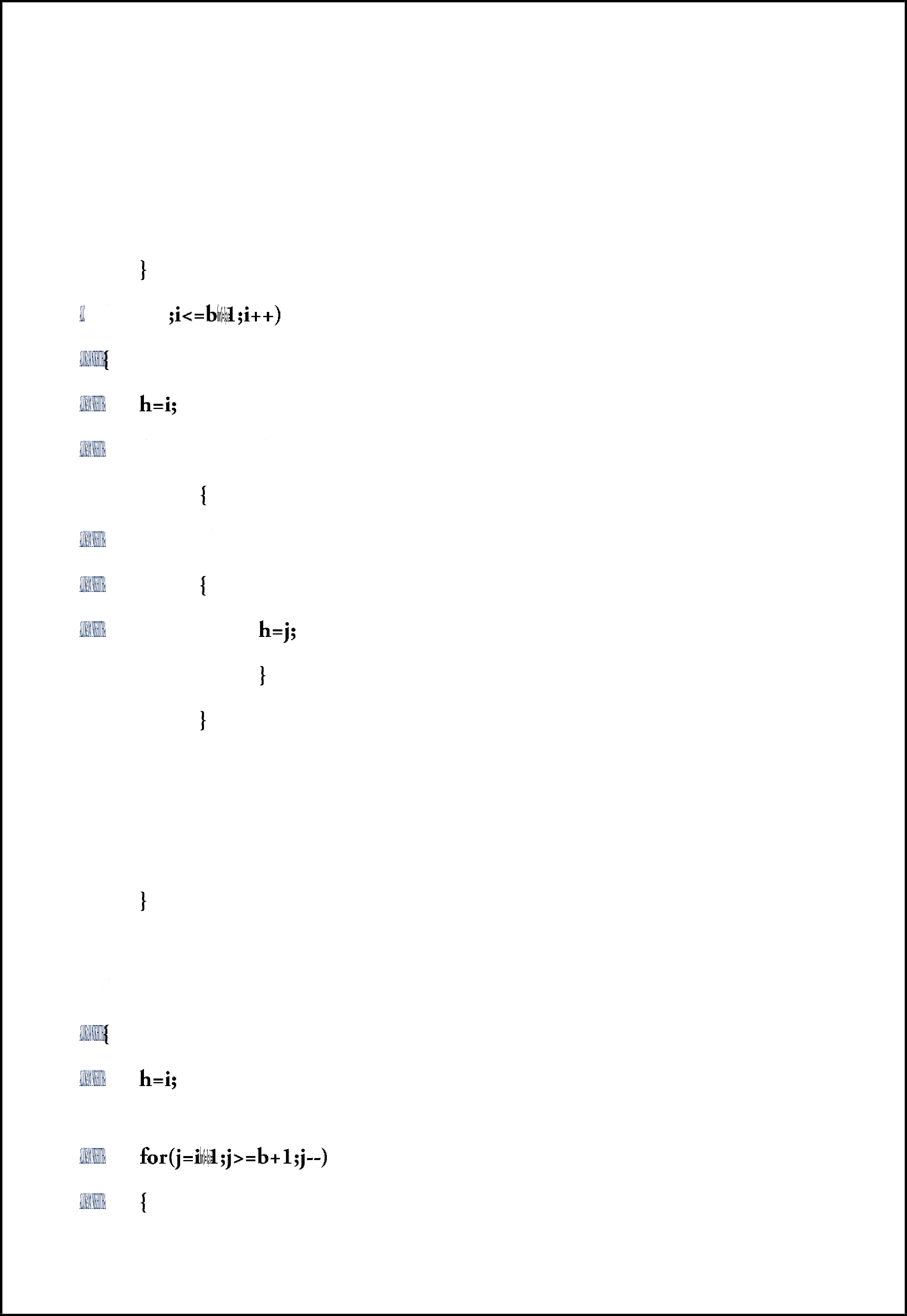
for(j=b;j>=i;j--)

if(A[j] <=m)

k=A[i]; A[i]=A[j]; A[j]=k;

else i

breaJc;



Vitor(i=2

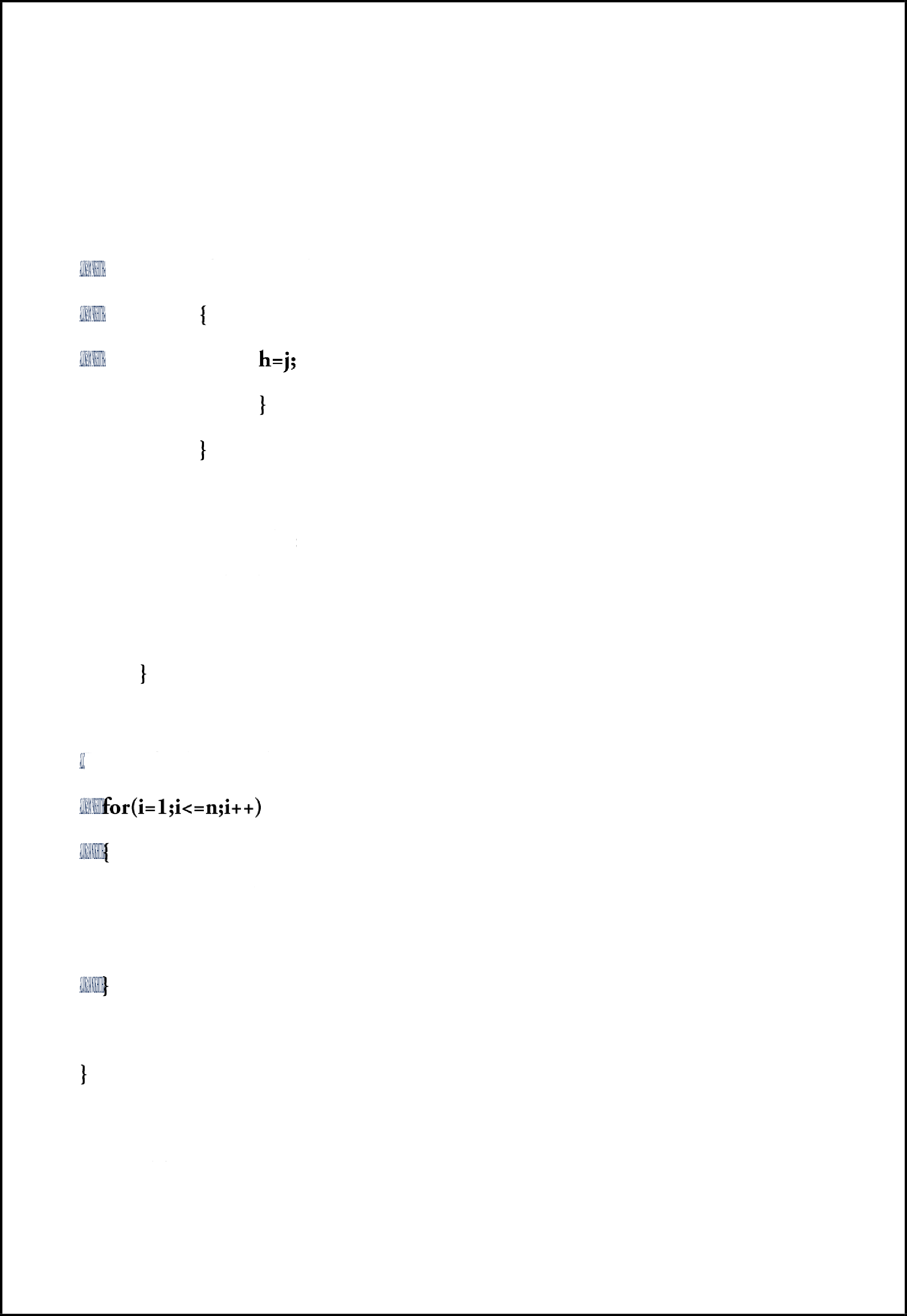
for(j=i+1;j<=b;j++)

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

k=A[i]; A[i]=A[h];

A[h]-k;

for(i=n;i>b+2;i--)



if(A[j] >=A[h])

k=A[i]; A[i]=A[h];

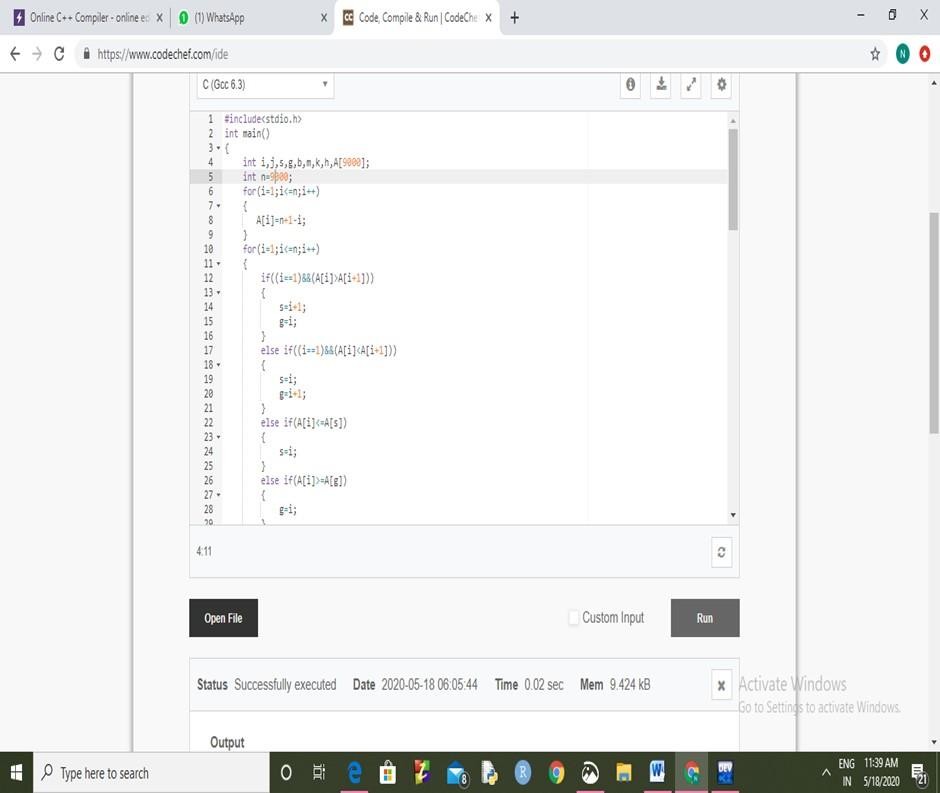
A[h]-k;

I rintf("the sorted array is:");

printf("\n %d",A[i]);

RESULTS-

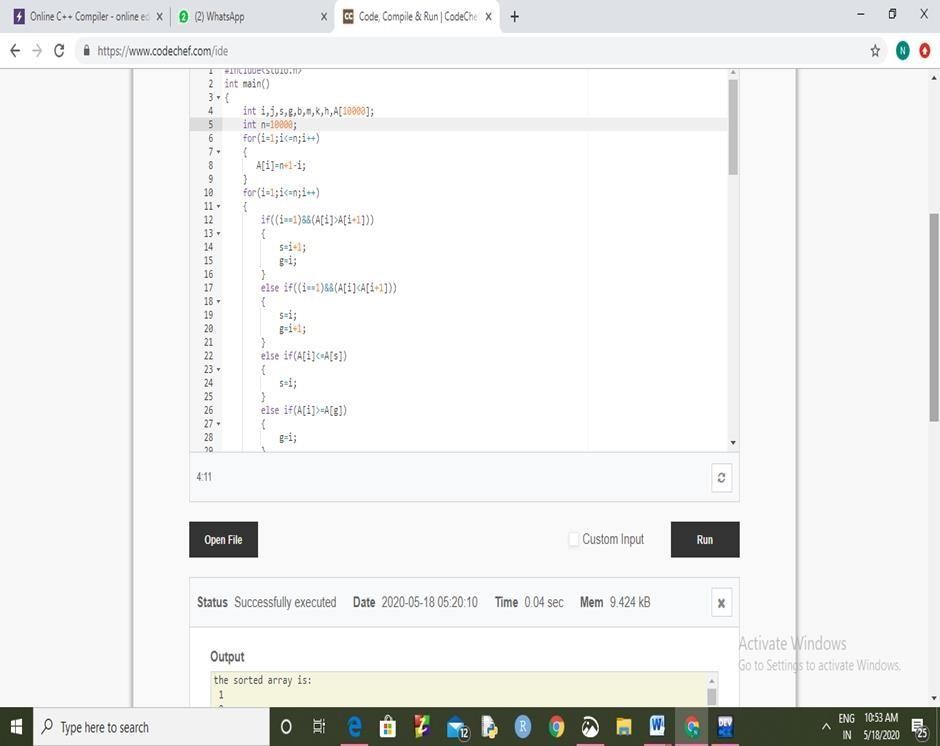
**FIGURE:1 n=9000**

**Time Complexity:0.02 sec**

## FIGURE:2

n=10000

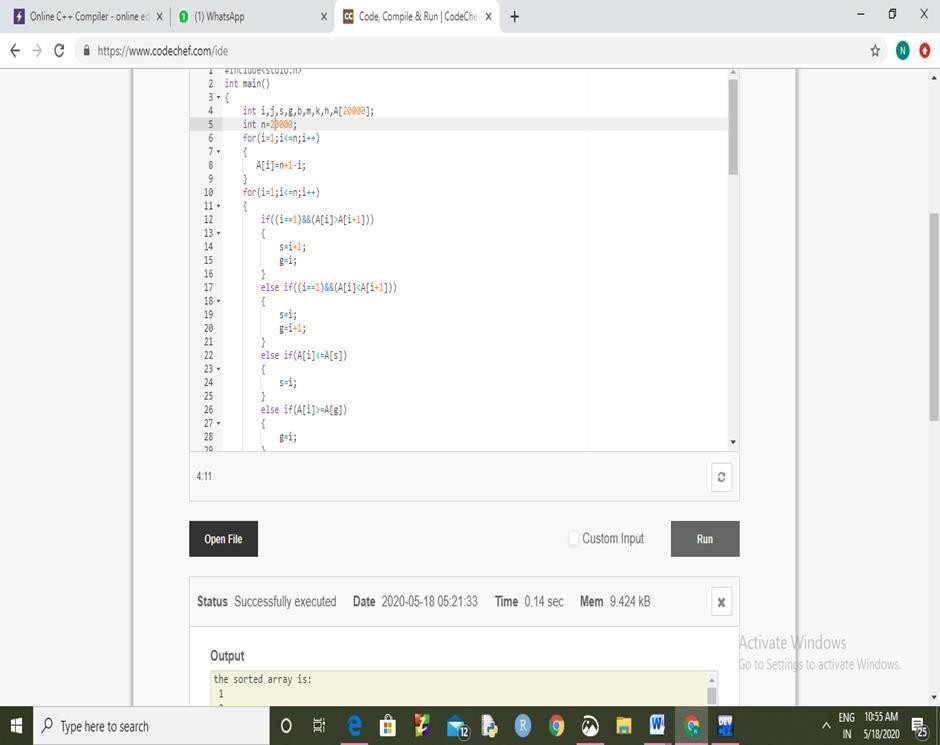
Time Complexity:0.04 sec



## FIGURE:3

n=20000

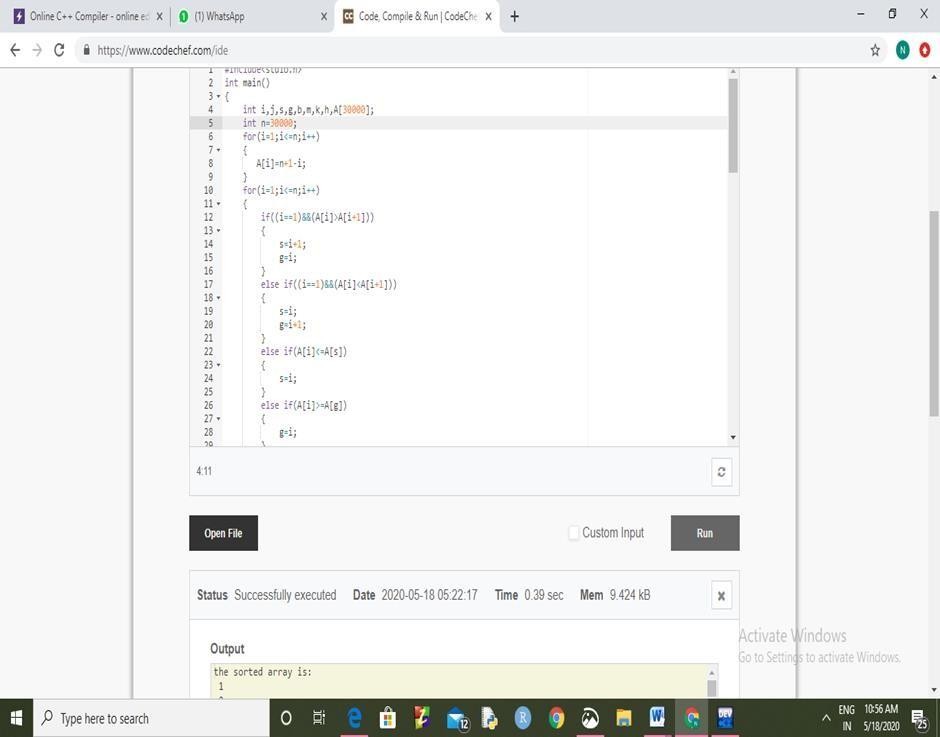
Time Complexity:0.14 sec



## FIGURE:4

n=30000

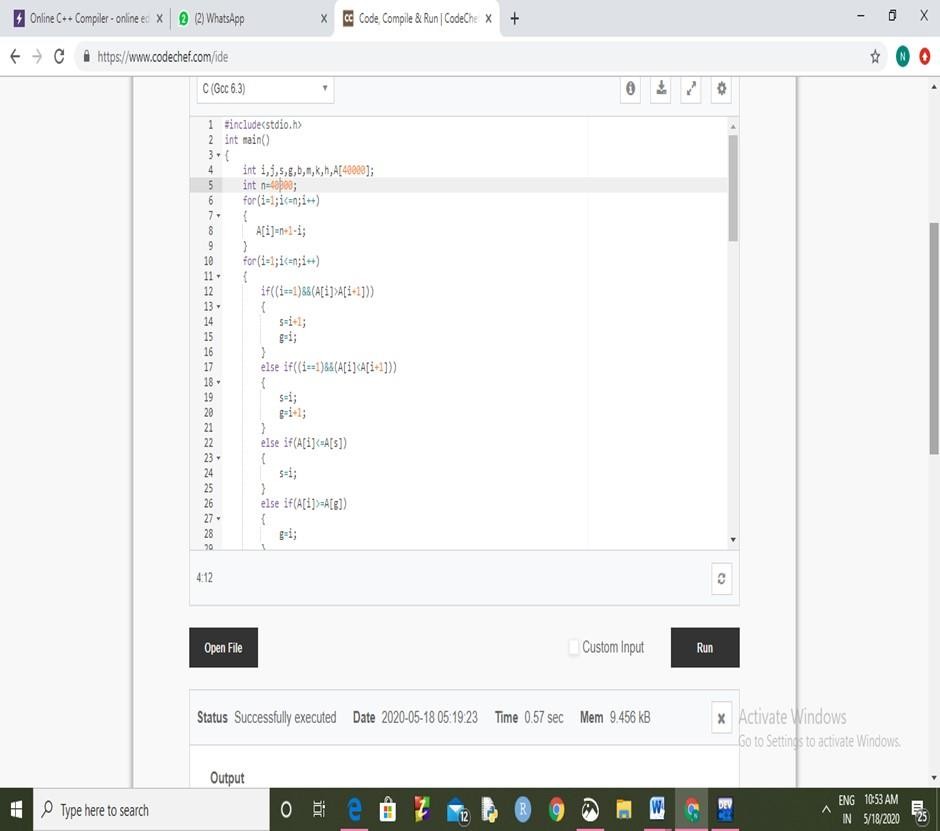
Time Complexity:0.39 sec



## FIGURE:5

n=40000

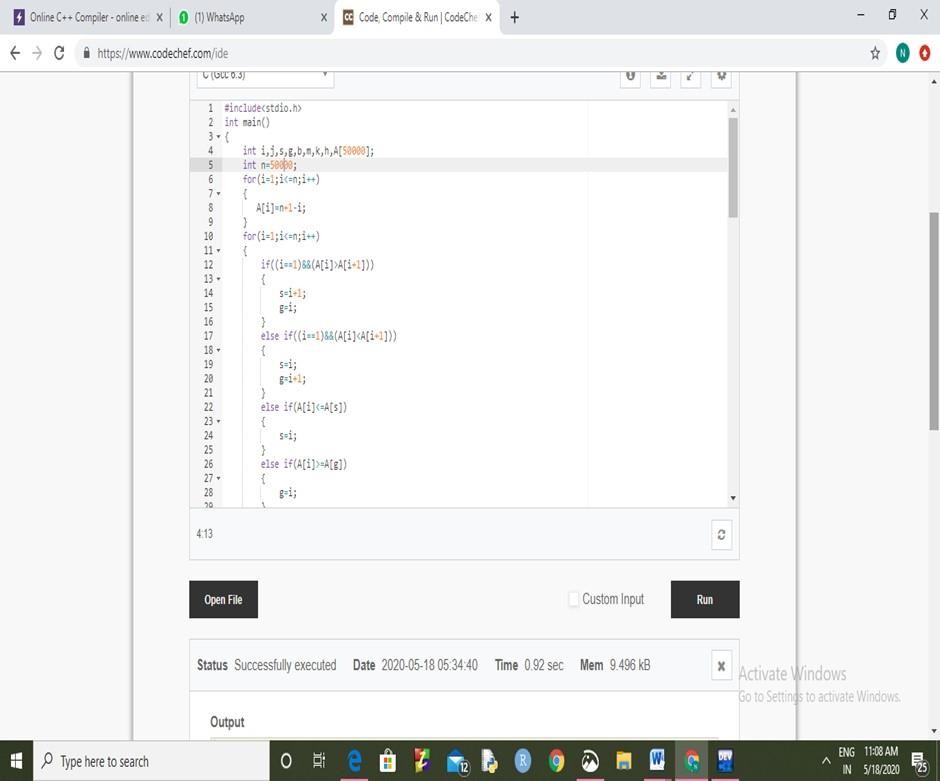
Time Complexity:0.57 sec



## FIGURE:6

n=50000

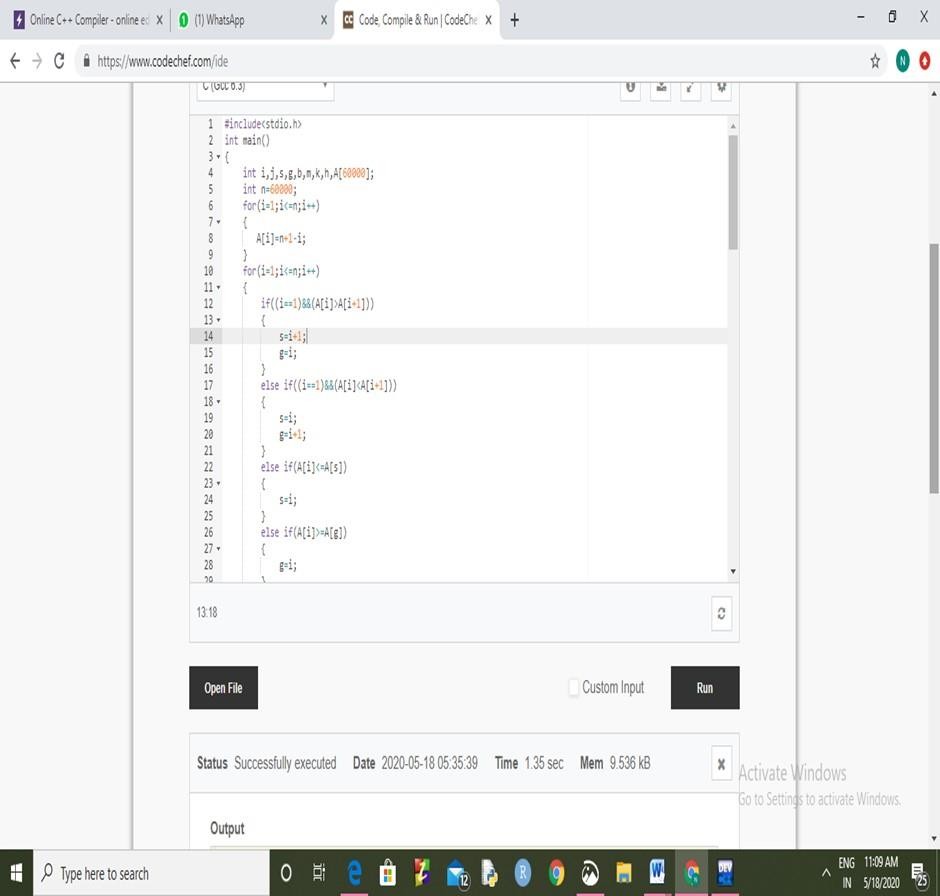
Time Complexity:0.92 sec



## FIGURE:7

n=60000

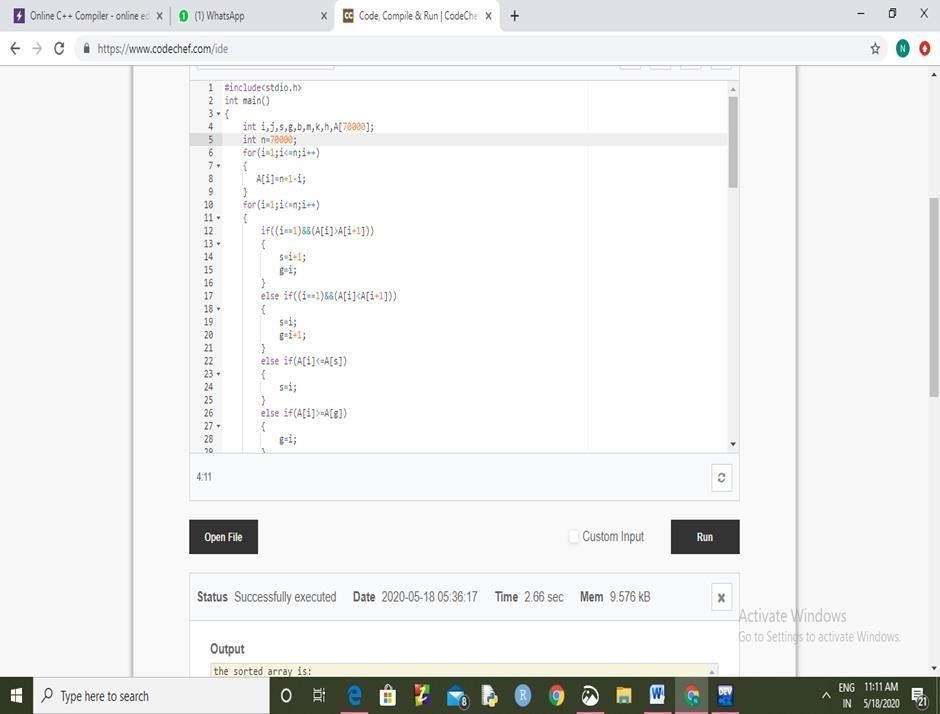
Time Complexity:1.35 sec



## FIGURE:8

n=70000

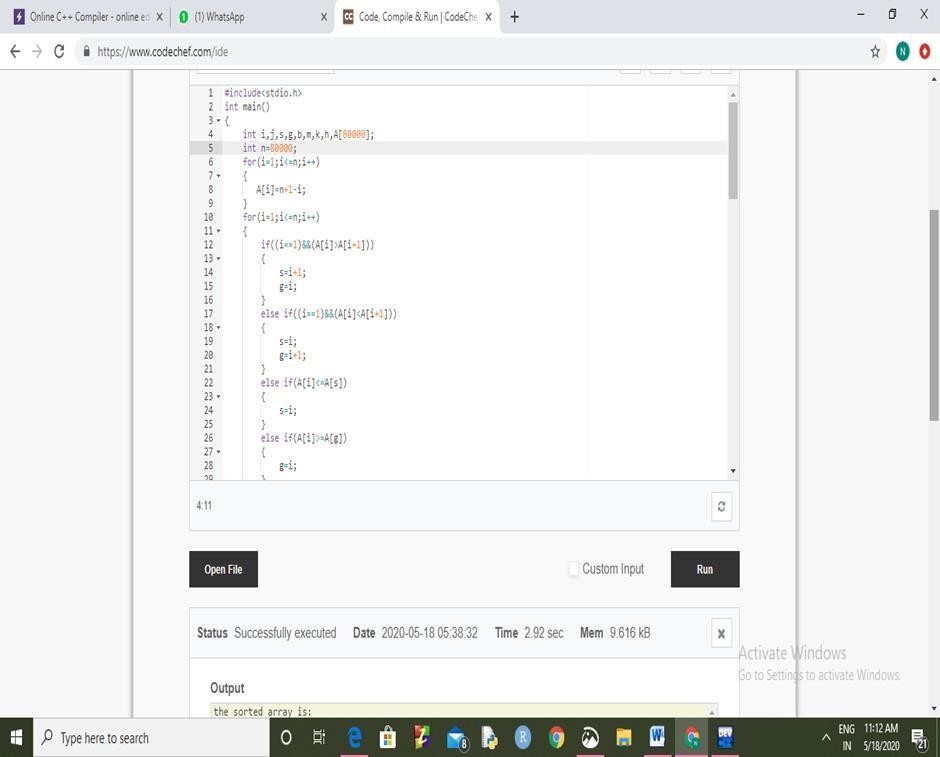
Time Complexity:2.66sec



## FIGURE:9

n=80000

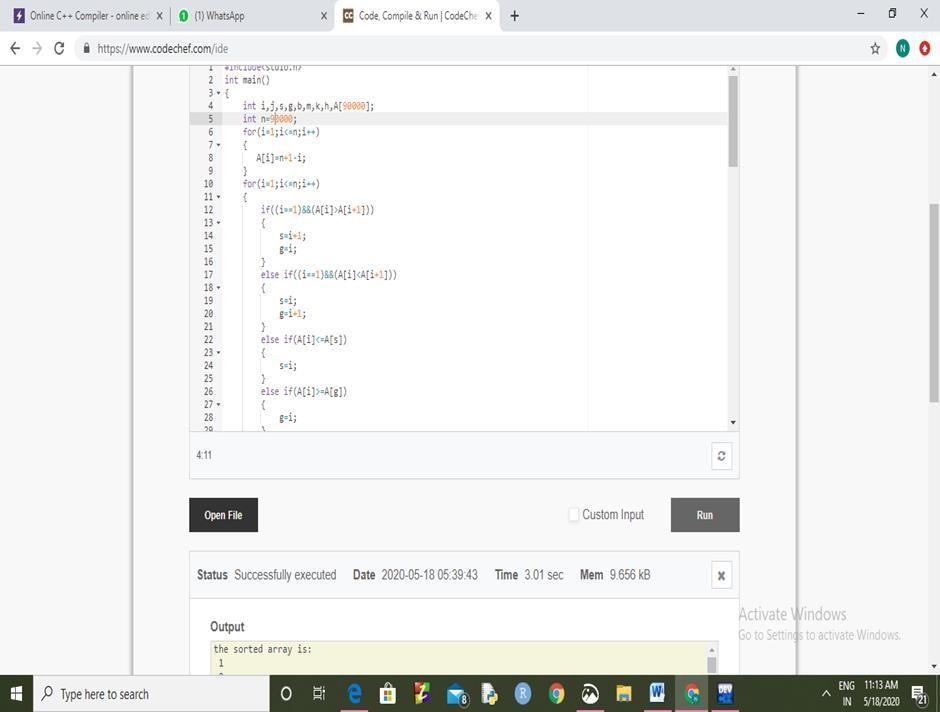
Time Complexity:2.92sec



## FIGURE:10

n=90000

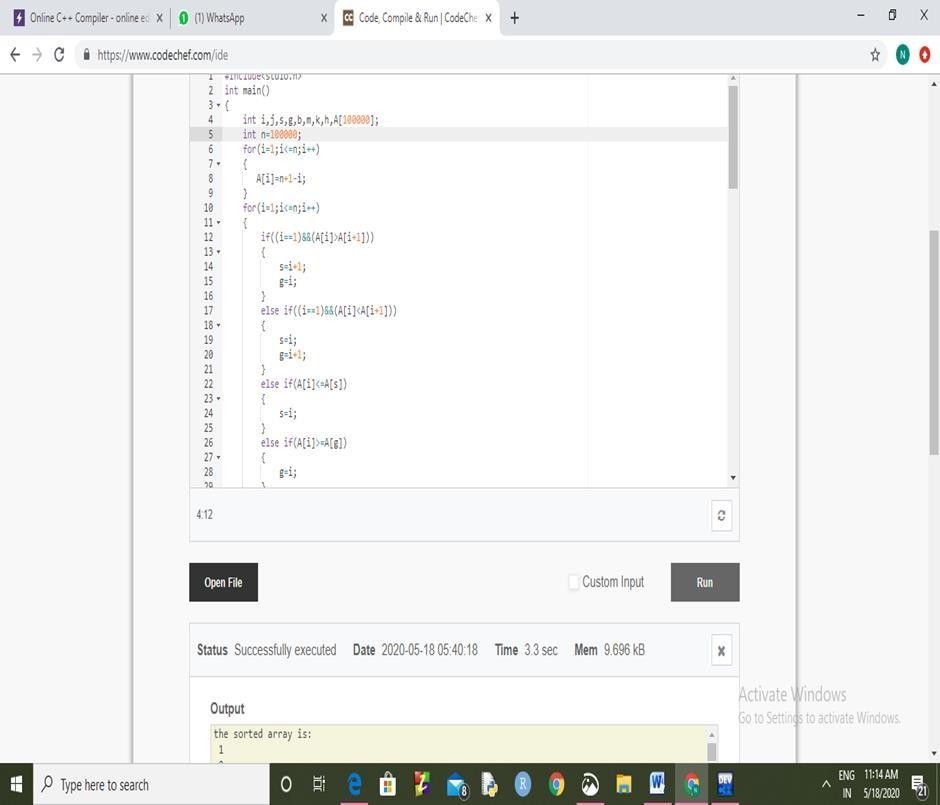
Time Complexity:3.01sec



## FIGURE:11

n=100000

Time Complexity:3.3 sec



ANALYSIS AND RESULTS.

Both algorithms are compared on the same elements of unordered list. In order to make a comparison of the proposed algorithms with the existing Selection sort, a number of tests were conducted for small as well as large number of elements. A Comparative study of execution time for the number of inputs is shown in tabular form as in Table-1 below

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 9000 | 0.04 | 0.02 | 0.02 |
| 10000 | 0.07 | 0.04 | 0.03 |
| 20000 | 0.21 | 0.14 | 0.07 |
| 30000 | 0.51 | 0.39 | 0.12 |
| 40000 | 0.84 | 0.57 | 0.27 |
| 50000 | 1.38 | 0.92 | 0.46 |
| 60000 | 1.81 | 1.35 | 0.46 |
| 70000 | 3.26 | 2.66 | 0.6 |
| 80000 | 4.22 | 2.92 | 1.3 |
| 90000 | 5 | 3.01 | 1.99 |
| 100000 | >5 | 3.3 | >1.99 |

**Table :1 Comparative study of execution time of Selection Sort and Enhanced Division Selection Sort**



6

5

4

3

2

1

0

9000 10000 20000 30000 40000 50000 60000 70000 80000 90000

Number of elements selection sort(in sec)

optimized selection sort (in sec)

Fig-1: Number of inputs vs. CPU time in (sec)

#### In Fig-1, the X Axis shows the number of elements and the Y Axis shows the time elapsed in seconds.

CONCLUSION:









References:

1. Arora Nitin, Kumar vivek and Kumar Suresh. “A Novel Sorting Algorithm and Comparison with Bubble Sort and Insertion Sort,”

International Journal of Computer Applications (0975-8887) vol. 45, No. 1, May 2012.

1. Sareen Pankaj, “ Comparison of Sorting Algorithms ( On the Basis of Average case)”, International Journal of Advanced Research in Computer Science and software Engineering ISSn: 2277128x, volume 3, Issue 3, March 2013, pp. 522-532