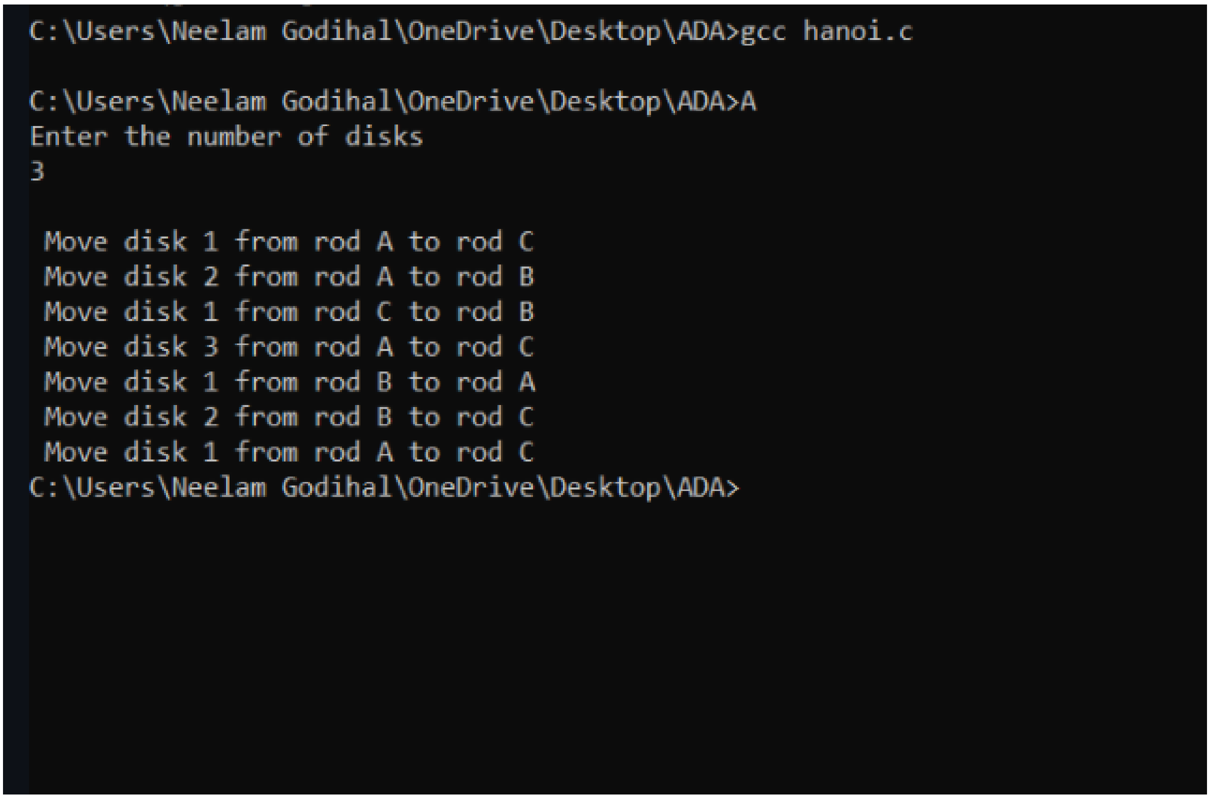
1st LAB CYCLE PROGRAMS:

1)Write a recursive program to

a. Solve Towers-of-Hanoi problem

b. To find GCD.

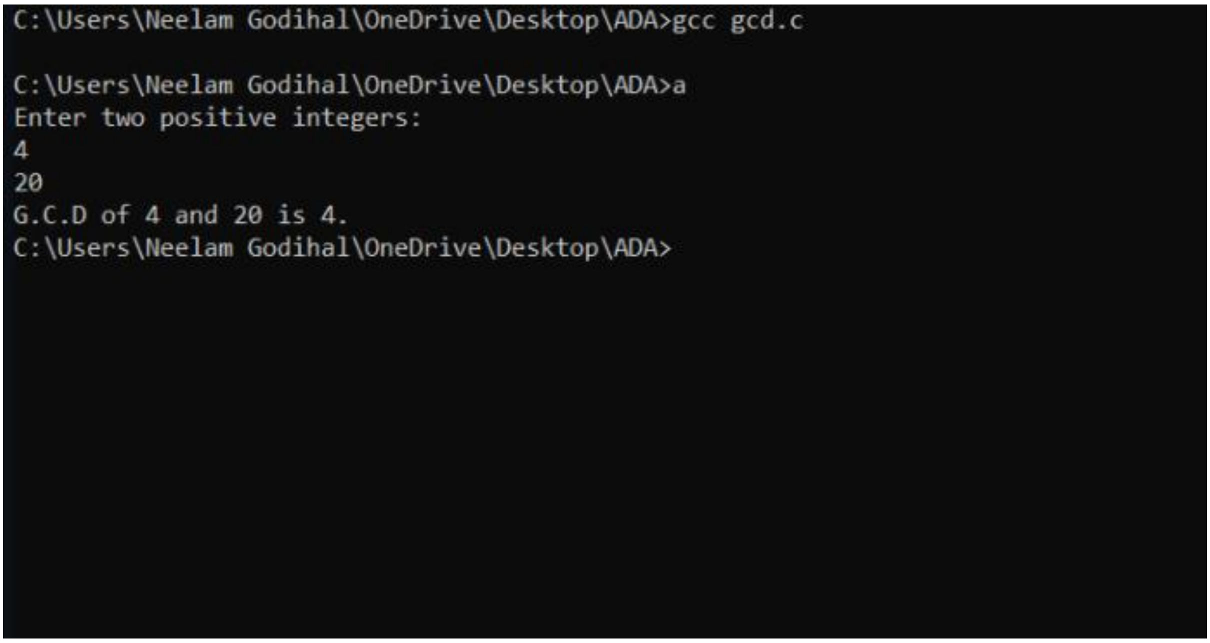
```
a. #include <stdio.h>

void hanoi(int n, char from, char to, char aux)
{
    if (n == 1)
    {
        printf("\n Move disk 1 from rod %c to rod %c", from, to);
        return;
    }
    hanoi(n-1, from, aux, to);
    printf("\n Move disk %d from rod %c to rod %c", n, from, to);
    hanoi(n-1, aux, to, from);
}

int main()
{
    int n;
            printf("Enter the number of disks\n");
            scanf("%d",&n);
    hanoi(n, 'A', 'C', 'B');  //names of the rods
    return 0;
}
```



```
C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>gcc hanoi.c

C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>A
Enter the number of disks
3

 Move disk 1 from rod A to rod C
 Move disk 2 from rod A to rod B
 Move disk 1 from rod C to rod B
 Move disk 3 from rod A to rod C
 Move disk 1 from rod B to rod A
 Move disk 2 from rod B to rod C
 Move disk 1 from rod A to rod C
C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>
```

b. #include <stdio.h>

```
int gcd(int n1, int n2);

int main() {
    int n1, n2;
    printf("Enter two positive integers:\n");
    scanf("%d %d", &n1, &n2);
    printf("G.C.D of %d and %d is %d.", n1, n2, gcd(n1, n2));
    return 0;
}

int gcd(int n1, int n2) {
    if (n2 != 0)
        return gcd(n2, n1 % n2);
    else
        return n1;
}
```



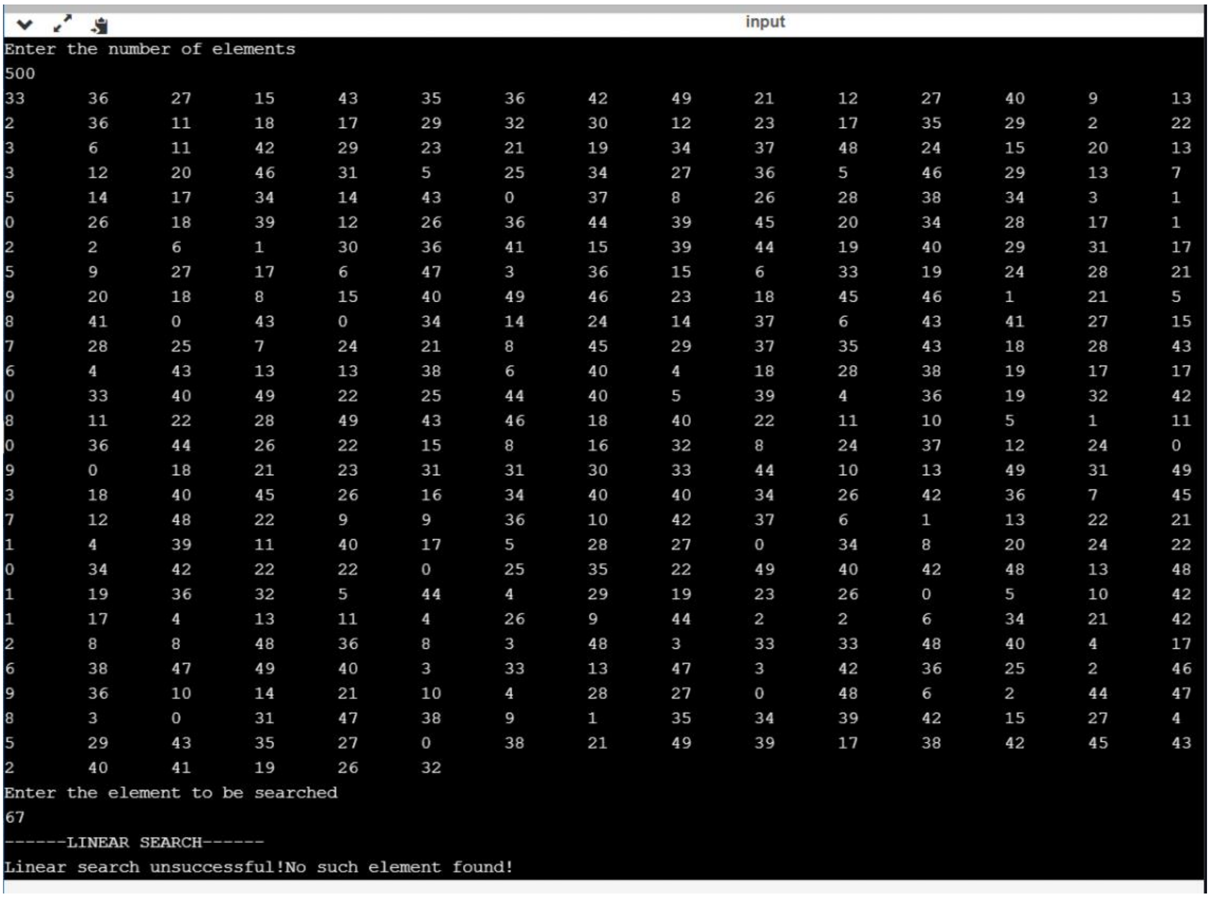
```
C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>gcc gcd.c

C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>a
Enter two positive integers:
4
20
G.C.D of 4 and 20 is 4.
C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>
```

2) Implement Recursive Binary search and Linear search and determine the time required to search an element. Repeat the experiment for different values of N and plot a graph of the time taken versus N.

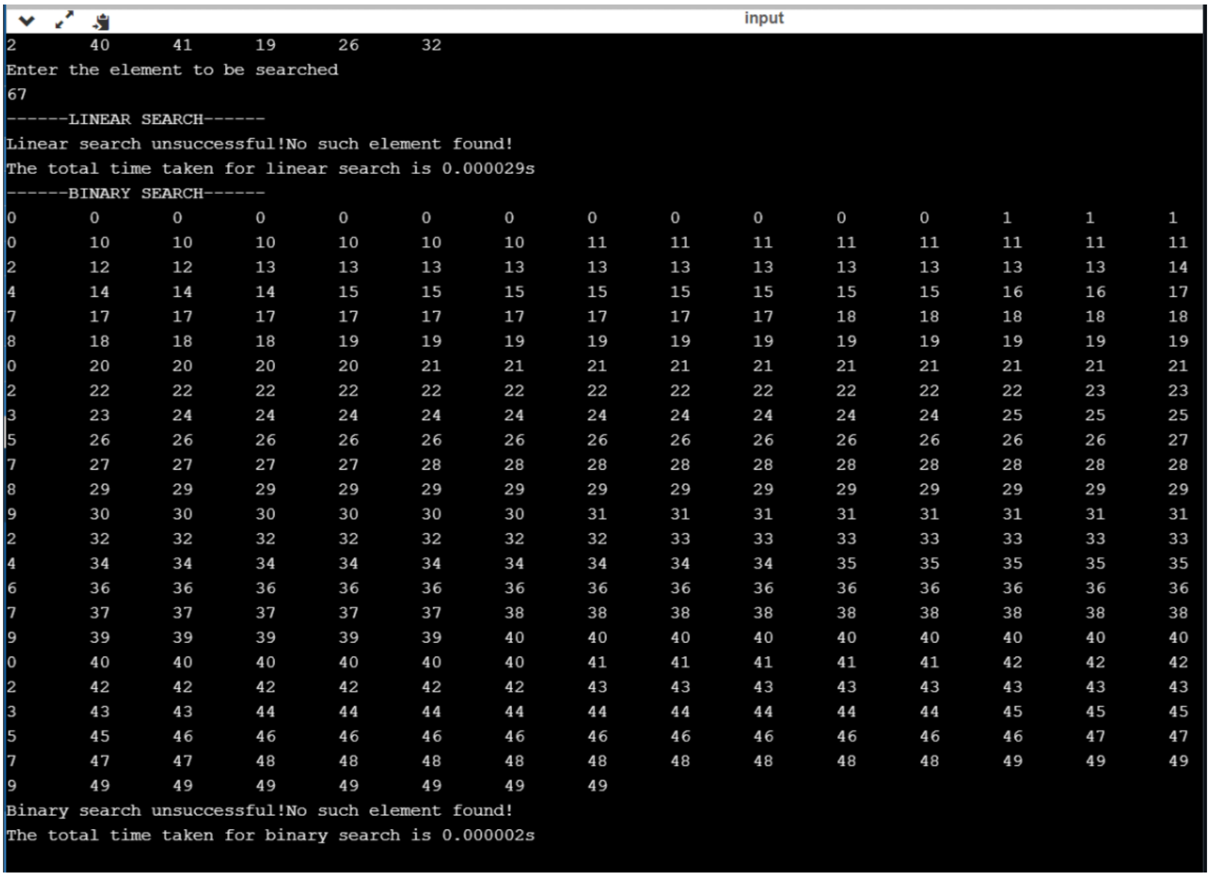
```
#include<stdio.h>

#include<time.h>

#include<stdlib.h>


int lin_rec(int ele,int n,int index,int arr[]){

        int pos=0;


        if(index==n)

                return -1;

        else if(arr[index]==ele){

                pos=index+1;

                return pos;

        }

        else{

                return lin_rec(ele,n,index+1,arr);

        }
        //return pos;
}

int bin_rec(int l, int r, int ele,int arr[])
{
   if (r >= l)
   {
        int mid = l + (r - l)/2;

        if (arr[mid] == ele)  return mid;

        if (arr[mid] > ele) return bin_rec( l, mid-1, ele,arr);

        return bin_rec( mid+1,r,ele,arr);
   }
   return -1;
}

int main(){
        int n,ele,pos;
        clock_t start_t,end_t;
        double total_t;

        printf("Enter the number of elements\n");
        scanf("%d",&n);
        int arr[n];

        for(int i=0;i<n;i++){
                arr[i]=rand()%50;
        }

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

```
    printf("%d\t",arr[i]);
}

printf("\nEnter the element to be searched\n");
scanf("%d",&ele);

start_t=clock();
pos=lin_rec(ele,n,0,arr);
end_t=clock();
total_t=(double)(end_t-start_t)/CLOCKS_PER_SEC;

printf("------LINEAR SEARCH------\n");
if(pos==-1)
    printf("Linear search unsuccessful!No such element found!\n");
else
    printf("Element is found at %d\n",pos);

printf("The total time taken for linear search is %fs\n",total_t);

//printf("Binary search\n");

printf("------BINARY SEARCH------\n");
for(int i=0;i<n-1;i++){
    for(int j=i+1;j<n;j++){
        if(arr[i]>arr[j]){
            int temp;
            temp=arr[i];
            arr[i]=arr[j];
            arr[j]=temp;
        }
    }
}
```

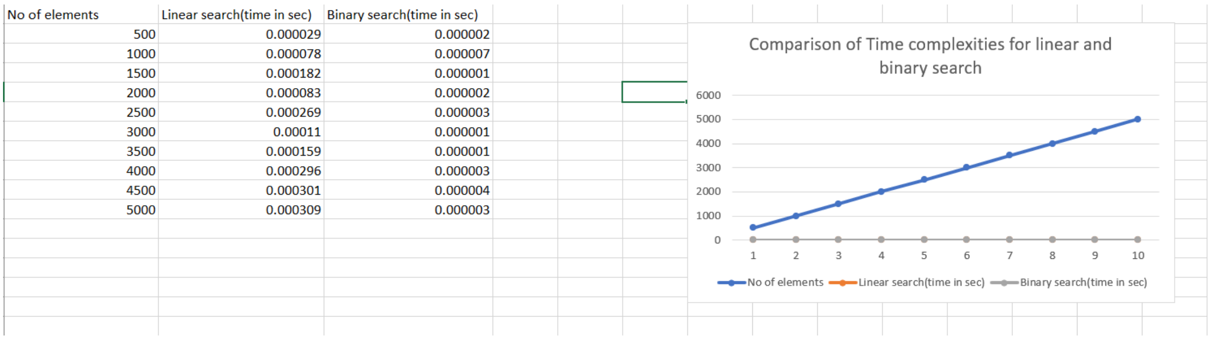
```
	start_t=clock();
	printf("%lu\n",start_t);
	pos=bin_rec(0,n-1,ele,arr);
	end_t=clock();
	printf("%lu\n",end_t);
	total_t=(end_t-start_t)/CLOCKS_PER_SEC;

	for(int i=0;i<n;i++){
		printf("%d\t",arr[i]);
	}

	if(pos==-1)
		printf("\nBinary search unsuccessful!No such element found!\n");
	else
		printf("\nElement is found at %d\n",pos+1);

	printf("The total time taken for binary search is %fs\n",total_t);
}
```



```
Enter the number of elements
500
33      36      27      15      43      35      36      42      49      21      12      27      40      9       13
2       36      11      18      17      29      32      30      12      23      17      35      29      2       22
3       6       11      42      29      23      21      19      34      37      48      24      15      20      13
3       12      20      46      31      5       25      34      27      36      5       46      29      13      7
5       14      17      34      14      43      0       37      8       26      28      38      34      3       1
0       26      18      39      12      26      36      44      39      45      20      34      28      17      1
2       2       6       1       30      36      41      15      39      44      19      40      29      31      17
5       9       27      17      6       47      3       36      15      6       33      19      24      28      21
9       20      18      8       15      40      49      46      23      18      45      46      1       21      5
8       41      0       43      0       34      14      24      14      37      6       43      41      27      15
7       28      25      7       24      21      8       45      29      37      35      43      18      28      43
6       4       43      13      13      38      6       40      4       18      28      38      19      17      17
0       33      40      49      22      25      44      40      5       39      4       36      19      32      42
8       11      22      28      49      43      46      18      40      22      11      10      5       1       11
0       36      44      26      22      15      8       16      32      8       24      37      12      24      0
9       0       18      21      23      31      31      30      33      44      10      13      49      31      49
3       18      40      45      26      16      34      40      40      34      26      42      36      7       45
7       12      48      22      9       9       36      10      42      37      6       1       13      22      21
1       4       39      11      40      17      5       28      27      0       34      8       20      24      22
0       34      42      22      22      0       25      35      22      49      40      42      48      13      48
1       19      36      32      5       44      4       29      19      23      26      0       5       10      42
1       17      4       13      11      4       26      9       44      2       2       6       34      21      42
2       8       8       48      36      8       3       48      3       33      33      48      40      4       17
6       38      47      49      40      3       33      13      47      3       42      36      25      2       46
9       36      10      14      21      10      4       28      27      0       48      6       2       44      47
8       3       0       31      47      38      9       1       35      34      39      42      15      27      4
5       29      43      35      27      0       38      21      49      39      17      38      42      45      43
2       40      41      19      26      32
Enter the element to be searched
67
------LINEAR SEARCH------
Linear search unsuccessful!No such element found!
```



```
2       40      41      19      26      32
Enter the element to be searched
67
------LINEAR SEARCH------
Linear search unsuccessful!No such element found!
The total time taken for linear search is 0.000029s
------BINARY SEARCH------
0       0       0       0       0       0       0       0       0       0       0       1       1       1
0       10      10      10      10      10      10      11      11      11      11      11      11      11      11
2       12      12      13      13      13      13      13      13      13      13      13      13      13      14
4       14      14      14      15      15      15      15      15      15      15      15      16      16      17
7       17      17      17      17      17      17      17      17      17      18      18      18      18      18
8       18      18      18      19      19      19      19      19      19      19      19      19      19      19
0       20      20      20      20      21      21      21      21      21      21      21      21      21      21
2       22      22      22      22      22      22      22      22      22      22      22      22      23      23
3       23      24      24      24      24      24      24      24      24      24      24      25      25      25
5       26      26      26      26      26      26      26      26      26      26      26      26      26      27
7       27      27      27      27      28      28      28      28      28      28      28      28      28      28
8       29      29      29      29      29      29      29      29      29      29      29      29      29      29
9       30      30      30      30      30      30      31      31      31      31      31      31      31      31
2       32      32      32      32      32      32      32      33      33      33      33      33      33      33
4       34      34      34      34      34      34      34      34      34      35      35      35      35      35
6       36      36      36      36      36      36      36      36      36      36      36      36      36      36
7       37      37      37      37      37      38      38      38      38      38      38      38      38      38
9       39      39      39      39      39      40      40      40      40      40      40      40      40      40
0       40      40      40      40      40      40      41      41      41      41      41      42      42      42
2       42      42      42      42      42      42      43      43      43      43      43      43      43      43
3       43      43      44      44      44      44      44      44      44      44      44      45      45      45
5       45      46      46      46      46      46      46      46      46      46      46      46      47      47
7       47      47      48      48      48      48      48      48      48      48      48      49      49      49
9       49      49      49      49      49      49      49
Binary search unsuccessful!No such element found!
The total time taken for binary search is 0.000002s
```

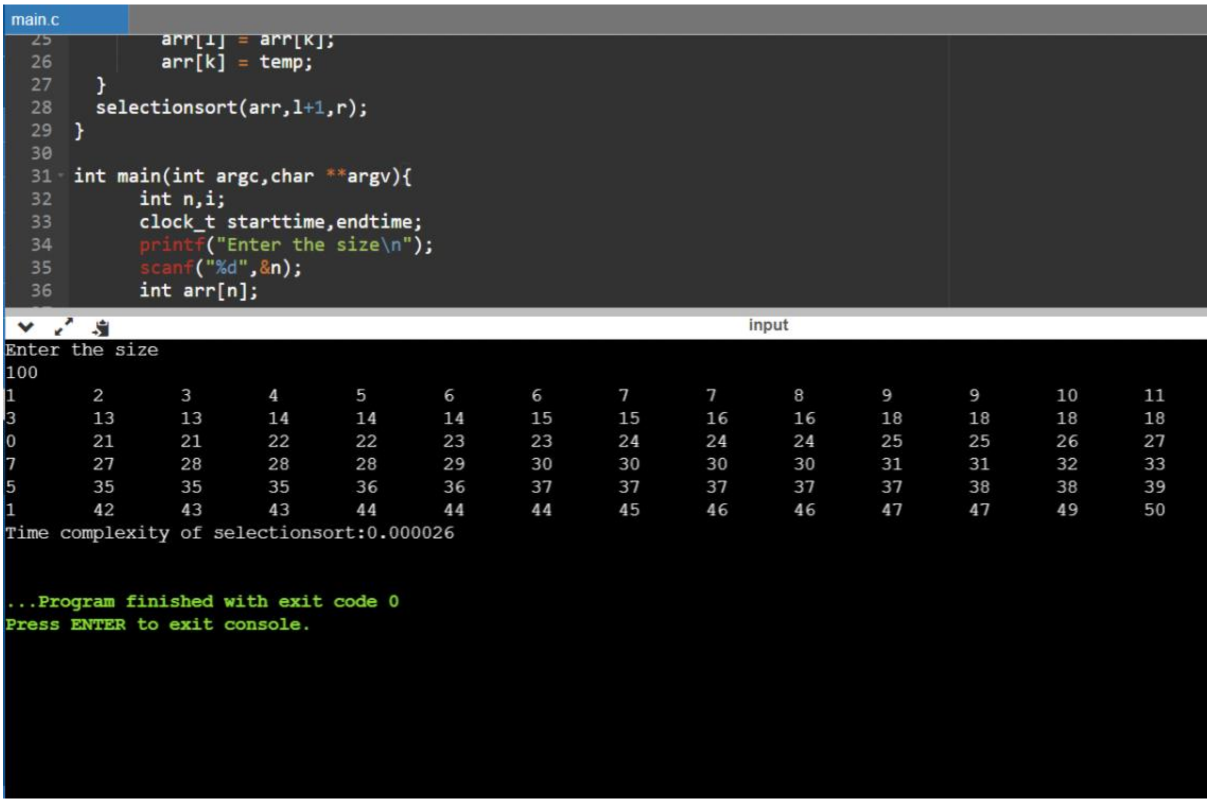


No of elements	Linear search(time in sec)	Binary search(time in sec)
500	0.000029	0.000002
1000	0.000078	0.000007
1500	0.000182	0.000001
2000	0.000083	0.000002
2500	0.000269	0.000003
3000	0.00011	0.000001
3500	0.000159	0.000001
4000	0.000296	0.000003
4500	0.000301	0.000004
5000	0.000309	0.000003

Comparison of Time complexities for linear and binary search
6000
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
7
8
9
10
No of elements
Linear search(time in sec)
Binary search(time in sec)

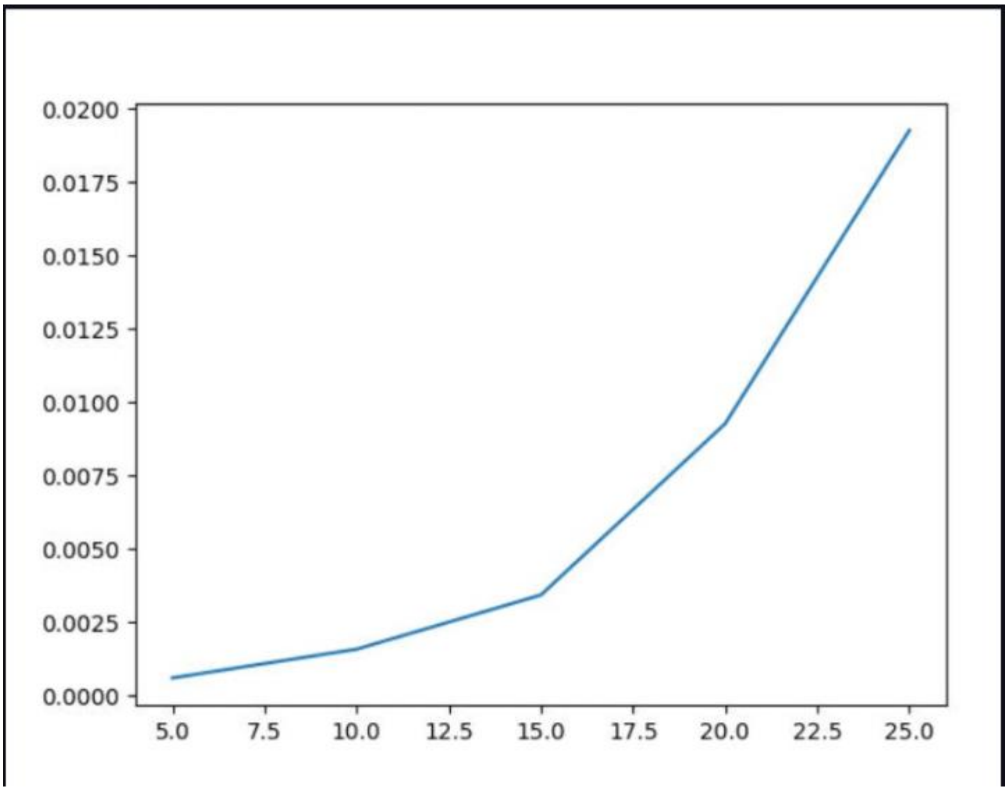
3)Sort a given set of N integer elements using Selection Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.

```
#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <time.h>


int min(int arr[],int l,int r){

  int minval = pow(2,31)-1;

  int minindex = -1;

  for(int i=l; i<=r; i++){

    if(arr[i] < minval){

      minval = arr[i];

      minindex = i;

    }

  }

  return minindex;

}


void selectionsort(int arr[],int l,int r){

        int temp,k;

        if(l == r)

            return;

        k = min(arr,l,r);

        if(l != k){
              temp = arr[l];
              arr[l] = arr[k];
              arr[k] = temp;
 }
 selectionsort(arr,l+1,r);
}

int main(int argc,char **argv){
        int n,i;
        clock_t starttime,endtime;
        printf("Enter the size\n");
        scanf("%d",&n);
        int arr[n];

        for(i=0; i<n; i++)
              arr[i] = (rand()%50)+1;

        starttime = clock();
        selectionsort(arr,0,n-1);
        endtime = clock();

        double time = ((double)(endtime-starttime)/CLOCKS_PER_SEC);

        for(i=0; i<n; i++)
              printf("%d\t",arr[i]);
        printf("\nTime complexity of selectionsort:%f\n",time);
        return 0;
}
```



```c
        arr[l] = arr[k];
        arr[k] = temp;
    }
    selectionsort(arr,l+1,r);
}

int main(int argc,char **argv){
      int n,i;
      clock_t starttime,endtime;
      printf("Enter the size\n");
      scanf("%d",&n);
      int arr[n];
```

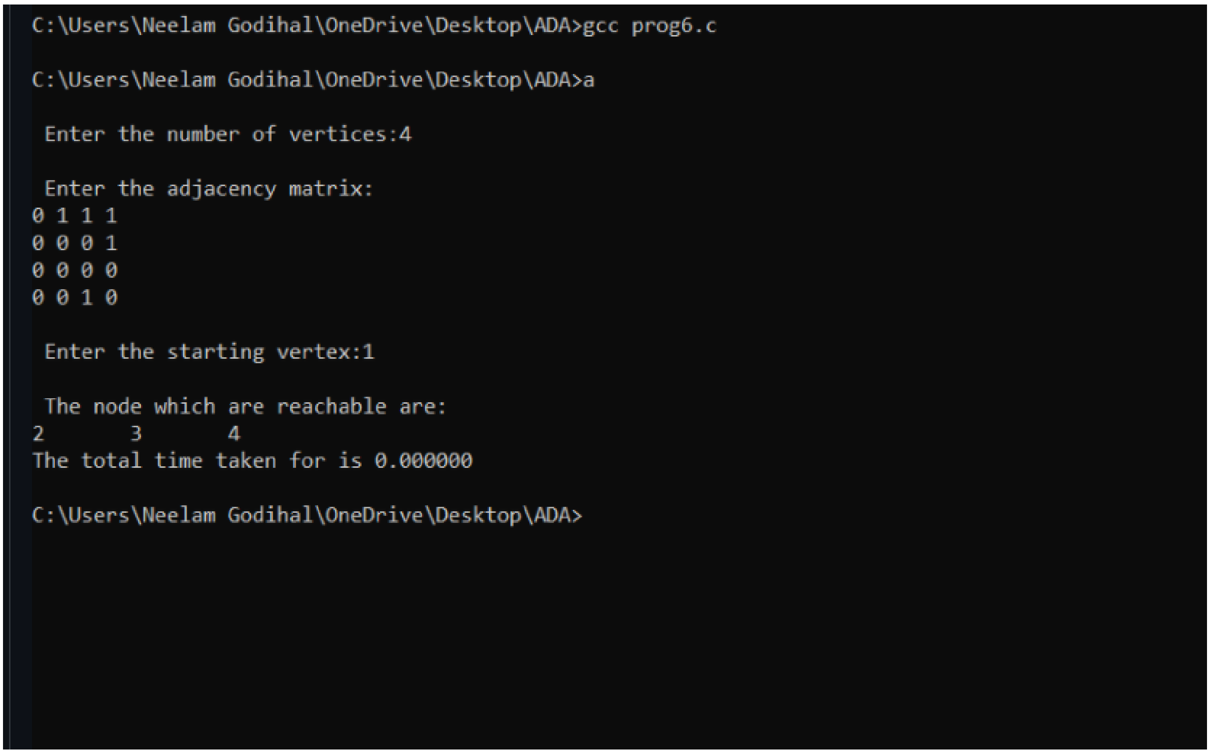
```
Enter the size
100
1       2       3       4       5       6       6       7       7       8       9       9       10      11
3       13      13      14      14      14      15      15      16      16      18      18      18      18
0       21      21      22      22      23      23      24      24      24      25      25      26      27
7       27      28      28      28      29      30      30      30      30      31      31      32      33
5       35      35      35      36      36      37      37      37      37      37      38      38      39
1       42      43      43      44      44      44      45      46      46      47      47      49      50
Time complexity of selectionsort:0.000026


...Program finished with exit code 0
Press ENTER to exit console.
```



4)Write program to do the following: a. Print all the nodes reachable from a given starting node in a digraph using BFS method. b. Check whether a given graph is connected or not using DFS method.

```
#include<stdio.h>

#include<time.h>

#include<stdlib.h>


int a[20][20],q[20],visited[20],n,i,j,f=0,r=-1;

void bfs(int v)

{

        for(i=1;i<=n;i++)

        if(a[v][i] && !visited[i])

        q[++r]=i;

        if(f<=r)

        {

                visited[q[f]]=1;

                bfs(q[f++]);

        }

}

void main()

{

        int v;

        clock_t start_t,end_t;

        double total_t;


        printf("\n Enter the number of vertices:");

        scanf("%d",&n);

        for(i=1;i<=n;i++)

        {

                q[i]=0;

                visited[i]=0;

        }

        printf("\n Enter the adjacency matrix:\n");
        for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        scanf("%d",&a[i][j]);

        printf("\n Enter the starting vertex:");
        scanf("%d",&v);
        start_t=clock();
        bfs(v);
        end_t=clock();
        total_t=(double)(end_t-start_t)/CLOCKS_PER_SEC;

        printf("\n The node which are reachable are:\n");
        for(i=1;i<=n;i++)
        if(visited[i])
        printf("%d\t",i);

        printf("\nThe total time taken for is %f\n",total_t);

}
```

}

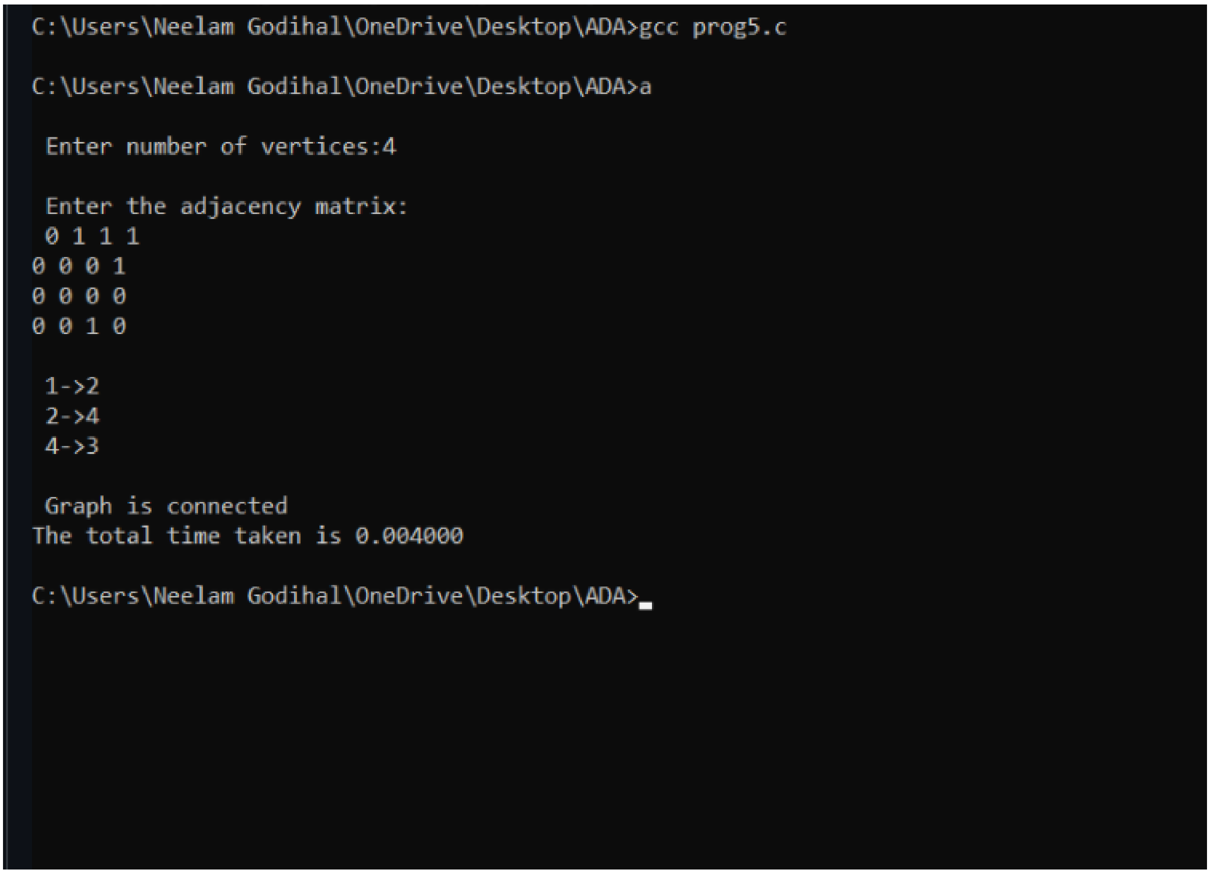


```
C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>gcc prog6.c

C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>a

 Enter the number of vertices:4

 Enter the adjacency matrix:
0 1 1 1
0 0 0 1
0 0 0 0
0 0 1 0

 Enter the starting vertex:1

 The node which are reachable are:
2       3       4
The total time taken for is 0.000000

C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>
```

```
#include<stdio.h>

#include<time.h>

#include<stdlib.h>


int a[20][20],visited[20],n;

void dfs(int v)

{

        int i;

        visited[v]=1;

        for(i=1;i<=n;i++)

        if(a[v][i] && !visited[i])

        {

                printf("\n %d->%d",v,i);

                dfs(i);

        }

}


void main()

{
```

```
int i,j,count=0;
clock_t start_t,end_t;
double total_t;
printf("\n Enter number of vertices:");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
        visited[i]=0;
        for(j=1;j<=n;j++)
        a[i][j]=0;
}
printf("\n Enter the adjacency matrix:\n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
scanf("%d",&a[i][j]);

start_t=clock();
dfs(1);
printf("\n");
for(i=1;i<=n;i++)
{
        if(visited[i])
        count++;
}
end_t=clock();
total_t=(double)(end_t-start_t)/CLOCKS_PER_SEC;

if(count==n)
printf("\n Graph is connected");
else
printf("\n Graph is not connected");
```

```
        printf("\nThe total time taken is %f\n",total_t);
}
```

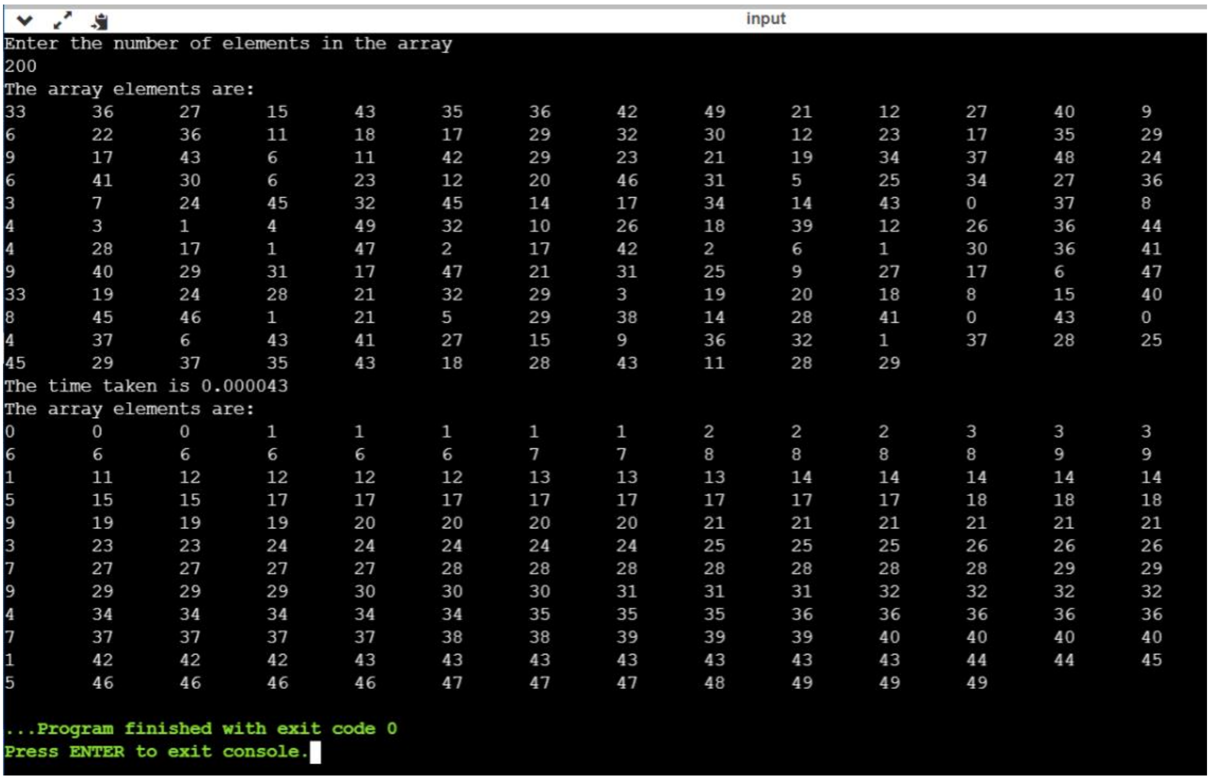


```
C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>gcc prog5.c

C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>a

 Enter number of vertices:4

 Enter the adjacency matrix:
 0 1 1 1
0 0 0 1
0 0 0 0
0 0 1 0

 1->2
 2->4
 4->3

 Graph is connected
The total time taken is 0.004000

C:\Users\Neelam Godihal\OneDrive\Desktop\ADA>
```

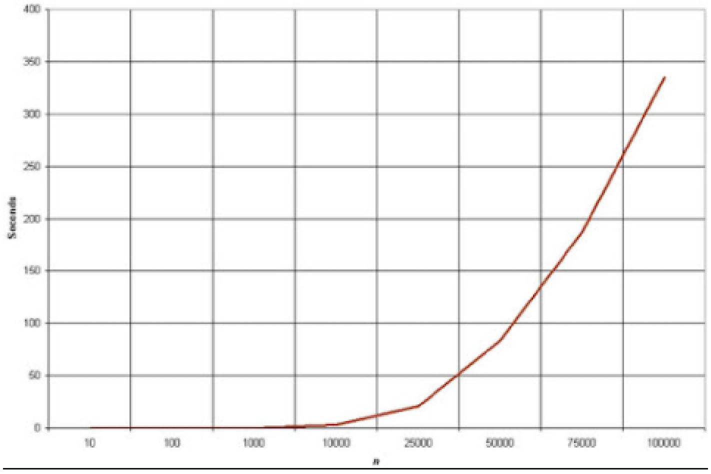
5)Sort a given set of N integer elements using Insertion Sort technique and compute its time taken.

```
#include<stdio.h>

#include<time.h>

#include<stdlib.h>


void insertSort(int arr[], int n)

{

    int i, key, j;

    for (i = 1; i < n; i++) {

        key = arr[i];

        j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}

void main(){
	int n;
	clock_t start_t,end_t;
	double total_t;

	printf("Enter the number of elements in the array\n");
	scanf("%d",&n);
	int arr[n];

	for(int i=0;i<n;i++){
		arr[i]=rand()%50;
	}

	printf("The array elements are:\n");
	for(int i=0;i<n;i++){
		printf("%d\t",arr[i]);
	}

	start_t=clock();
	insertSort(arr, n);
	end_t=clock();
	total_t=(double)(end_t-start_t)/CLOCKS_PER_SEC;

	printf("\nThe time taken is %f\n",total_t);
```

```
        printf("The array elements are:\n");
        for(int i=0;i<n;i++){
                printf("%d\t",arr[i]);
        }
}
```



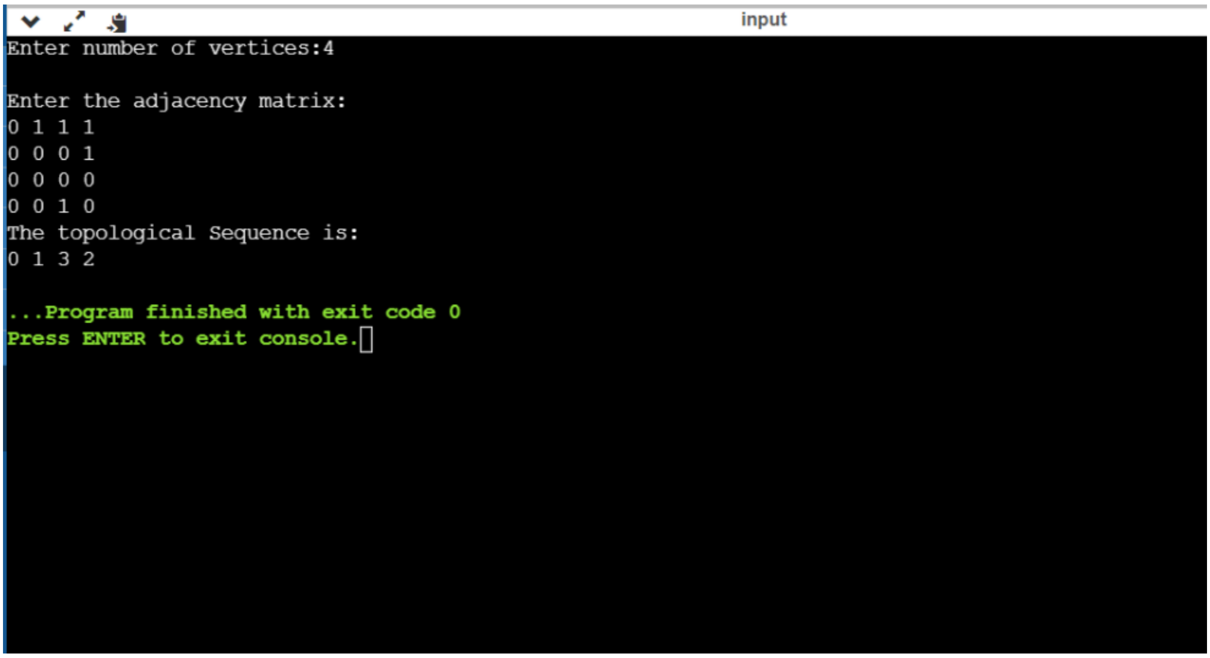
```
Enter the number of elements in the array
200
The array elements are:
33      36      27      15      43      35      36      42      49      21      12      27      40      9
6       22      36      11      18      17      29      32      30      12      23      17      35      29
9       17      43      6       11      42      29      23      21      19      34      37      48      24
6       41      30      6       23      12      20      46      31      5       25      34      27      36
3       7       24      45      32      45      14      17      34      14      43      0       37      8
4       3       1       4       49      32      10      26      18      39      12      26      36      44
4       28      17      1       47      2       17      42      2       6       1       30      36      41
9       40      29      31      17      47      21      31      25      9       27      17      6       47
33      19      24      28      21      32      29      3       19      20      18      8       15      40
8       45      46      1       21      5       29      38      14      28      41      0       43      0
4       37      6       43      41      27      15      9       36      32      1       37      28      25
45      29      37      35      43      18      28      43      11      28      29
The time taken is 0.000043
The array elements are:
0       0       0       1       1       1       1       1       2       2       2       3       3       3
6       6       6       6       6       6       7       7       8       8       8       8       9       9
1       11      12      12      12      12      13      13      13      14      14      14      14      14
5       15      15      17      17      17      17      17      17      17      17      18      18      18
9       19      19      19      20      20      20      20      21      21      21      21      21      21
3       23      23      24      24      24      24      24      25      25      25      26      26      26
7       27      27      27      27      28      28      28      28      28      28      28      29      29
9       29      29      29      30      30      30      31      31      31      32      32      32      32
4       34      34      34      34      34      35      35      35      36      36      36      36      36
7       37      37      37      37      38      38      39      39      39      40      40      40      40
1       42      42      42      43      43      43      43      43      43      43      44      44      45
5       46      46      46      46      47      47      47      48      49      49      49

...Program finished with exit code 0
Press ENTER to exit console.
```



6) Write program to obtain the Topological ordering of vertices in a given digraph.

```
#include<stdio.h>
#include<conio.h>
```

```
int a[10][10],n,indegre[10];

void find_indegre()
{
        int j,i,sum;
                for(j=0;j<n;j++)
                {
                        sum=0;
                        for(i=0;i<n;i++)
                        sum+=a[i][j];
                        indegre[j]=sum;
                }
}

void topology()
{
        int i,u,v,t[10],s[10],top=-1,k=0;
        find_indegre();
        for(i=0;i<n;i++)
        {
                if(indegre[i]==0)
                s[++top]=i;
        }

        while(top!=-1)
        {
                u=s[top--];
                t[k++]=u;
                for(v=0;v<n;v++)
                {
                        if(a[u][v]==1)
                        {
                                indegre[v]--;
                                if(indegre[v]==0)
                                        s[++top]=v;
                        }
                }
        }
        printf("The topological Sequence is:\n");
        for(i=0;i<n;i++)
        printf("%d ",t[i]);
}

void main()
{
        int i,j;
        printf("Enter number of vertices:");
        scanf("%d",&n);

        printf("\nEnter the adjacency matrix:\n");

        for(i=0;i<n;i++)
        {
                for(j=0;j<n;j++)
                scanf("%d",&a[i][j]);
        }
        topology();
}
```

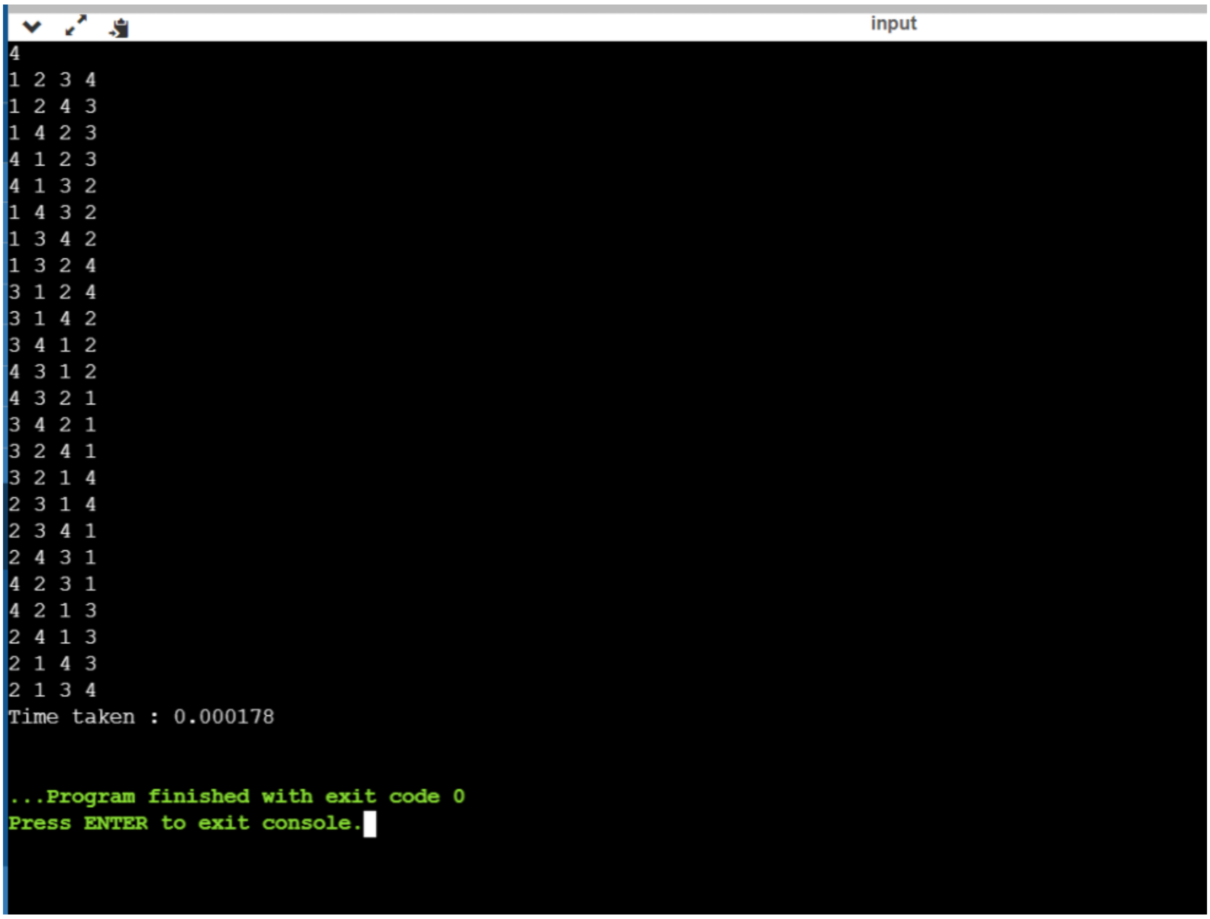


```
Enter number of vertices:4

Enter the adjacency matrix:
0 1 1 1
0 0 0 1
0 0 0 0
0 0 1 0
The topological Sequence is:
0 1 3 2

...Program finished with exit code 0
Press ENTER to exit console.
```

7) Implement Johnson Trotter algorithm to generate permutations.

```
#include <stdio.h>

#include <time.h>


void display(int arr[],int n){

  int i;

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

    printf("%d ",arr[i]);

  printf("\n");

}


void swap(int *p, int *q){

  int temp = *p;

  *p = *q;

  *q = temp;

}


void swapdir(char *p, char *q){

  char temp = *p;
  *p = *q;
  *q = temp;
}

int largestmobilenumber(int n,int arr[],char dir[],int max){
  int i;
  if(arr[0] == max && dir[0] == 'l'){
    dir[0] = 'r';
    return largestmobilenumber(n,arr,dir,max-1);
  }
  else if(arr[n-1] == max && dir[n-1] == 'r'){
    dir[n-1] = 'l';
    return largestmobilenumber(n,arr,dir,max-1);
  }
  for(i=0; i<n; i++){
    if(arr[i] == max){
      if((dir[i]=='l' && arr[i]>arr[i-1]) || (dir[i]=='r' && arr[i]>arr[i+1]))
        return i;
      else{
        if(dir[i] == 'l')
          dir[i] = 'r';
        else
          dir[i] = 'l';
        return largestmobilenumber(n,arr,dir,max-1);
      }
    }
  }
  return -1;
}

void permutations(int n){
  int i,arr[n];
  char dir[n];
  for(i=0; i<n; i++){
    arr[i] = i+1;
    dir[i] = 'l';
  }
  while(i != -1){
    display(arr,n);
    i = largestmobilenumber(n,arr,dir,n);
    if(i != -1){
      if(dir[i] == 'r'){
        swap(&arr[i],&arr[i+1]);
        swapdir(&dir[i],&dir[i+1]);
      }
      else{
        swap(&arr[i],&arr[i-1]);
        swapdir(&dir[i],&dir[i-1]);
      }
    }
  }
}

int main(void){
  int n;
  clock_t starttime,endtime;
  double efftime;
  scanf("%d",&n);
  starttime = clock();
  permutations(n);
  endtime = clock();
  efftime = ((double)(endtime - starttime))/CLOCKS_PER_SEC;
  printf("Time taken : %lf\n",efftime);
  return 0;
```

}



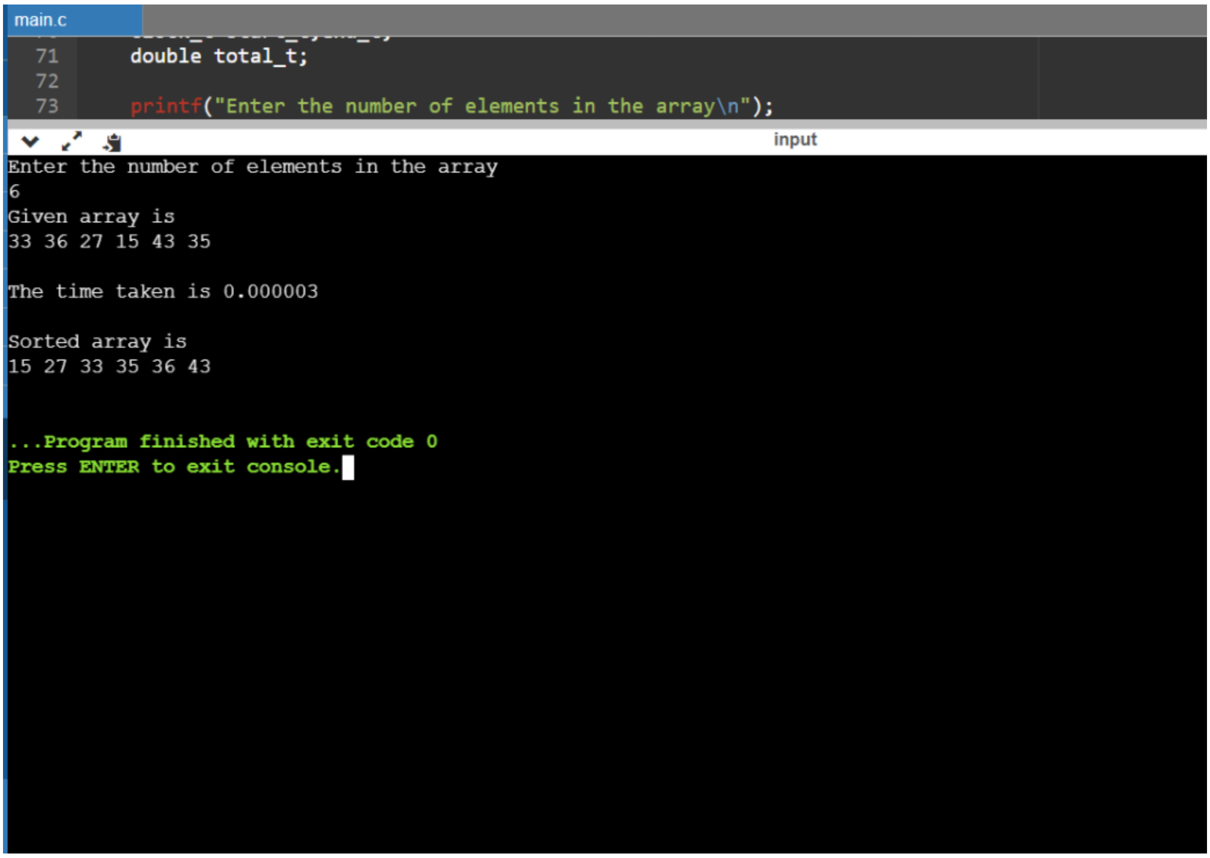
```
4
1 2 3 4
1 2 4 3
1 4 2 3
4 1 2 3
4 1 3 2
1 4 3 2
1 3 4 2
1 3 2 4
3 1 2 4
3 1 4 2
3 4 1 2
4 3 1 2
4 3 2 1
3 4 2 1
3 2 4 1
3 2 1 4
2 3 1 4
2 3 4 1
2 4 3 1
4 2 3 1
4 2 1 3
2 4 1 3
2 1 4 3
2 1 3 4
Time taken : 0.000178


...Program finished with exit code 0
Press ENTER to exit console.
```

8) Sort a given set of N integer elements using Merge Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.

```
#include <stdio.h>

#include <stdlib.h>

#include<time.h>


void merge(int arr[], int l, int m, int r)

{

    int i, j, k;

    int n1 = m - l + 1;

    int n2 = r - m;


    int L[n1], R[n2];
```

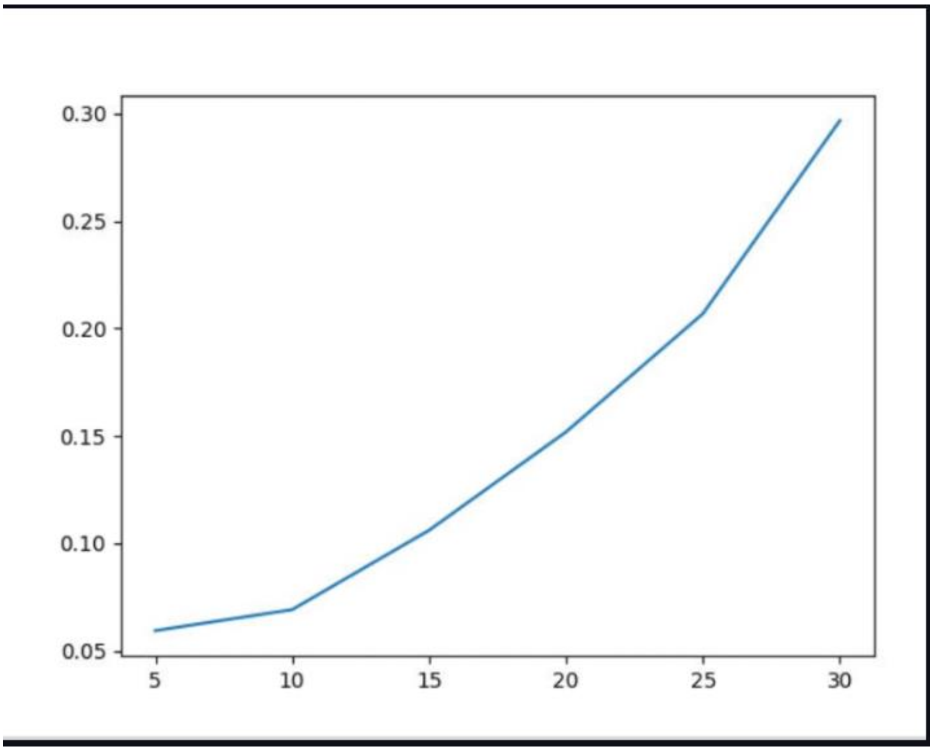
```
for (i = 0; i < n1; i++)
    L[i] = arr[l + i];
for (j = 0; j < n2; j++)
    R[j] = arr[m + 1 + j];

i = 0;
j = 0;
k = l;
while (i < n1 && j < n2) {
    if (L[i] <= R[j]) {
        arr[k] = L[i];
        i++;
    }
    else {
        arr[k] = R[j];
        j++;
    }
    k++;
}

while (i < n1) {
    arr[k] = L[i];
    i++;
    k++;
}

while (j < n2) {
    arr[k] = R[j];
    j++;
    k++;
}
```

```
}

void mergeSort(int arr[], int l, int r)
{
    if (l < r) {

        int m = l + (r - l) / 2;

        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

        merge(arr, l, m, r);
    }
}

void printArray(int A[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", A[i]);
    printf("\n");
}

int main()
{
            int n;
            clock_t start_t,end_t;
            double total_t;

            printf("Enter the number of elements in the array\n");
            scanf("%d",&n);
            int arr[n];
        for(int i=0;i<n;i++){
                arr[i]=rand()%50;
        }

    printf("Given array is \n");
    printArray(arr, n);

        start_t=clock();
    mergeSort(arr, 0, n - 1);
        end_t=clock();
        total_t=(double)(end_t-start_t)/CLOCKS_PER_SEC;

        printf("\nThe time taken is %f\n",total_t);

    printf("\nSorted array is \n");
    printArray(arr, n);
    return 0;
}
```



```
    double total_t;

    printf("Enter the number of elements in the array\n");
```

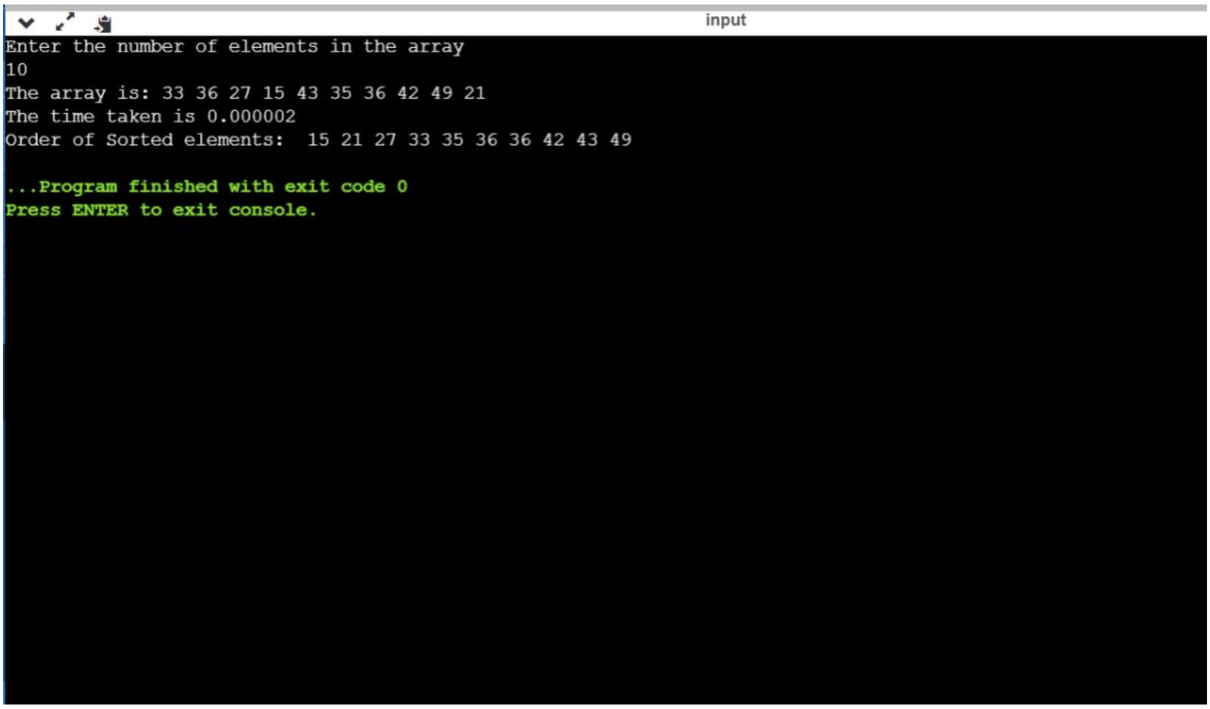
```
Enter the number of elements in the array
6
Given array is
33 36 27 15 43 35

The time taken is 0.000003

Sorted array is
15 27 33 35 36 43


...Program finished with exit code 0
Press ENTER to exit console.
```



9) Sort a given set of N integer elements using Quick Sort technique and compute its time taken.

```
#include<stdio.h>

#include <stdlib.h>

#include<time.h>


void quicksort(int arr[25],int first,int last){

  int i, j, pivot, temp;


  if(first<last){

    pivot=first;

    i=first;

    j=last;


    while(i<j){

      while(arr[i]<=arr[pivot]&&i<last)

        i++;

      while(arr[j]>arr[pivot])

        j--;

      if(i<j){

        temp=arr[i];

        arr[i]=arr[j];

        arr[j]=temp;

      }

    }


    temp=arr[pivot];

    arr[pivot]=arr[j];

    arr[j]=temp;

    quicksort(arr,first,j-1);

    quicksort(arr,j+1,last);
  }
}

int main(){
   int n;
           clock_t start_t,end_t;
           double total_t;

           printf("Enter the number of elements in the array\n");
           scanf("%d",&n);
           int arr[n];

           for(int i=0;i<n;i++){
                      arr[i]=rand()%50;
           }

           printf("The array is:");
           for(int i=0;i<n;i++)
     printf(" %d",arr[i]);

           start_t=clock();
   quicksort(arr,0,n-1);
   end_t=clock();
           total_t=(double)(end_t-start_t)/CLOCKS_PER_SEC;

           printf("\nThe time taken is %f\n",total_t);

   printf("Order of Sorted elements: ");
   for(int i=0;i<n;i++)
     printf(" %d",arr[i]);

   return 0;
```

}



```
Enter the number of elements in the array
10
The array is: 33 36 27 15 43 35 36 42 49 21
The time taken is 0.000002
Order of Sorted elements:  15 21 27 33 35 36 36 42 43 49

...Program finished with exit code 0
Press ENTER to exit console.
```