Lab 1: Divide and Conquer

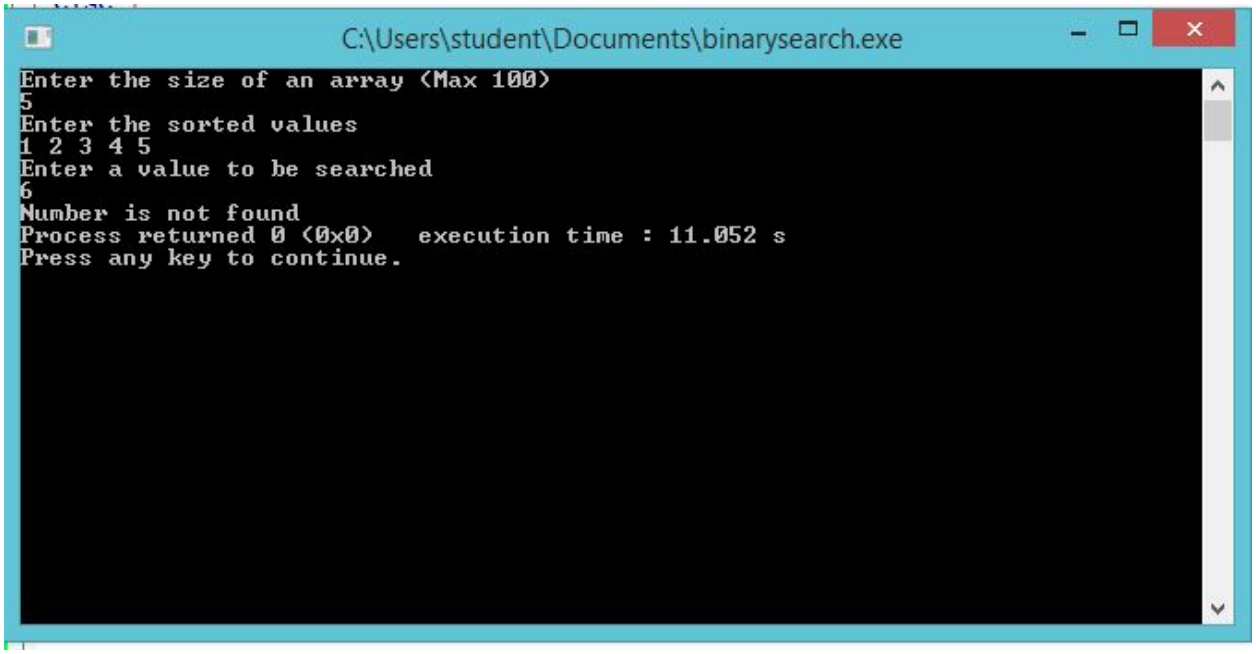
Date : 14.2.18

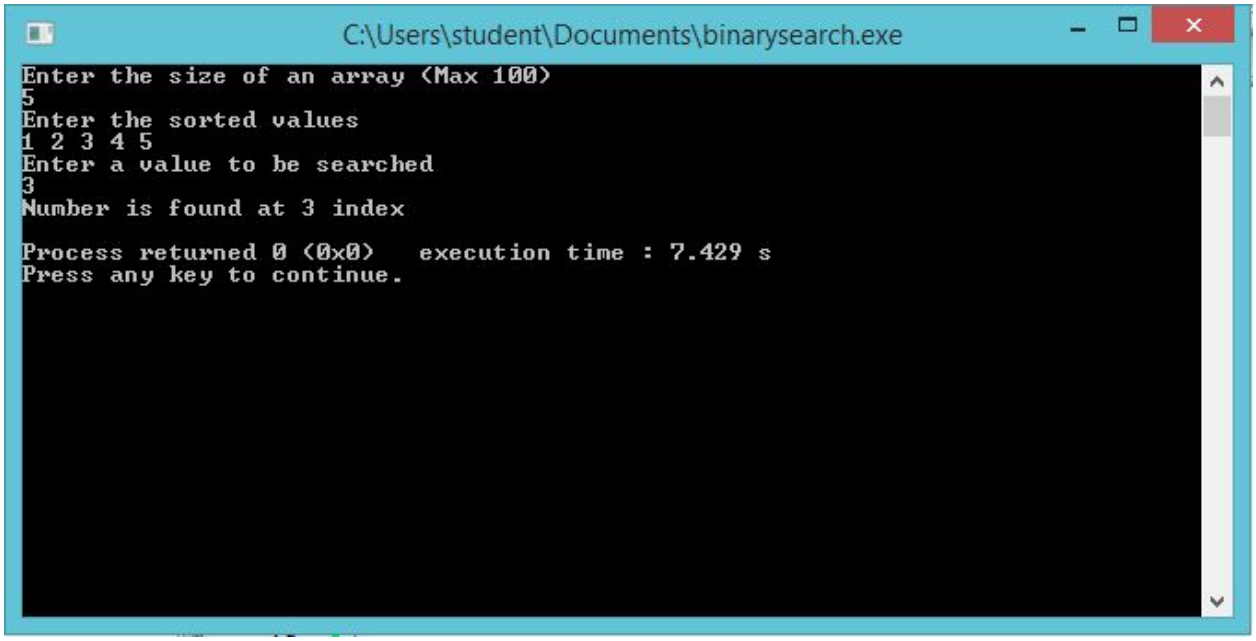
Submitted By : Abhineet Singh

1. ***Write a program that search an element in the sorted list using the binary search method of divide and conquer approach. The elements can be read from a file or can be generated using the random number generator.***

```
#include <iostream>
using namespace std;

int BinarySearch(int arr[], int num, int beg, int end)
{
 int mid;

 if (beg > end){
  cout << "Number is not found";
  return 0;
 }

 else {
  mid = (beg + end) / 2;

  if(arr[mid] == num){
   cout << "Number is found at " << mid + 1 << " index \n";
   return 0;
  }

 else if (num > arr[mid]) {
   BinarySearch (arr, num, mid+1, end);
  }

   else if (num < arr[mid]) {
   BinarySearch (arr, num, beg , mid-1);
  }
 }
}
```

```
int main() {
int arr[100], num, i, n, beg, end;

cout <<"Enter the size of an array (Max 100) \n";
cin >> n;

cout <<"Enter the sorted values \n";
for(i=0; i<n; i++) {
 cin >> arr[i];
}

cout <<"Enter a value to be searched \n";
cin >> num;

beg = 0;
end = n-1;
BinarySearch (arr, num, beg, end);
return 0;
}
```

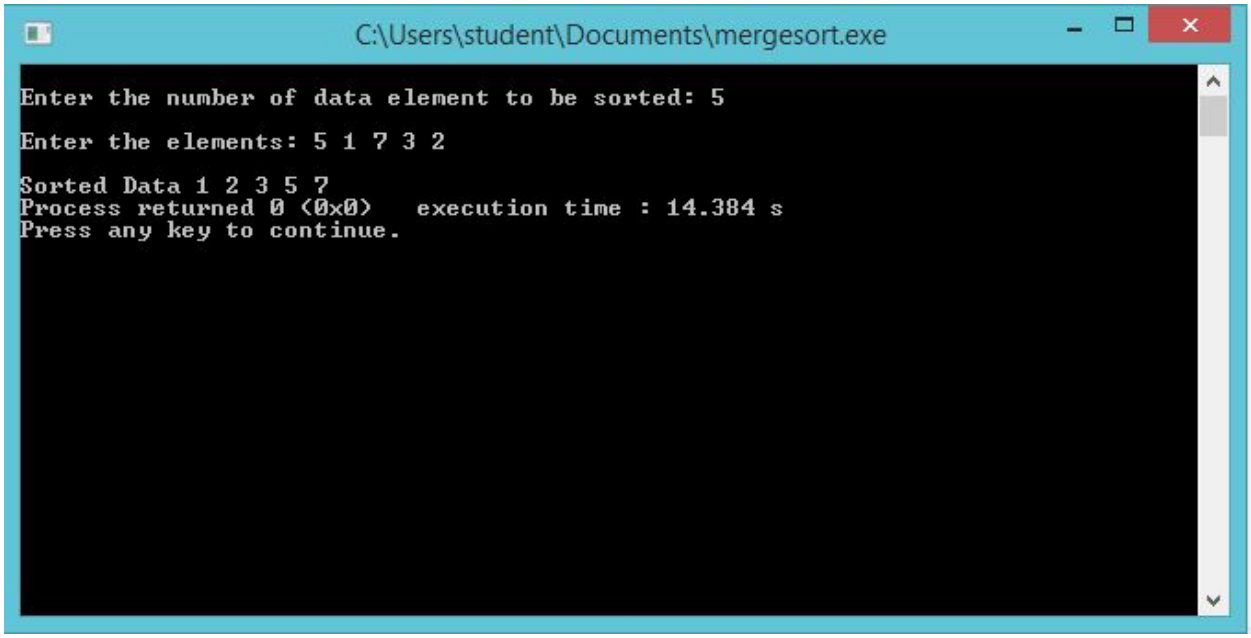

C:\Users\student\Documents\binarysearch.exe
Enter the size of an array (Max 100)
5
Enter the sorted values
1 2 3 4 5
Enter a value to be searched
6
Number is not found
Process returned 0 (0x0)   execution time : 11.052 s
Press any key to continue.

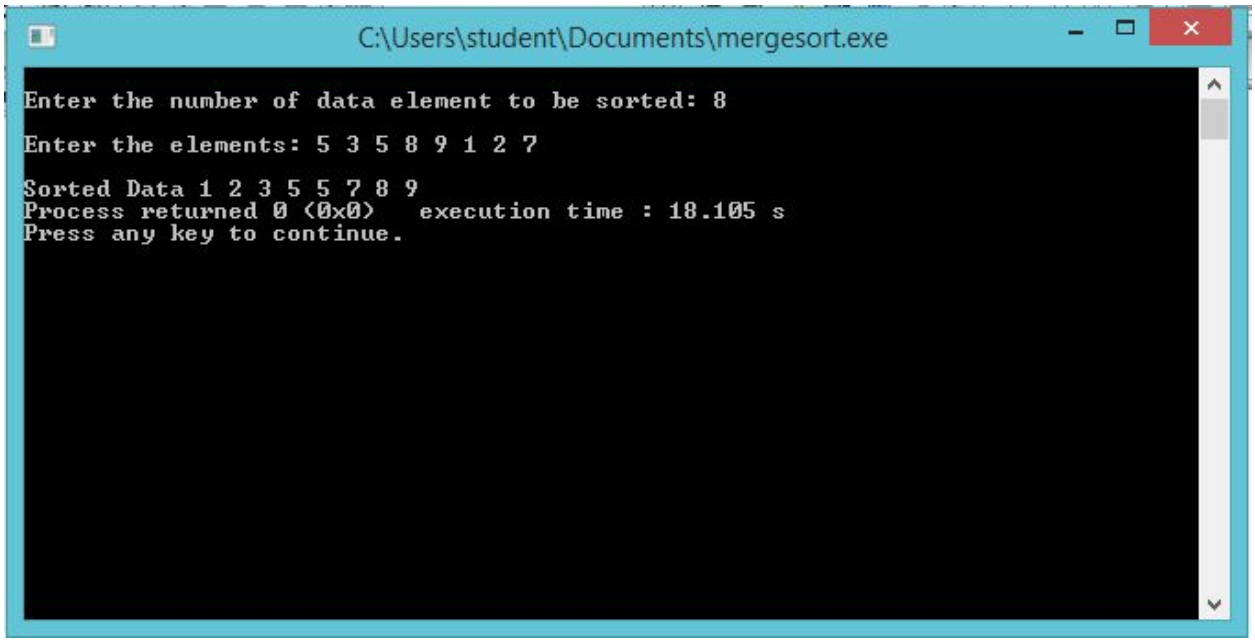

C:\Users\student\Documents\binarysearch.exe
Enter the size of an array (Max 100)
5
Enter the sorted values
1 2 3 4 5
Enter a value to be searched
3
Number is found at 3 index
Process returned 0 (0x0)   execution time : 7.429 s
Press any key to continue.

2. ***Sort a given set of elements using the Merge sort method of divide and conquer approach. The elements can be read from a file or can be generated using the random number generator.***

```
#include <iostream>
using namespace std;

void Merge(int *a, int low, int high, int mid)
{
	int i, j, k, temp[high-low+1];
	i = low;
	k = 0;
	j = mid + 1;

	while (i <= mid && j <= high)
	{
		if (a[i] < a[j])
		{
			temp[k] = a[i];
			k++;
			i++;
		}
		else
		{
			temp[k] = a[j];
			k++;
			j++;
		}
	}
	while (i <= mid)
	{
		temp[k] = a[i];
		k++;
		i++;
	}
	while (j <= high)
	{
		temp[k] = a[j];
		k++;
		j++;
	}
	for (i = low; i <= high; i++)
{
```

```
                a[i] = temp[i-low];
        }
}

void MergeSort(int *a, int low, int high)
{
        int mid;
        if (low < high)
        {
                mid=(low+high)/2;
                MergeSort(a, low, mid);
                MergeSort(a, mid+1, high);
                Merge(a, low, high, mid);
        }
}

int main()
{
        int n, i;
        cout<<"\nEnter the number of data element to be sorted: ";
        cin>>n;
   cout<<"\nEnter the elements: ";
        int arr[n];
        for(i = 0; i < n; i++)
        {
                cin>>arr[i];
        }
        MergeSort(arr, 0, n-1);
        cout<<"\nSorted Data ";
        for (i = 0; i < n; i++)
      cout<<arr[i]<<" ";
        return 0;
}
```

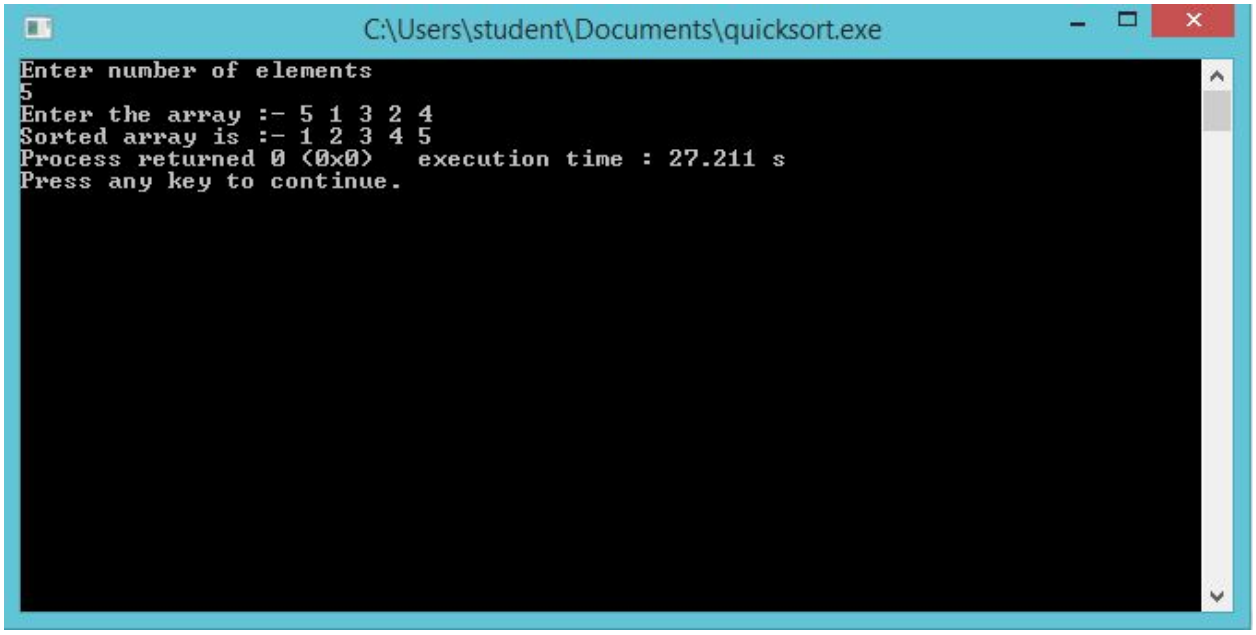

C:\Users\student\Documents\mergesort.exe
Enter the number of data element to be sorted: 5
Enter the elements: 5 1 7 3 2
Sorted Data 1 2 3 5 7
Process returned 0 (0x0)   execution time : 14.384 s
Press any key to continue.

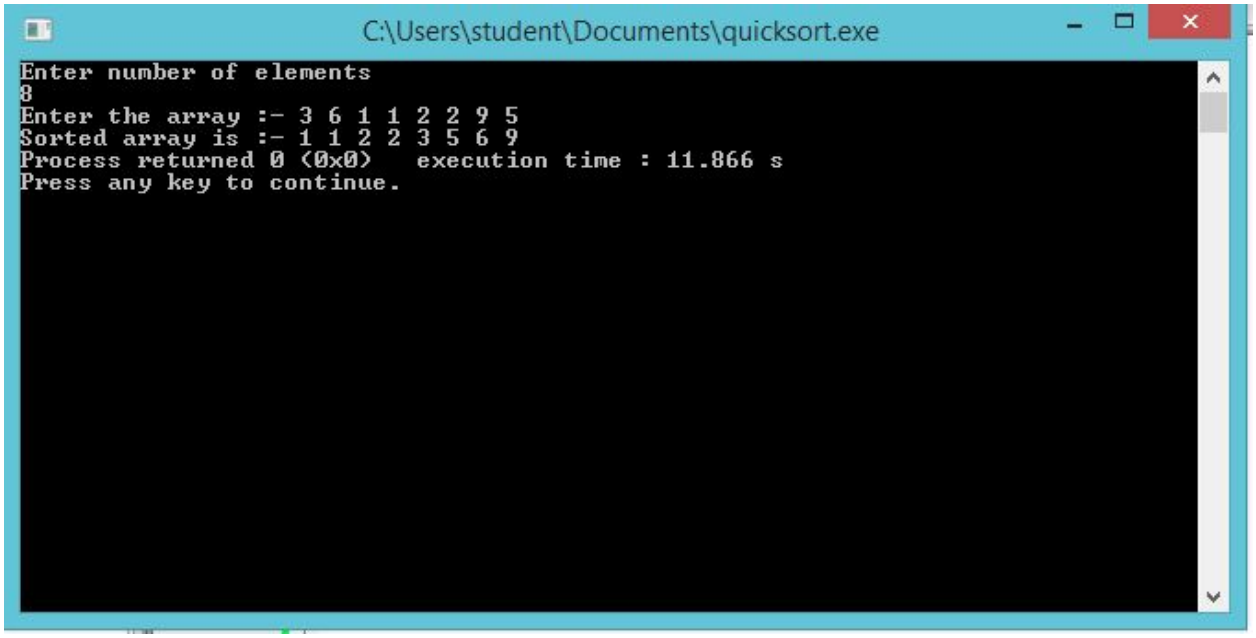

C:\Users\student\Documents\mergesort.exe
Enter the number of data element to be sorted: 8
Enter the elements: 5 3 5 8 9 1 2 7
Sorted Data 1 2 3 5 5 7 8 9
Process returned 0 (0x0)   execution time : 18.105 s
Press any key to continue.

3. ***Sort a given set of elements using the Quicksort method of divide and conquer approach. The elements can be read from a file or can be generated using the random number generator.***

```
#include<iostream>
using namespace std;

void quicksort(int ,int ,int);
int partition(int ,int,int);

int partition(int *a,int s,int e)
{
        int piviot=a[e];
   int pind=s;
   int i,t;
   for(i=s;i<e;i++)
   {
        if(a[i]<=piviot)
      {
         t=a[i];
         a[i]=a[pind];
         a[pind]=t;
         pind++;
      }
         }
   t=a[e];
   a[e]=a[pind];
   a[pind]=t;
   return pind;
}

void quicksort(int *a,int s,int e)
{
        if(s<e)
        {
                int pind=partition(a,s,e);
      quicksort(a,s,pind-1);
      quicksort(a,pind+1,e);
         }
}
```

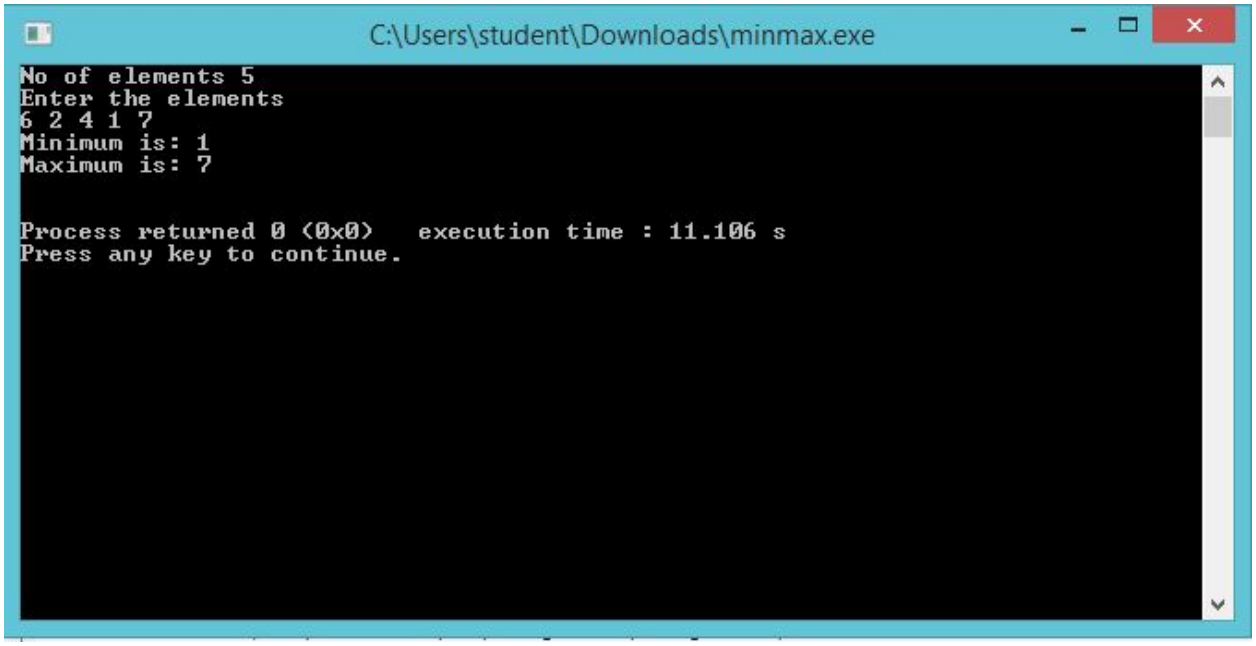
```
int main()
{
        int n;
   cout<<"Enter number of elements"<<endl;
   cin>>n;
   int a[n];
   cout<<"Enter the array :- ";
   for(int i=0;i<n;i++)
   {
        cin>>a[i];
        }
        quicksort(a,0,n-1);
   cout<<"Sorted array is :- ";
   for(int i=0;i<n;i++)
   {
        cout<<a[i]<<" ";
        }
        return 0;
}
```

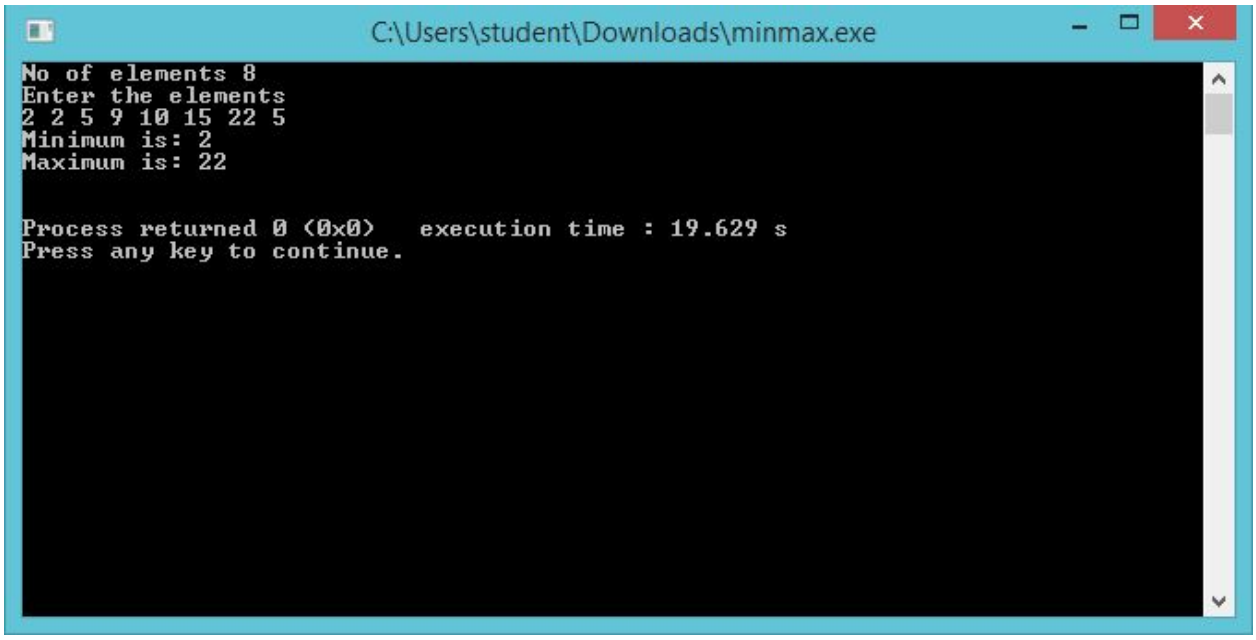

C:\Users\student\Documents\quicksort.exe
Enter number of elements
5
Enter the array :- 5 1 3 2 4
Sorted array is :- 1 2 3 4 5
Process returned 0 (0x0)   execution time : 27.211 s
Press any key to continue.


C:\Users\student\Documents\quicksort.exe
Enter number of elements
8
Enter the array :- 3 6 1 1 2 2 9 5
Sorted array is :- 1 1 2 2 3 5 6 9
Process returned 0 (0x0)   execution time : 11.866 s
Press any key to continue.

4. ***Find the minimum and maximum element from an array integer using Divide and Conquer approach.***

```
#include <iostream>
#include <limits>
using namespace std;
void maxmin(int arr[], int l, int h, int &min, int & max)
{
   if (l == h)
      min = max = arr[h];
   else
   {
      int mid = (l+h) / 2;
      int leftMin,leftMax,rightMin,rightMax;
      maxmin(arr, l, mid, leftMin, leftMax);
      maxmin(arr, mid + 1, h, rightMin, rightMax);
      if (leftMin < rightMin)
         min = leftMin;
      else
         min = rightMin;
      if (leftMax > rightMax)
         max = leftMax;
      else
         max = rightMax;
   }
}
int main()
{
   int n;
         cout<<"No of elements ";
         cin>>n;
         int arr[n];
               cout<<"Enter the elements\n";
               for(int i=0;i<n;i++){
               cin>>arr[i];
         }
   int min;
   int max;
   maxmin(arr, 0, n - 1, min, max);
   cout << "Minimum is: " << min << '\n' << "Maximum is: " << max << "\n\n";
   return 0; }
```


C:\Users\student\Downloads\minmax.exe
No of elements 5
Enter the elements
6 2 4 1 7
Minimum is: 1
Maximum is: 7
Process returned 0 (0x0)   execution time : 11.106 s
Press any key to continue.


C:\Users\student\Downloads\minmax.exe
No of elements 8
Enter the elements
2 2 5 9 10 15 22 5
Minimum is: 2
Maximum is: 22
Process returned 0 (0x0)   execution time : 19.629 s
Press any key to continue.