15B17CI371 – Data Structures Lab

ODD 2024

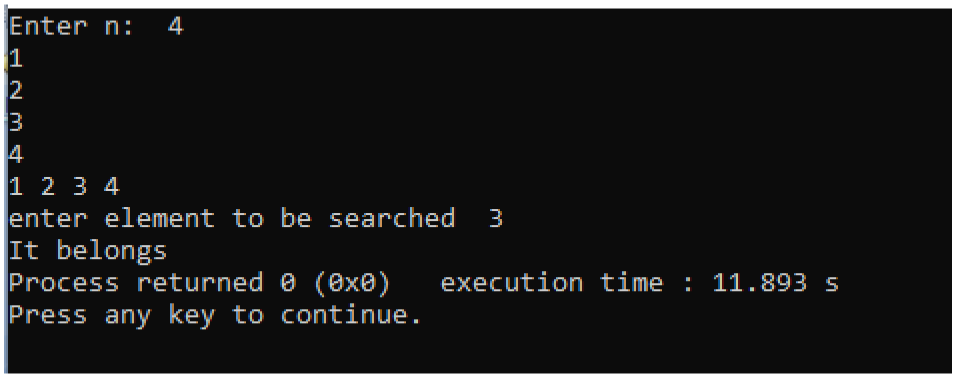
Week 5-LAB A

Practice Lab

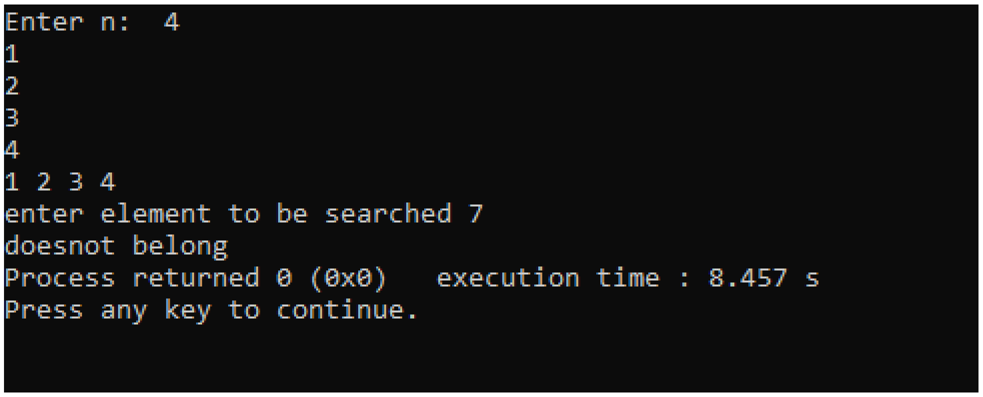
1.

```
#include <iostream>

using namespace std;

struct node

{

  int data;

  struct node* next;

};

struct node* insertatend(struct node* head,int d)

{

  if(head==NULL)

  {

    struct node* p=new struct node;

    head=p;

    p->next=NULL;

    p->data=d;

  }

  else{

     struct node*ptr=head;

    while(ptr->next!=NULL)
```

```
        {


            ptr=ptr->next;

        }

        struct node *p=new struct node;

        p->next=NULL;

        ptr->next=p;

        p->data=d;


    }

    return head;


}

void traversal(struct node* head)

{

    struct node*ptr=head;


    while(ptr!=NULL)

    {

        cout<<ptr->data<<" ";

        ptr=ptr->next;

    }

}

void belongs(struct node* head,int data)

{
    struct node*ptr=head;
    int f=0;


    while(ptr!=NULL)
    {
        if(ptr->data==data)
        {
            cout<<"It belongs";
            f=1;
            break;
        }
        ptr=ptr->next;


    }
    if(f==0)
        {
            cout<<"doesnot belong";
        }
}


int main()
{
    struct node *head=NULL;
    int n;
    cout<<"Enter n:  " ;
    cin>>n;

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

    {

        int c;

        cin>>c;

        head=insertatend(head,c);

    }

    traversal(head);

    int q;

    cout<<endl<<"enter element to be searched";

    cin>>q;

    belongs(head,q);

}
```



```
Enter n:  4
1
2
3
4
1 2 3 4
enter element to be searched  3
It belongs
Process returned 0 (0x0)   execution time : 11.893 s
Press any key to continue.
```

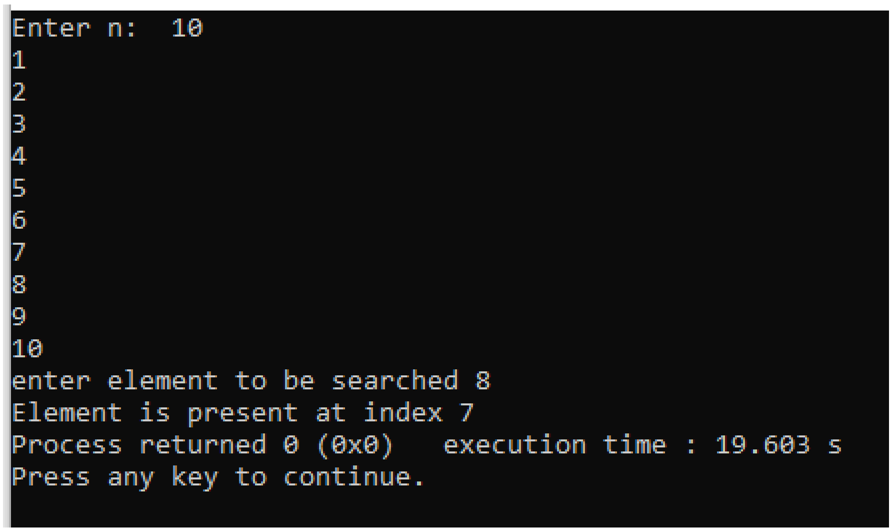


```
Enter n:  4
1
2
3
4
1 2 3 4
enter element to be searched 7
doesnot belong
Process returned 0 (0x0)   execution time : 8.457 s
Press any key to continue.
```

2.

```
#include <iostream>

using namespace std;

int binarysearch(int arr[], int low, int high, int x)

{

    while (low <= high) {

        int mid = low + (high - low) / 2;



        if (arr[mid] == x)

            return mid;






        if (arr[mid] < x)

            low = mid + 1;
        else
            high = mid - 1;
    }



    return -1;
}


int main(void)
{
        int n;
    cout<<"Enter n:  " ;
    cin>>n;
    int arr[n];
    for(int i=0;i<n;i++)
    {
       cin>>arr[i];
    }



   int x;
   cout<<"enter element to be searched";
   cin>>x;
```

```
    int result = binarysearch(arr, 0, n - 1, x);

    if(result == -1) cout << "Element is not present in array";

    else cout << "Element is present at index " << result;

    return 0;

}
```



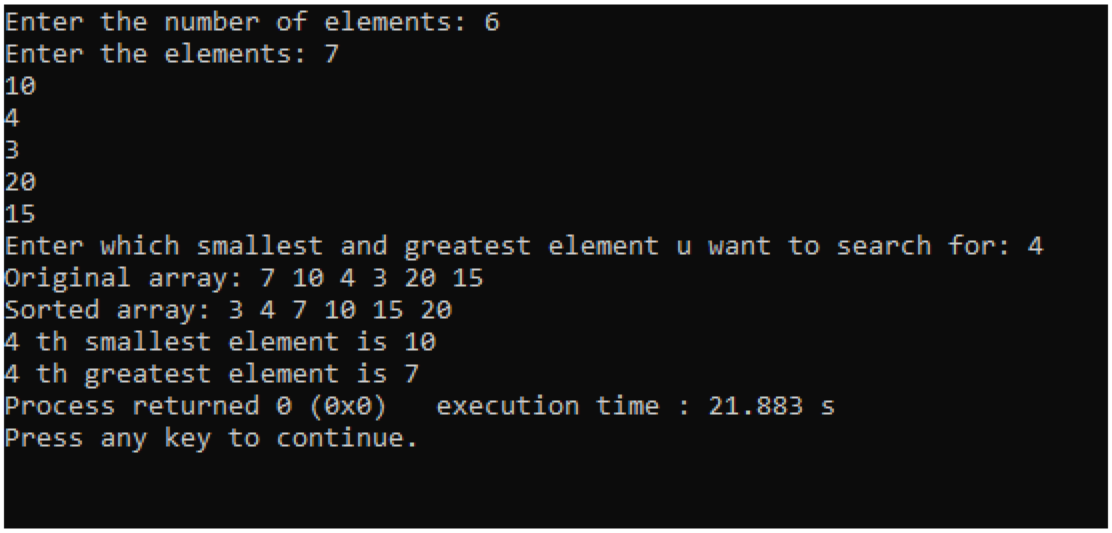
```
Enter n:  10
1
2
3
4
5
6
7
8
9
10
enter element to be searched 8
Element is present at index 7
Process returned 0 (0x0)   execution time : 19.603 s
Press any key to continue.
```

Time Complexity: O(log N) (already sorted)
Auxiliary Space: O(1)

3.

```
#include <iostream>

using namespace std;


void swap(int& a, int& b) {

    int temp = a;

    a = b;
    b = temp;
}


int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = low - 1;


    for (int j = low; j < high; ++j) {
        if (arr[j] < pivot) {
            ++i;
            swap(arr[i], arr[j]);
        }
    }
    swap(arr[i + 1], arr[high]);
    return i + 1;
}


void quickSort(int arr[], int low, int high) {
    if (low < high) {


        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
```

```
void printarray( int arr[], int size) {
    for (int i = 0; i < size; ++i) {
        cout << arr[i] << " ";
    }
    cout << endl;
}
int smallest(int arr[],int k)
{
    return arr[k-1];
}

int greatest(int arr[],int k,int n)
{
    return arr[n-k];
}



int main() {


int size;
cout << "Enter the number of elements: ";
cin >> size;
int arr[size];
cout << "Enter the elements: ";
for (int i = 0; i < size; ++i) {

cin >> arr[i];

}

int key;

cout << "Enter which smallest and greatest element u want to search for: ";

cin >> key;


  cout << "Original array: ";

  printarray(arr, size);


  quickSort(arr, 0, size - 1);


  cout << "Sorted array: ";

  printarray(arr, size);

  int x=smallest(arr,key);

  cout<<key<<" th smallest element is "<<x<<endl;

  int y=greatest(arr,key,size);

  cout<<key<<" th greatest element is "<<y;



  return 0;

}
```

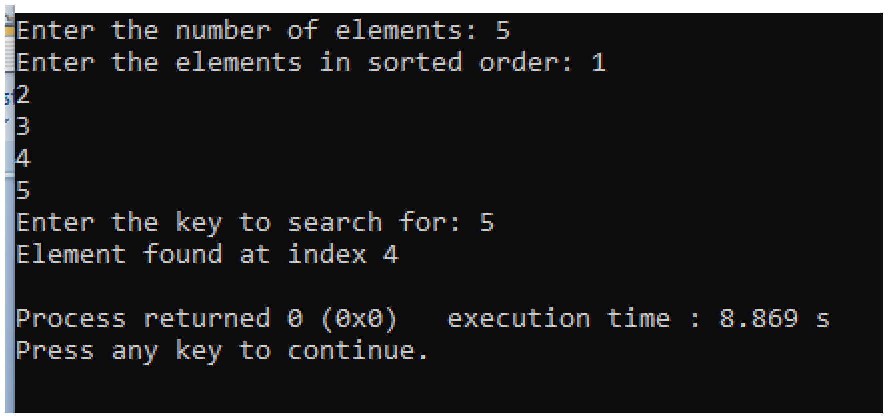


```
Enter the number of elements: 6
Enter the elements: 7
10
4
3
20
15
Enter which smallest and greatest element u want to search for: 4
Original array: 7 10 4 3 20 15
Sorted array: 3 4 7 10 15 20
4 th smallest element is 10
4 th greatest element is 7
Process returned 0 (0x0)   execution time : 21.883 s
Press any key to continue.
```

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 {
```

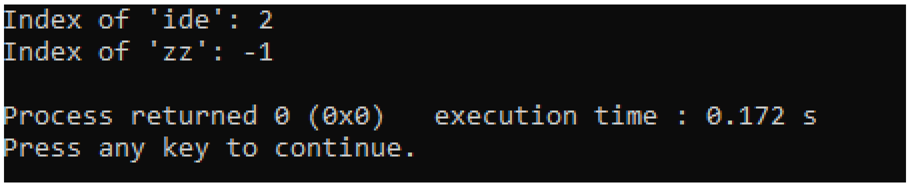
```
cout << "Element not found in the array" << endl;

}

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: 5
Element found at index 4

Process returned 0 (0x0)   execution time : 8.869 s
Press any key to continue.
```

5.

```
#include <iostream>

#include <string>

using namespace std;

int binarySearch(const string arr[], int size, const string& x) {

    int left = 0;

    int right = size - 1;



    while (left <= right) {

        int mid = left + (right - left) / 2;

        if (arr[mid] == x) {

            return mid;
        } else if (arr[mid] < x) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }

    return -1;
}

int main() {
    string arr1[] = {"Hi", "Folks", "ide", "for", "practice"};
    int size1 = sizeof(arr1) / sizeof(arr1[0]);
    string x1 = "ide";
    int index1 = binarySearch(arr1, size1, x1);
    cout << "Index of '" << x1 << "': " << index1 << endl;

    string arr2[] = {"Hi", "Folks", "ide", "for", "practic"};
    int size2 = sizeof(arr2) / sizeof(arr2[0]);
    string x2 = "zz";
    int index2 = binarySearch(arr2, size2, x2);
    cout << "Index of '" << x2 << "': " << index2 << endl;

    return 0;
}
```



```
Index of 'ide': 2
Index of 'zz': -1

Process returned 0 (0x0)   execution time : 0.172 s
Press any key to continue.
```