18CSC201J

DS LAB EXERCISES
CT-1

DATE: 20.10.2021

NAME: ANINDYA SHANKAR DASGUPTA
REG NO: RA2011026010120
CSE Q1 (AI-ML)

RA2011026010120

Simple Problems:

Q:

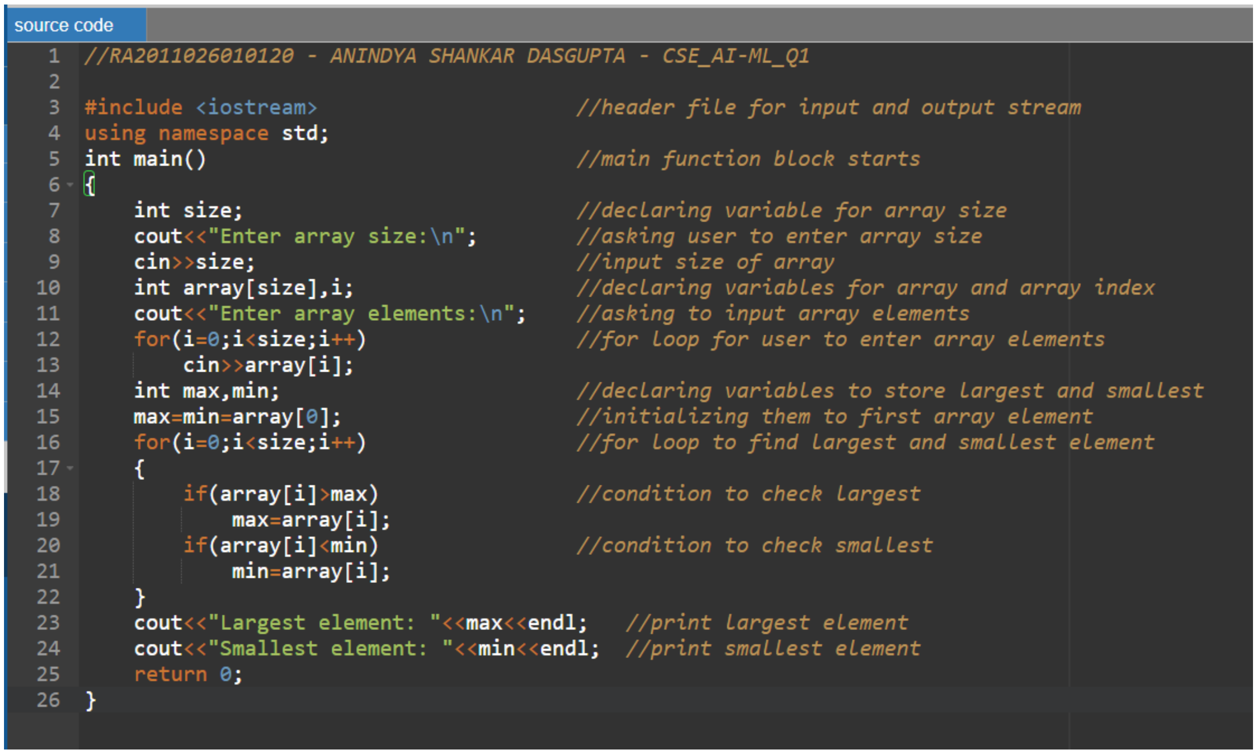
1. (4.) Print the largest and smallest element in an unsorted array.

A:

- Approach:

  1. Firstly, there can be duplicate elements in the unsorted array. In that case, there will be multiple largest and/or smallest elements present in the array, and unnecessary computational space will be used.
  2. Secondly, it is not mentioned in the question whether the array is linear or multidimensional. For multidimensional arrays, the procedure will be different. In this case, we are only computing for a linear(1D) array. For any multidimensional array, we can transfer the elements to a linear array, and use the same technique.

- Code (with documentation):



source code

```cpp
//RA2011026010120 - ANINDYA SHANKAR DASGUPTA - CSE_AI-ML_Q1

#include <iostream>                     //header file for input and output stream
using namespace std;
int main()                              //main function block starts
{
    int size;                           //declaring variable for array size
    cout<<"Enter array size:\n";        //asking user to enter array size
    cin>>size;                          //input size of array
    int array[size],i;                  //declaring variables for array and array index
    cout<<"Enter array elements:\n";    //asking to input array elements
    for(i=0;i<size;i++)                 //for loop for user to enter array elements
        cin>>array[i];
    int max,min;                        //declaring variables to store largest and smallest
    max=min=array[0];                   //initializing them to first array element
    for(i=0;i<size;i++)                 //for loop to find largest and smallest element
    {
        if(array[i]>max)                //condition to check largest
            max=array[i];
        if(array[i]<min)                //condition to check smallest
            min=array[i];
    }
    cout<<"Largest element: "<<max<<endl;   //print largest element
    cout<<"Smallest element: "<<min<<endl;  //print smallest element
    return 0;
}
```

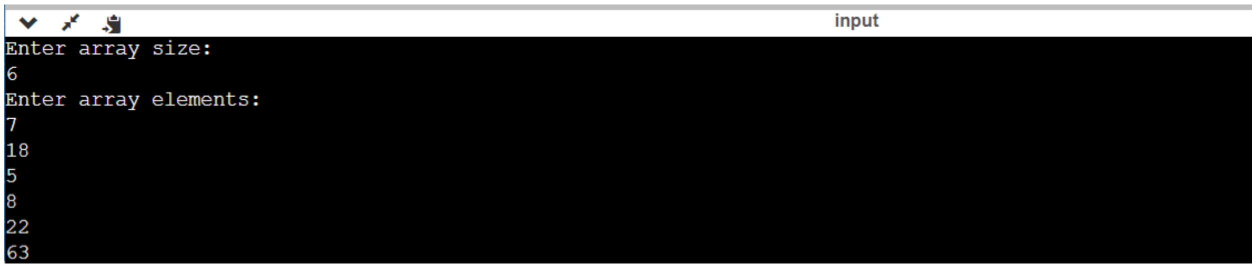
- <u>Dry-run:</u>
  Sample input:
  size=6
  array[6]={7,18,5,8,22,63} (- : garbage value)

Line no.	size	array[0]	array[1]	array[2]	array[3]	array[4]	array[5]	i	max	min
5	-									
9	6									
10	6	-	-	-	-	-	-	-		
12	6	7	-	-	-	-	-	0		
	6	7	18	-	-	-	-	1		
For loop	6	7	18	5	-	-	-	2		
	6	7	18	5	8	-	-	3		
	6	7	18	5	8	22	-	4		
13	6	7	18	5	8	22	63	5		
13	6	7	18	5	8	22	63	5		
14	6	7	18	5	8	22	63	5	-	-
15	6	7	18	5	8	22	63	5	7	7
16	6	7	18	5	8	22	63	0	7	7
	6	7	18	5	8	22	63	1	18	7
For loop	6	7	18	5	8	22	63	2	18	5
	6	7	18	5	8	22	63	3	18	5
	6	7	18	5	8	22	63	4	22	5
22	6	7	18	5	8	22	63	5	63	5
26	6	7	18	5	8	22	63	5	63	5

Clearly, Output: Largest element = max = 63
Smallest element = min = 5

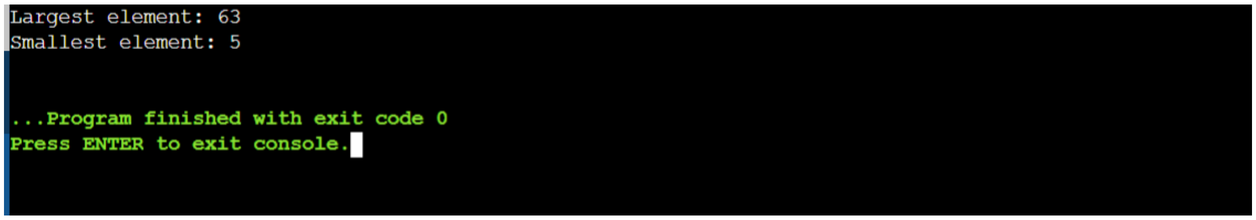
- Input/Output:

Input-



```
Enter array size:
6
Enter array elements:
7
18
5
8
22
63
```

Output-



```
Largest element: 63
Smallest element: 5


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

- Result:
The code was compiled successfully and executed in the C++ compiler to give the required output.
    - Time complexity:

Statement	Frequency
For i=0 to size-1	size+1
cin>>array[i]	1
For i =0 to size-1	size+1
if(array[i]>max)	size
max=arr[i]	1
if(array[i]<min)	size
min=arr[i]	1
All other lines of code	1*10 = 10
TOTAL	**4*size+15**

As the Step count of algorithm is 4*size+15, which means linear grown with time. Also, asymptotic upper bound running time = O(4*size+15) = O(size).

- Space complexity = no. of each type of variable * memory occupied by each such variable

  = 4 + 4 + 4 + 4 + (4*size) ---->memory allocated to array

  = 16 + (4*size) bytes

  (4 integer variables + 1 integer array of size = size)

  Asymptotic upper bound of space occupied = O(4*size+16) = O(size).

RA2011026010120

Scenario based or Difficult Problems:

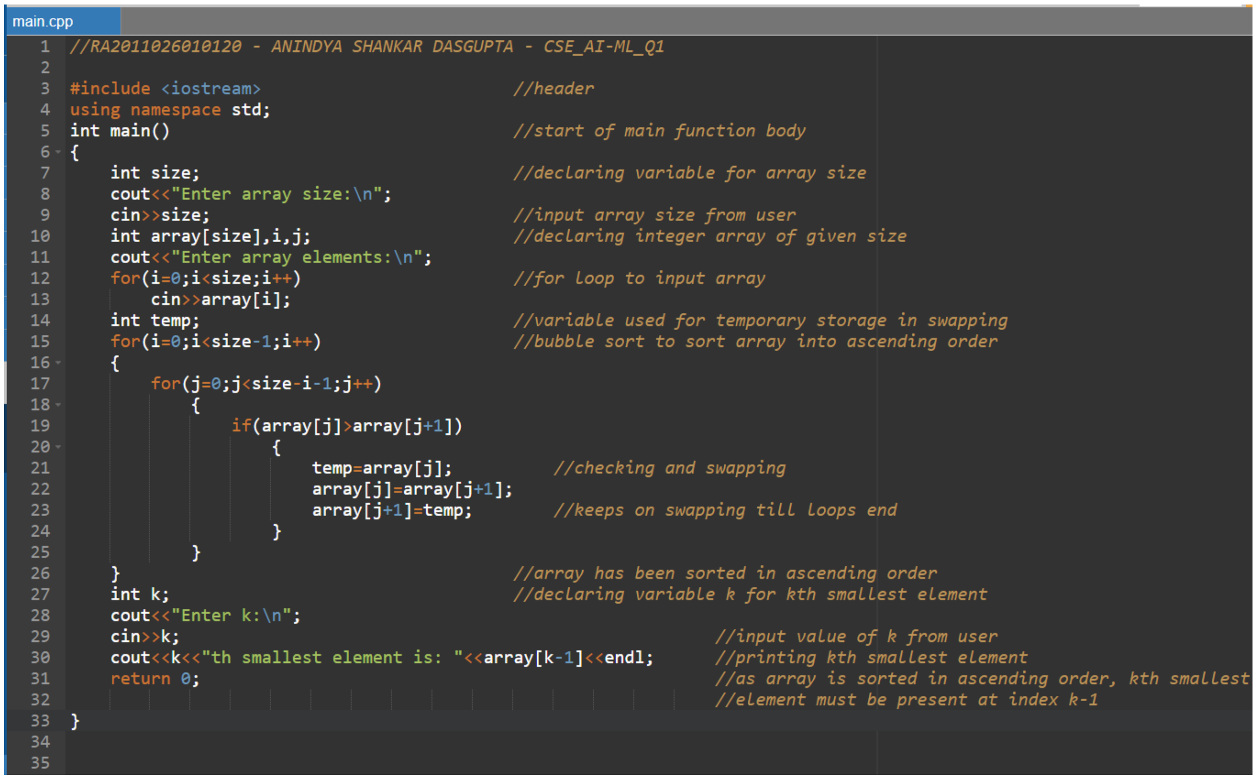
Q:

1. (13.) Given an array **arr[]** and an integer **K** where K is smaller than size of array, the task is to find the **Kth smallest** element in the given array. It is given that all array elements are distinct.

A:

- Approach:
    1. Firstly, it is not stated within the question whether the given array is sorted or not. If it is unsorted we need to sort the array first. Here, we will deal with an unsorted array.
    2. Secondly, it is not given whether the array can be multidimensional or not.Here, we deal with a linear array. In case, the array is multidimensional we can transfer the array elements to a linear array and use the same logic thereafter.

- Code (with documentation):



main.cpp

```
//RA2011026010120 - ANINDYA SHANKAR DASGUPTA - CSE_AI-ML_Q1

#include <iostream>                           //header
using namespace std;
int main()                                    //start of main function body
{
    int size;                                 //declaring variable for array size
    cout<<"Enter array size:\n";
    cin>>size;                                //input array size from user
    int array[size],i,j;                      //declaring integer array of given size
    cout<<"Enter array elements:\n";
    for(i=0;i<size;i++)                       //for loop to input array
        cin>>array[i];
    int temp;                                 //variable used for temporary storage in swapping
    for(i=0;i<size-1;i++)                     //bubble sort to sort array into ascending order
    {
        for(j=0;j<size-i-1;j++)
        {
            if(array[j]>array[j+1])
            {
                temp=array[j];                //checking and swapping
                array[j]=array[j+1];
                array[j+1]=temp;              //keeps on swapping till loops end
            }
        }
    }                                         //array has been sorted in ascending order
    int k;                                    //declaring variable k for kth smallest element
    cout<<"Enter k:\n";
    cin>>k;                                                     //input value of k from user
    cout<<k<<"th smallest element is: "<<array[k-1]<<endl;      //printing kth smallest element
    return 0;                                                   //as array is sorted in ascending order, kth smallest
                                                                //element must be present at index k-1
}
```

- <u>Dry-run:</u>

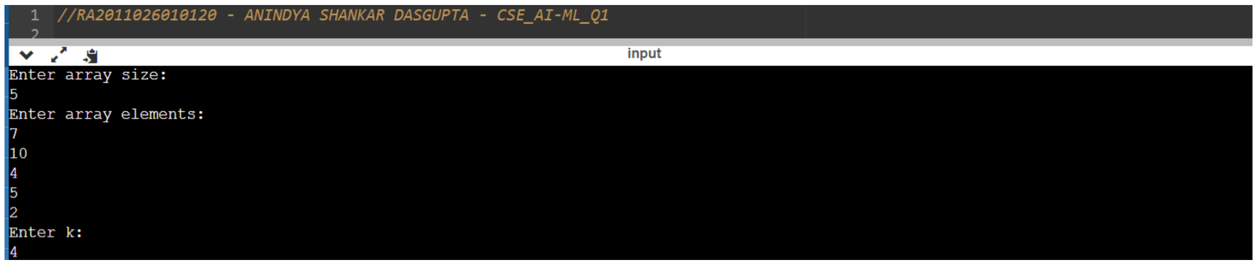
Sample input: size = 5 array[5] = {7,10,4,5,2} k = 4

Line no.	size	array [0]	array [1]	array [2]	array [3]	array [4]	i	j	k	temp	array [k-1]
7	-										
9	5										
10	5	-	-	-	-	-	-	-			
12	5	7	-	-	-	-	0	-			
	5	7	10	-	-	-	1	-			
For loop	5	7	10	4	-	-	2	-			
	5	7	10	4	5	-	3	-			
13	5	7	10	4	5	2	4	-			
14	5	7	10	4	5	2	4	-		-	
15	5	7	10	4	5	2	0	0		-	
	5	7	4	10	5	2	0	1		10	
	5	7	4	5	10	2	0	2		10	
	5	7	4	5	2	10	0	3		10	
Bub-ble sort	5	7	4	5	2	10	1	0		7	
	5	4	7	5	2	10	1	1		7	
	5	4	5	7	2	10	1	2		7	
	5	4	5	2	7	10	2	0		5	
	5	4	2	5	7	10	2	1		5	
26	5	2	4	5	7	10	3	0		4	
27	5	2	4	5	7	10	3	0	-	4	
29	5	2	4	5	7	10	3	0	4	4	7
30	5	2	4	5	7	10	3	0	4	4	7

Sample output: k = 4. So, 4th element is 7.

- <u>Input/Output:</u>

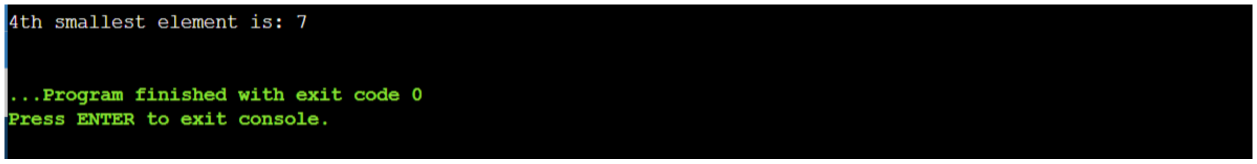
Input-



```
//RA2011026010120 - ANINDYA SHANKAR DASGUPTA - CSE_AI-ML_Q1
```

```
Enter array size:
5
Enter array elements:
7
10
4
5
2
Enter k:
4
```

Output-



```
4th smallest element is: 7


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

- <u>Result:</u>

The code was compiled successfully and executed in the C++ compiler to give the required output.

  - Time complexity:

Statement	Frequency
For i=0 to i=size-1	size+1
cin>>array[i]	1
For i=0 to i=size-2	size
For j=0 to j=size-i-2	size*size (worst-case scenario)
if(array[i]>array[i+1])	size*size
temp=array[i]	size
array[i]=array[i+1]	size
array[i+1]=temp	size
All other lines of code	1*11
TOTAL	**2size²+5size+13**

Step count of algorithm is 2size²+5size+13, which means quadratic grown with time. Asymptotic upper bound time complexity = <u>O(2size²+5size+13) = O(size²).</u>

- Space complexity = no. of each type of variable * memory occupied by each variable

  = 4*5 + (4*size) bytes

  (5 integer variables + 1 integer array of size = size)

  = 20 + (4*size) bytes

Asymptotic upper-bound of space complexity = O(4size+20) = O(size).