

#### ALGO LAB 4

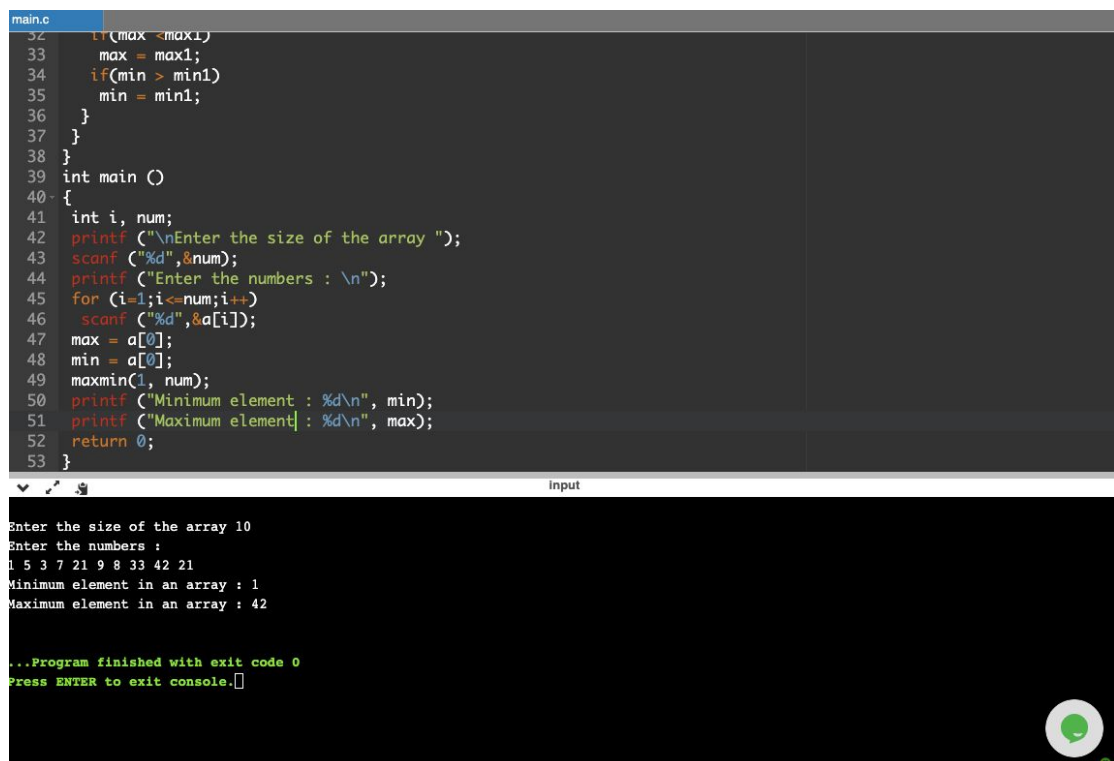
1. Given a set S of distinct integers, use the divide and conquer approach to write a recursive algorithm that finds the kth smallest integer in S.

ANS.

```
#include<stdio.h>
int max, min;
int a[100];
void
maxmin (int i, int j)
{
    int max1, min1, mid;
    if (i == j)
    {
        max = min = a[i];
    }
    else
    {
        if (i == j - 1)
        {
            if (a[i] < a[j])
            {
                max = a[j];
                min = a[i];
            }
            else
            {
                max = a[i];
                min = a[j];
            }
        }
        else
        {
            mid = (i + j) / 2;
            maxmin (i, mid);
            max1 = max;
            min1 = min;
            maxmin (mid + 1, j);
            if (max < max1)
                max = max1;
            if (min > min1)
                min = min1;
        }
    }
}
```

```
}

int
main ()
{
    int i, num;
    printf ("\nEnter the size of the array ");
    scanf ("%d", &num);
    printf ("Enter the numbers : \n");
    for (i = 1; i <= num; i++)
        scanf ("%d", &a[i]);
    max = a[0];
    min = a[0];
    maxmin (1, num);
    printf ("Minimum element : %d\n", min);
    printf ("Maximum element : %d\n", max);
    return 0;
}
```



The screenshot displays a C program in a code editor and its execution in a terminal window. The code defines an array 'a' of size 100, prompts the user for the array size and elements, and then finds the minimum and maximum values using a recursive function 'maxmin'. The terminal output shows the user entering 10 for the size and 10 numbers, resulting in a minimum of 1 and a maximum of 42.

```
main.c
32     if(max < max1)
33         max = max1;
34     if(min > min1)
35         min = min1;
36     }
37 }
38 }
39 int main ()
40 {
41     int i, num;
42     printf ("\nEnter the size of the array ");
43     scanf ("%d",&num);
44     printf ("Enter the numbers : \n");
45     for (i=1;i<=num;i++)
46         scanf ("%d",&a[i]);
47     max = a[0];
48     min = a[0];
49     maxmin(1, num);
50     printf ("Minimum element : %d\n", min);
51     printf ("Maximum element : %d\n", max);
52     return 0;
53 }
```

Input

```
Enter the size of the array 10
Enter the numbers :
1 5 3 7 21 9 8 33 42 21
Minimum element in an array : 1
Maximum element in an array : 42

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

2. Given a set S of distinct integers, use the divide and conquer approach to write a recursive algorithm that outputs the minimum and maximum element of S.

```
#include<stdio.h>
//int iter = 0;
void printIt(int *arr,int n) {
    printf("\n");
    for(int c = 0; c < n ; c++)
        printf("%d ", arr[c]);
    printf("\n");
}

int mean(int* arr, int n) {
    int sum=0;
    for(int i=0; i<n; i++) {
        sum=sum+arr[i];
    }
    return(sum/n);
}

void swap(int* xp, int* yp) {
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

int findPosition(int arr[],int arr_c[],int n,int mean) {
    int i, j, min_idx,value,flag = 0;

    for (i = 0; i < n - 1; i++) {
        min_idx = i;
        for (j = i + 1; j < n; j++)
            if (arr[j] < arr[min_idx])
                min_idx = j;
        swap(&arr[min_idx], &arr[i]);
    }

    for(int i=0; i<n; i++) {
        if(arr[i] >= mean && flag == 0) {
            //printf("%d = %d = %d\n",mean,arr[i],i);
            value = arr[i];
            flag =1;
            break;
        }
    }
}
```

```
    }  
}  
  
for(int i=0; i<n; i++) {  
    if(value == arr_c[i]) {  
        //printf("%d = %d = %d\n",mean,arr_c[i],i);  
        return i;  
    }  
}  
}
```

```
int partition(int *arr, int n,int pos,int k) {
```

```
    //printf(arr,n);  
    //printf("\n");  
    int arr_left[20]={0},left = 0;  
    int arr_right[20]={0},right = 0;  
    //printf("\n%d -- %d\n",pos,n);  
    int mean_local = arr[pos];  
  
    for(int c = 0; c < n ; c++) {  
        if( c == pos)  
            continue;  
        //printf("\n%d - %d",arr[c],mean);  
        if( arr[c] < mean_local ) {  
            arr_left[left] = arr[c];  
            left++;  
        }  
        else if( arr[c] >= mean_local) {  
            arr_right[right] = arr[c];  
            right++;  
        }  
    }  
}
```

```
// printf("\nleft - ");  
// printf(arr_left,left);
```

```
// printf("\nright - ");  
// printf(arr_right,right);
```

```
//printf("\n%d -- %d\n",pos,mean);
```

```
for(int c = 0; c < left ; c++) {  
    arr[c] = arr_left[c];
```

```
}

arr[left] = mean_local;

for(int c = 0; c < right ; c++) {
    arr[left+1+c] = arr_right[c];
}
//printf("%d = %d - %d",iter++,mean_local,k-1);
if (mean_local==arr[k-1])
    return mean_local;
if (mean_local>arr[k]) {
    //partition(copied1,n,pos,k);
    int copied[16],copied1[16];
    int loop;
    for(loop = 0; loop < n; loop++) {
        copied[loop] = arr_left[loop];
        copied1[loop]= arr_left[loop];
    }
    int mean_n = mean(arr_left,left);
    int pos = findPosition(arr_left,copied,left,mean_n);
    //printf(copied1,left);
    //printf("left = ");
    partition(copied1,left,pos,k);
}
else if (mean_local<arr[k-1]) {
    int copied[16],copied1[16];
    int loop;
    for(loop = 0; loop < n; loop++) {
        copied[loop] = arr_right[loop];
        copied1[loop]= arr_right[loop];
    }
    int mean_n = mean(arr_right,right);
    int pos = findPosition(arr_right,copied,right,mean_n);
    //printf("right = ");
    //printf(copied1,right);
    partition(copied1,right,pos,k);
}

}

}

int main()
{
    int arr[]={8,53,87,53,23,6,3,8,10,20,30,40,53};
    int i = 0;
```

```
int k = 3; //K'th smallest

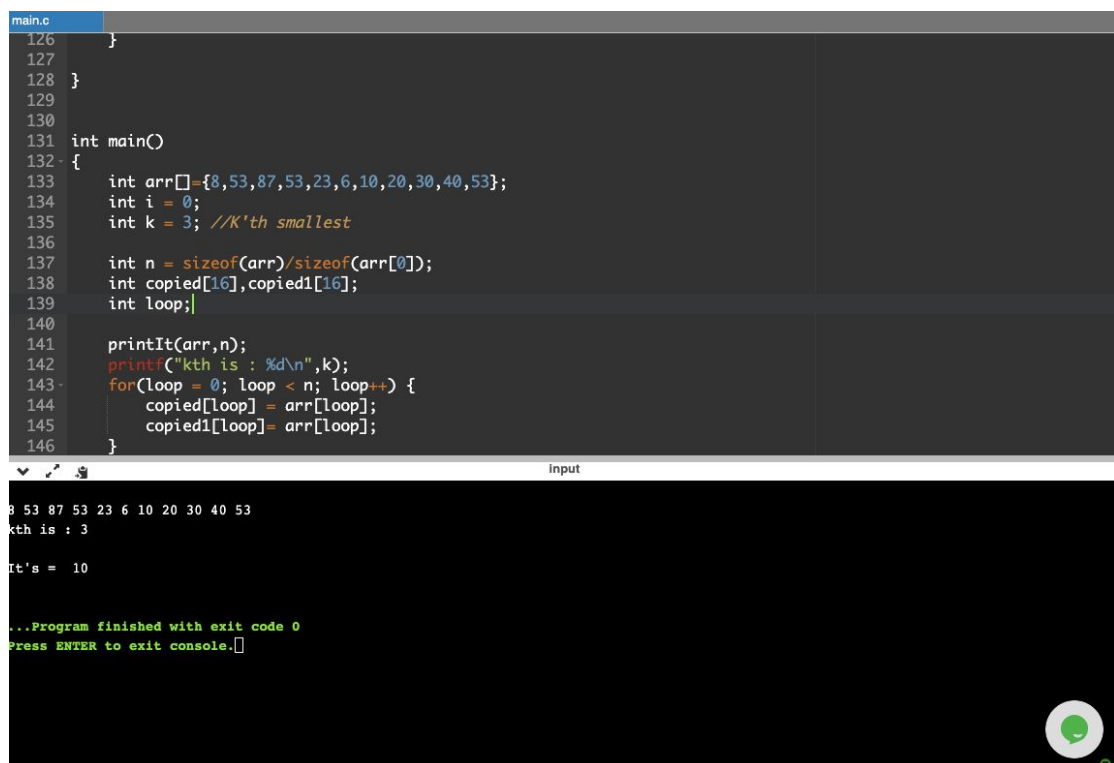
int n = sizeof(arr)/sizeof(arr[0]);
int copied[16],copied1[16];
int loop;

for(loop = 0; loop < n; loop++) {
    copied[loop] = arr[loop];
    copied1[loop]= arr[loop];
}

int mean_n = mean(arr,n);
int pos = findPosition(arr,copied,n,mean_n);

int ans = partition(copied1,n,pos,k);
printf("\nIt's = %d \n",ans);

//printf("\n \t %d",copied1[ans]);
//printIt(copied1,n);
return 0;
}
```



The screenshot displays a C program in a code editor and its execution output in a terminal window. The code in `main.c` defines an array `arr` with 16 elements: `{8, 53, 87, 53, 23, 6, 10, 20, 30, 40, 53}`. It calculates the size of the array, copies it into `copied` and `copied1` arrays, and prints the array. It then finds the position of the 3rd smallest element (k=3) and partitions the array around it. The output in the terminal window shows the array elements, the value of k (3), the position of the 3rd smallest element (10), and the final partitioned array.

```
main.c
126 }
127 }
128 }
129
130
131 int main()
132 {
133     int arr[]={8,53,87,53,23,6,10,20,30,40,53};
134     int i = 0;
135     int k = 3; //K'th smallest
136
137     int n = sizeof(arr)/sizeof(arr[0]);
138     int copied[16],copied1[16];
139     int loop;
140
141     printIt(arr,n);
142     printf("kth is : %d\n",k);
143     for(loop = 0; loop < n; loop++) {
144         copied[loop] = arr[loop];
145         copied1[loop]= arr[loop];
146     }
147 }

input
8 53 87 53 23 6 10 20 30 40 53
kth is : 3
It's = 10

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