//all op on max heap(replacement)

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

int n=0;

void create_max_heap(int arr[]);
int extract_max(int arr[]);
void insert_key(int arr[]);
void increase_key(int arr[]);
void decrease_key(int arr[]);
void delete_key(int arr[]);
void heap_sort(int arr[]);
void display(int arr[]);
void increase(int arr[],int id,int key);
void max_heapify(int arr[],int i);

int main(){
	int arr[20],ch;

	while(ch!=8){
		printf("MAIN MENU\n");
		printf("1.create max heap\n2.extract max ele\n3.insert key\n4.increase key\n5.decrease key\n6.delete key\n7.heap sort\n8.exit\n");
		printf("enter your choice: ");
		scanf("%d",&ch);
		switch(ch){
			case 1:create_max_heap(arr);
			break;
			case 2:
				printf("enter the size of array(heap): ");
```

```
        scanf("%d",&n);
        printf("\nenter the array eles\n");
        for(int i=0;i<n;i++){
                scanf("%d",&arr[i]);
        }
        printf("the array(heap) is: ");
        for(int i=0;i<n;i++){
                printf("%d ",arr[i]);
        }
        printf("\n");
            while(n>0){
                    int max_ele=extract_max(arr);
                    printf("extracted max element:%d\n",max_ele);
              }
           break;
                        case 3:insert_key(arr);
                        break;
                        case 4:increase_key(arr);
                        break;
                        case 5:decrease_key(arr);
                        break;
                        case 6:delete_key(arr);
                        break;
                        case 7:heap_sort(arr);
                        break;
                        case 8:exit(0);
                        default:
                        printf("invalid choice\n");
                }
        }
}
```

```
void create_max_heap(int arr[]){
	printf("enter the size of array(heap): ");
	scanf("%d",&n);
	printf("\nenter the array eles\n");
	for(int i=0;i<n;i++){
		scanf("%d",&arr[i]);
	}
	printf("the array(heap) is: ");
	for(int i=0;i<n;i++){
		printf("%d ",arr[i]);
	}
	int largest_non_leaf=(n-1)/2;
	for(int i=largest_non_leaf;i>=0;i--){
		max_heapify(arr,i);
	}
	printf("\nthe max heap is: ");
	display(arr);
}

int extract_max(int arr[]){
int max_ele=arr[0];
	arr[0]=arr[n-1];
	n--;
	max_heapify(arr,0);
	return max_ele;
}

void insert_key(int arr[]){
	int key;
```

```
	printf("enter the size of array(heap): ");
	scanf("%d",&n);
	printf("\nenter the array eles\n");
	for(int i=0;i<n;i++){
		scanf("%d",&arr[i]);
	}
	printf("the array(heap) is: ");
	for(int i=0;i<n;i++){
		printf("%d ",arr[i]);
	}
	printf("\nenter the val to be inserted: ");
	scanf("%d",&key);
	n++;
	arr[n-1]=-99999;
	increase(arr,n-1,key);
	printf("\nafter insert key op the heap(max heap) is: ");
	display(arr);
}

void increase_key(int arr[]){
	int key,id;
	printf("enter the size of array(heap): ");
	scanf("%d",&n);
	printf("\nenter the array eles\n");
	for(int i=0;i<n;i++){
		scanf("%d",&arr[i]);
	}
	printf("the array(heap) is: ");
	for(int i=0;i<n;i++){
		printf("%d ",arr[i]);
	}
```

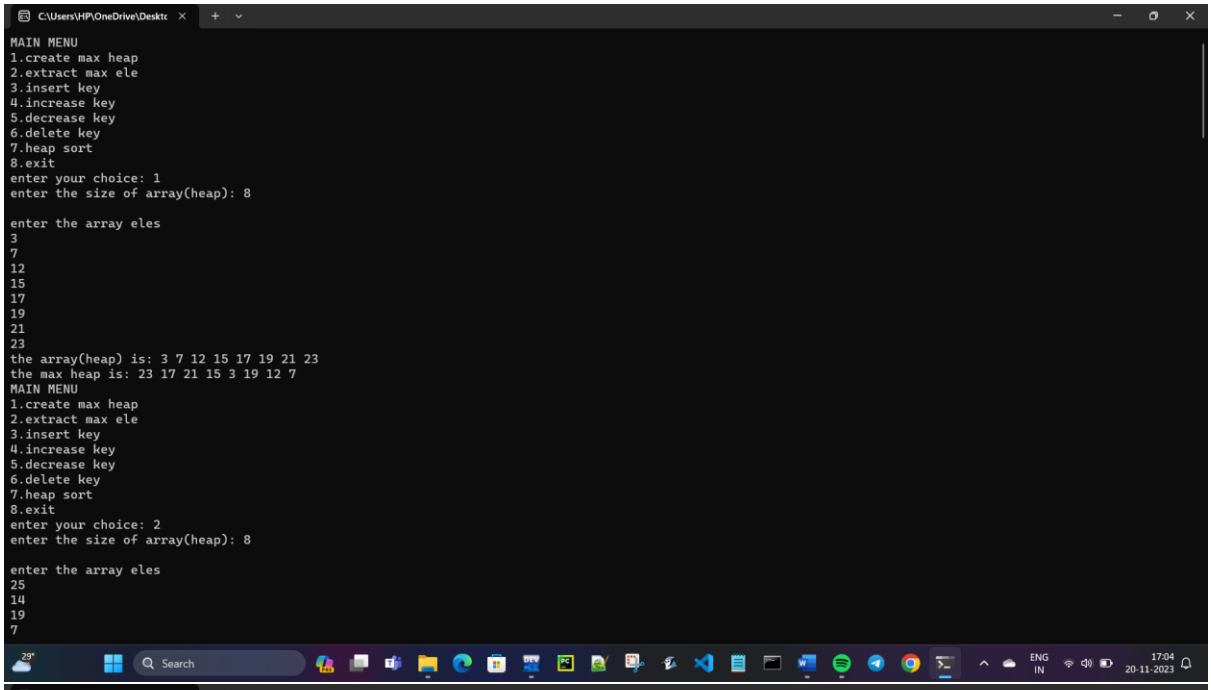
```
	printf("\nenter the index of the element to be increased: ");
	scanf("%d",&id);
	printf("\nenter the increased val: ");
	scanf("%d",&key);
	increase(arr,id,key);
	printf("\nafter increase key the heap(max heap) is: ");
	display(arr);
}


void decrease_key(int arr[]){
	int key,id;
	printf("enter the size of array(heap): ");
	scanf("%d",&n);
	printf("\nenter the array eles\n");
	for(int i=0;i<n;i++){
		scanf("%d",&arr[i]);
	}
	printf("the array(heap) is: ");
	for(int i=0;i<n;i++){
		printf("%d ",arr[i]);
	}
	printf("\nenter the index of the element to be decreased: ");
	scanf("%d",&id);
	printf("\nenter the decreased val: ");
	scanf("%d",&key);
	if(arr[id]<key){
		printf("ERROR:the element is already lesser than key\n");
	}
	else{
		arr[id]=key;
```

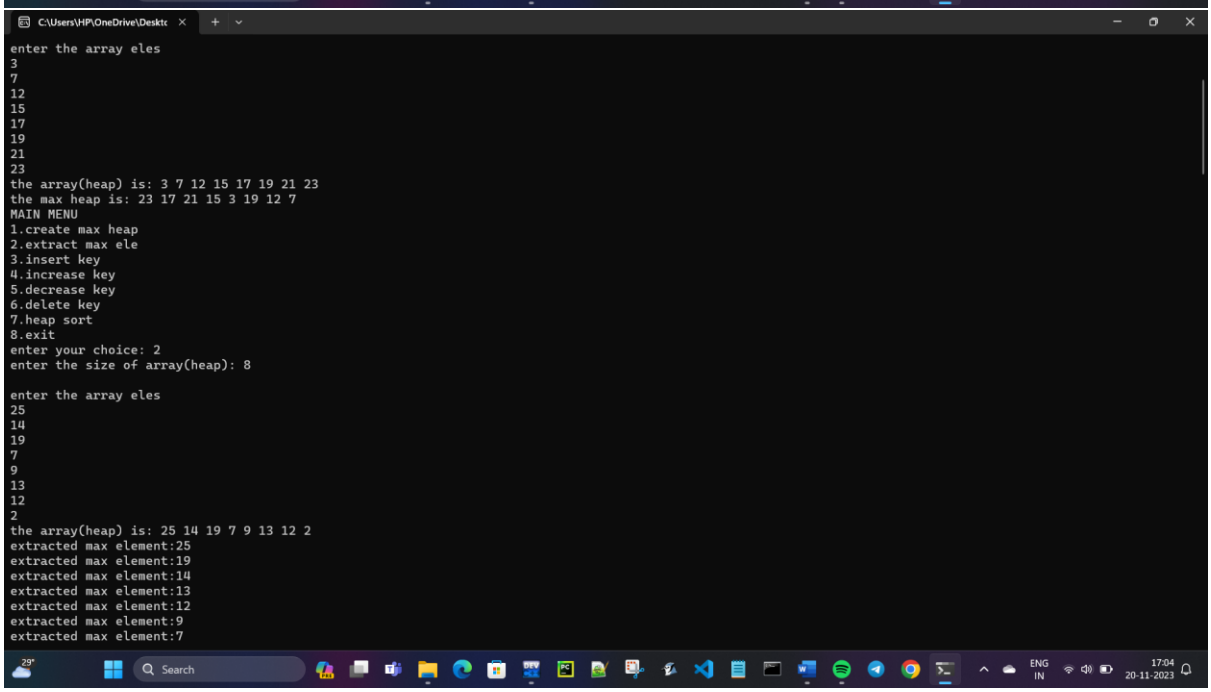
```
                max_heapify(arr,id); //as key<arr[id],arr[id]<arr[root],so key<arr[root] so no need to apply mh from root
        }
        printf("\nafter decrease key the heap(max heap) is: ");
        display(arr);
}

void delete_key(int arr[]){
        int id;
        printf("enter the size of array(heap): ");
        scanf("%d",&n);
        printf("\nenter the array eles\n");
        for(int i=0;i<n;i++){
                scanf("%d",&arr[i]);
        }
        printf("the array(heap) is: ");
        for(int i=0;i<n;i++){
                printf("%d ",arr[i]);
        }
        printf("\nenter the index of the element to be deleted: ");
        scanf("%d",&id);
        int del_ele=arr[id];
        arr[id]=arr[n-1];
        n--;
        printf("\n%d is deleted\n",del_ele);
        max_heapify(arr,id);
        printf("\nafter deletion the heap(max heap) is: ");
        display(arr);
}

void heap_sort(int arr[]){
```

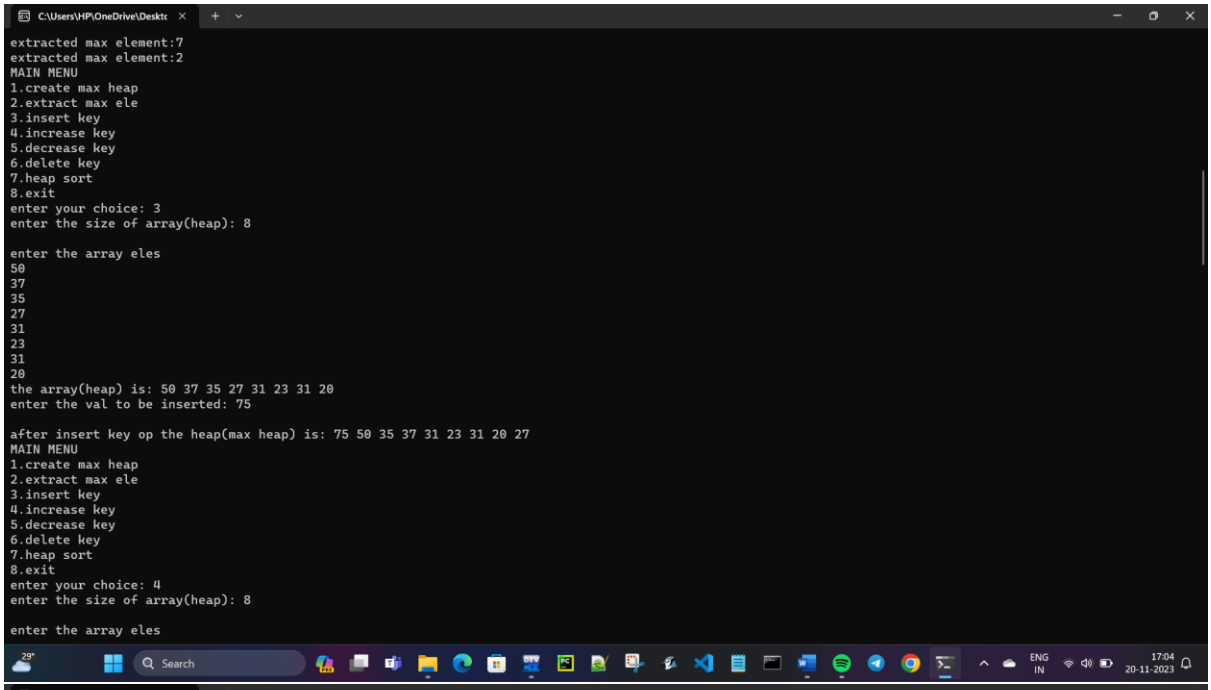
```
	printf("enter the size of array(heap): ");
	scanf("%d",&n);
	printf("\nenter the array eles\n");
	for(int i=0;i<n;i++){
		scanf("%d",&arr[i]);
	}
	printf("the array(heap) is: ");
	for(int i=0;i<n;i++){
		printf("%d ",arr[i]);
	}
	int copy=n;
	for(int i=n-1;i>=1;i--){  //as when n=2 then arr[0] will be swapped with arr[1](as it is a mh so arr[0]>arr[1],arr[1]=2nd last smallest,arr[0]=smallest after heap sort),when n=1 then there are no eles to compare with arr[root] so the loop will run till 1
		int max=arr[0];
		arr[0]=arr[i];
		arr[i]=max;
		n--;   //now the loop will not count the max and heapify the full heap from root till n-1(n-1 doesnt includes the max)
		max_heapify(arr,0);
	}
	n=copy;
	printf("\nafter heap sort the heap(max heap) is: ");
	for(int i=n-1;i>=0;i--){
		printf("%d ",arr[i]);
	}
	printf("\n");
}

void display(int arr[]){
	for(int i=0;i<n;i++){
		printf("%d ",arr[i]);
```

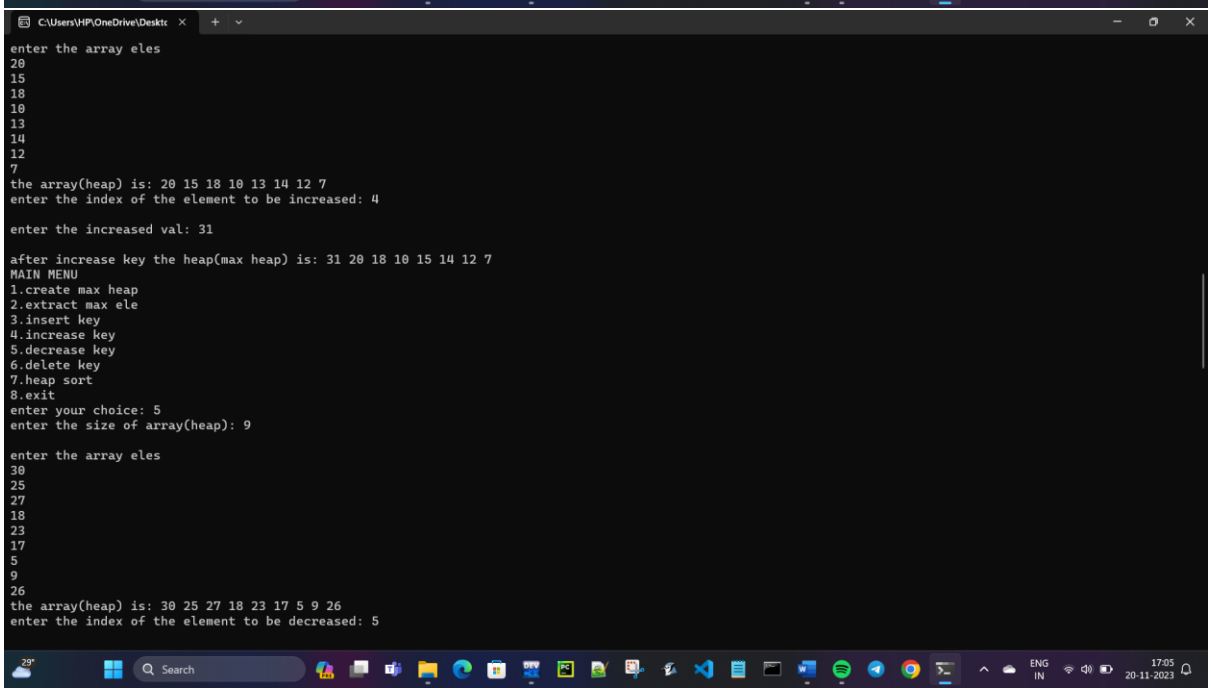
```
	}
	printf("\n");
}

void increase(int arr[],int id,int key){
	if(arr[id]>key){
		printf("ERROR:the element is already greater than key\n");
	}
	else{
		arr[id]=key;
	   while(id>0 && arr[(id-1)/2]<arr[id])  //as key>arr[id],arr[id]>all eles of that subtree(root=id),so key>all eles of that subtree(root=id) so no need to apply mh from root to end,may the increased value is greater then its parent so apply mh to id's parent till root
		{
	   int temp=arr[(id-1)/2];
	   arr[(id-1)/2]=arr[id];
	   arr[id]=temp;
	   id=(id-1)/2;
	}
}
}

void max_heapify(int arr[],int i){
	int rc,lc,largest;
	lc=2*i+1;
	rc=2*i+2;
	if(lc<n && arr[i]<arr[lc]){
		largest=lc;
	}
	else{
		largest=i;
	}
```

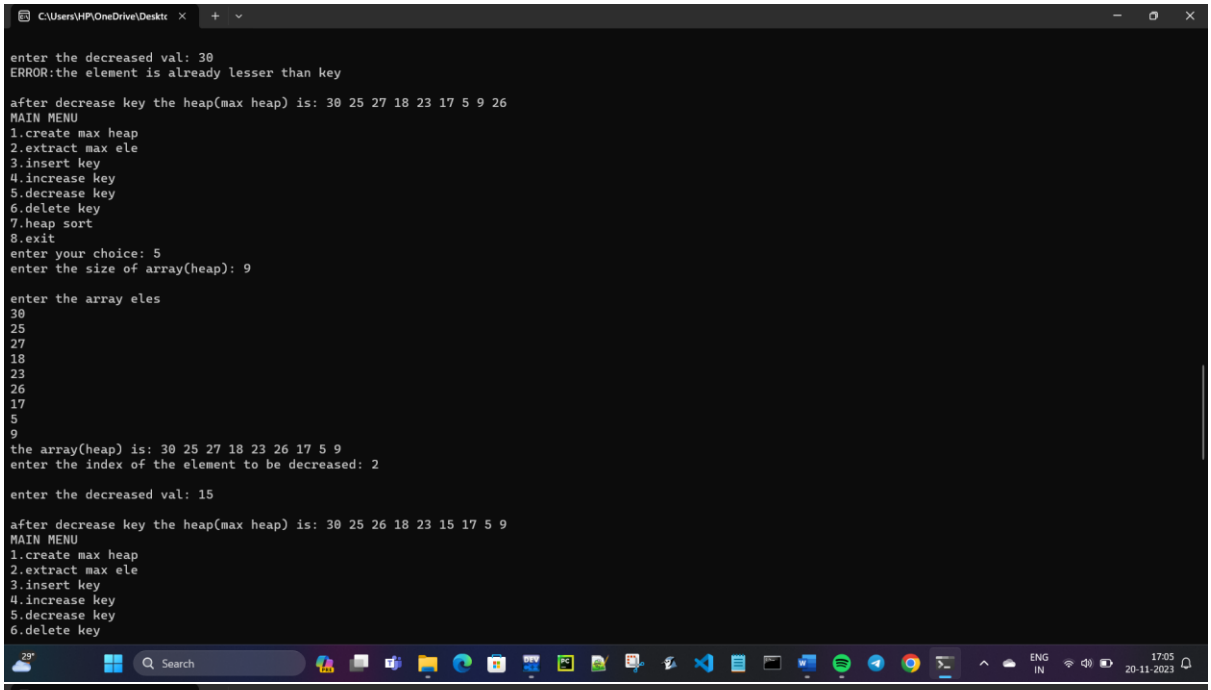
```
        if(rc<n && arr[largest]<arr[rc]){
                largest=rc;
        }
        if(largest!=i){
                int temp=arr[i];
                arr[i]=arr[largest];
                arr[largest]=temp;
                max_heapify(arr,largest);
        }
}
```

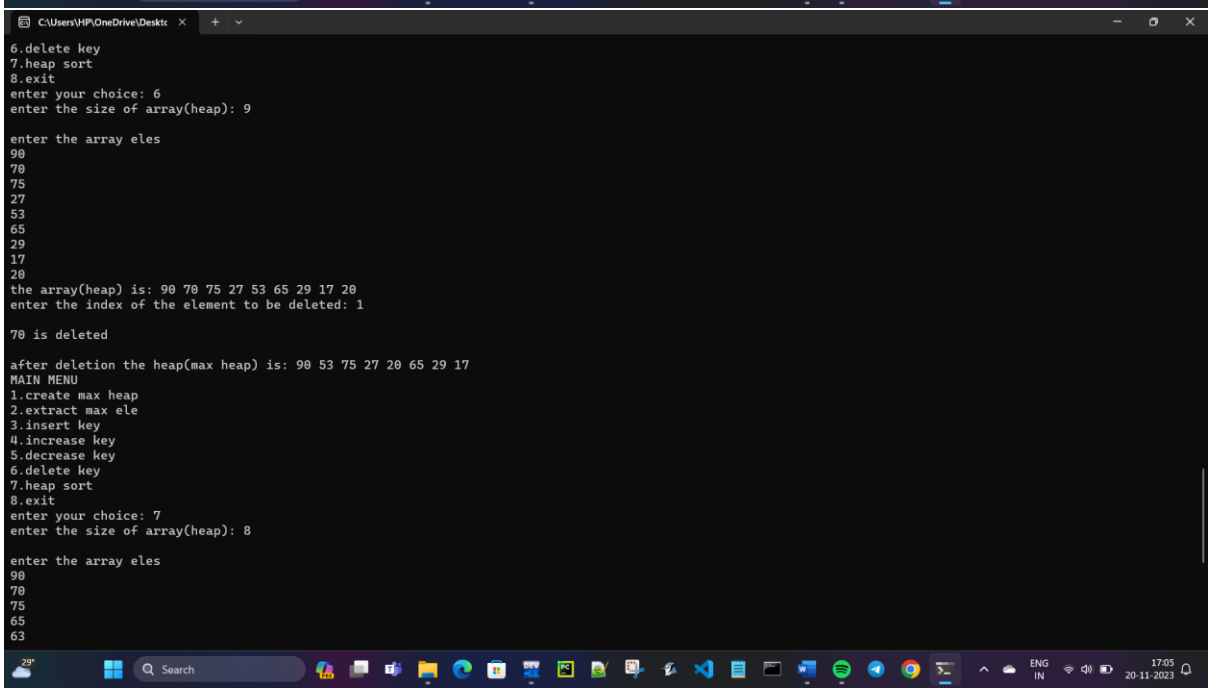

C:\Users\HP\OneDrive\Desktc
MAIN MENU
1.create max heap
2.extract max ele
3.insert key
4.increase key
5.decrease key
6.delete key
7.heap sort
8.exit
enter your choice: 1
enter the size of array(heap): 8
enter the array eles
3
7
12
15
17
19
21
23
the array(heap) is: 3 7 12 15 17 19 21 23
the max heap is: 23 17 21 15 3 19 12 7
MAIN MENU
1.create max heap
2.extract max ele
3.insert key
4.increase key
5.decrease key
6.delete key
7.heap sort
8.exit
enter your choice: 2
enter the size of array(heap): 8
enter the array eles
25
14
19
7
Search
ENG
IN
17:04
20-11-2023

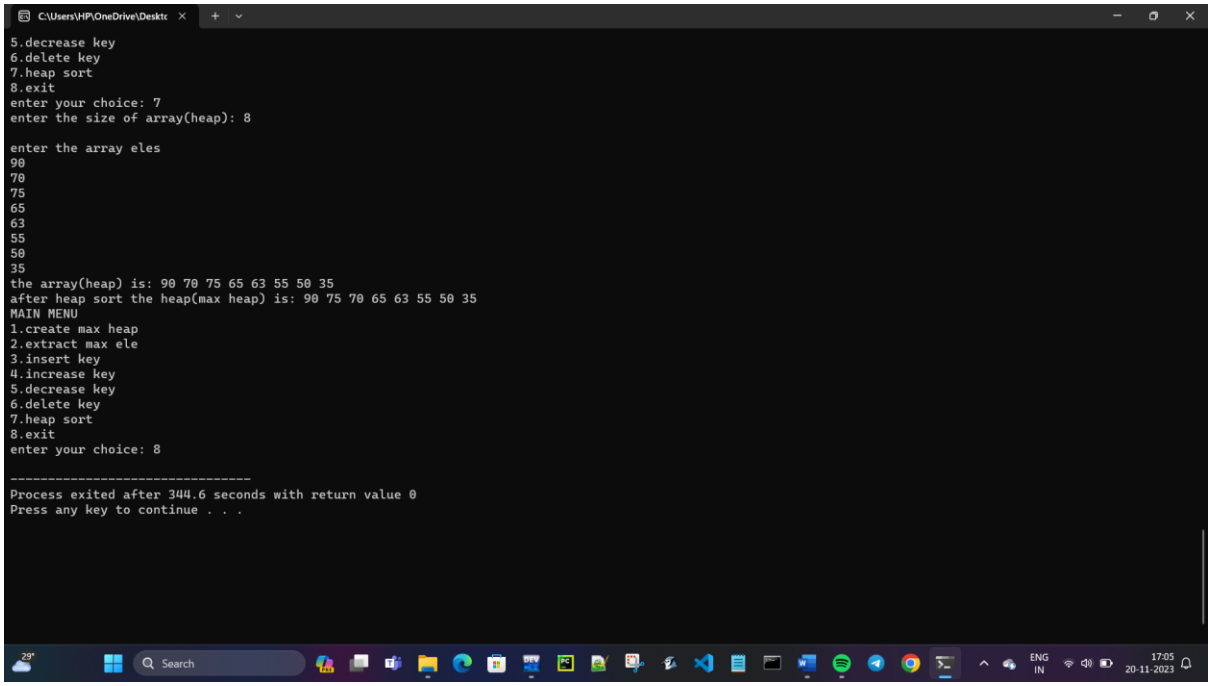

C:\Users\HP\OneDrive\Desktc
enter the array eles
3
7
12
15
17
19
21
23
the array(heap) is: 3 7 12 15 17 19 21 23
the max heap is: 23 17 21 15 3 19 12 7
MAIN MENU
1.create max heap
2.extract max ele
3.insert key
4.increase key
5.decrease key
6.delete key
7.heap sort
8.exit
enter your choice: 2
enter the size of array(heap): 8
enter the array eles
25
14
19
7
9
13
12
2
the array(heap) is: 25 14 19 7 9 13 12 2
extracted max element:25
extracted max element:19
extracted max element:14
extracted max element:13
extracted max element:12
extracted max element:9
extracted max element:7
Search
ENG
IN
17:04
20-11-2023


C:\Users\HP\OneDrive\Desktc
extracted max element:7
extracted max element:2
MAIN MENU
1.create max heap
2.extract max ele
3.insert key
4.increase key
5.decrease key
6.delete key
7.heap sort
8.exit
enter your choice: 3
enter the size of array(heap): 8
enter the array eles
50
37
35
27
31
23
31
20
the array(heap) is: 50 37 35 27 31 23 31 20
enter the val to be inserted: 75
after insert key op the heap(max heap) is: 75 50 35 37 31 23 31 20 27
MAIN MENU
1.create max heap
2.extract max ele
3.insert key
4.increase key
5.decrease key
6.delete key
7.heap sort
8.exit
enter your choice: 4
enter the size of array(heap): 8
enter the array eles
Search
ENG
IN
17:04
20-11-2023


C:\Users\HP\OneDrive\Desktc
enter the array eles
20
15
18
10
13
14
12
7
the array(heap) is: 20 15 18 10 13 14 12 7
enter the index of the element to be increased: 4
enter the increased val: 31
after increase key the heap(max heap) is: 31 20 18 10 15 14 12 7
MAIN MENU
1.create max heap
2.extract max ele
3.insert key
4.increase key
5.decrease key
6.delete key
7.heap sort
8.exit
enter your choice: 5
enter the size of array(heap): 9
enter the array eles
30
25
27
18
23
17
5
9
26
the array(heap) is: 30 25 27 18 23 17 5 9 26
enter the index of the element to be decreased: 5
Search
ENG
IN
17:05
20-11-2023


C:\Users\HP\OneDrive\Desktc
enter the decreased val: 30
ERROR:the element is already lesser than key
after decrease key the heap(max heap) is: 30 25 27 18 23 17 5 9 26
MAIN MENU
1.create max heap
2.extract max ele
3.insert key
4.increase key
5.decrease key
6.delete key
7.heap sort
8.exit
enter your choice: 5
enter the size of array(heap): 9
enter the array eles
30
25
27
18
23
26
17
5
9
the array(heap) is: 30 25 27 18 23 26 17 5 9
enter the index of the element to be decreased: 2
enter the decreased val: 15
after decrease key the heap(max heap) is: 30 25 26 18 23 15 17 5 9
MAIN MENU
1.create max heap
2.extract max ele
3.insert key
4.increase key
5.decrease key
6.delete key
Search
ENG
IN
17:05
20-11-2023


C:\Users\HP\OneDrive\Desktc
6.delete key
7.heap sort
8.exit
enter your choice: 6
enter the size of array(heap): 9
enter the array eles
90
70
75
27
53
65
29
17
20
the array(heap) is: 90 70 75 27 53 65 29 17 20
enter the index of the element to be deleted: 1
70 is deleted
after deletion the heap(max heap) is: 90 53 75 27 20 65 29 17
MAIN MENU
1.create max heap
2.extract max ele
3.insert key
4.increase key
5.decrease key
6.delete key
7.heap sort
8.exit
enter your choice: 7
enter the size of array(heap): 8
enter the array eles
90
70
75
65
63
Search
ENG
IN
17:05
20-11-2023


C:\Users\HP\OneDrive\Desktc
5.decrease key
6.delete key
7.heap sort
8.exit
enter your choice: 7
enter the size of array(heap): 8
enter the array eles
90
70
75
65
63
55
50
35
the array(heap) is: 90 70 75 65 63 55 50 35
after heap sort the heap(max heap) is: 90 75 70 65 63 55 50 35
MAIN MENU
1.create max heap
2.extract max ele
3.insert key
4.increase key
5.decrease key
6.delete key
7.heap sort
8.exit
enter your choice: 8
Process exited after 344.6 seconds with return value 0
Press any key to continue . . .
29°
Search
ENG
IN
17:05
20-11-2023