//delete elem from max heap(by creating max heap)

```
#include<stdio.h>
int n=0;

void create_max_heap(int arr[]);
void delete_key(int arr[],int id);
void max_heapify(int arr[],int i);

int main(){
	int arr[20],i,id;
	printf("enter the size of array(heap): ");
	scanf("%d",&n);
	printf("\nenter the array eles\n");
	for(i=0;i<n;i++){
		scanf("%d",&arr[i]);
	}
	printf("the array(heap) is: ");
	for(i=0;i<n;i++){
		printf("%d ",arr[i]);
	}
	create_max_heap(arr);
	printf("\nafter max heapify the array(heap) is: ");
	for(i=0;i<n;i++){
		printf("%d ",arr[i]);
	}
	printf("\nenter the index of the element to be deleted: ");
	scanf("%d",&id);
	delete_key(arr,id);
	printf("\nafter deletion the heap(max heap) is: ");
	for(i=0;i<n;i++){
		printf("%d ",arr[i]);
```

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

void create_max_heap(int arr[]){
        int largest_non_leaf=(n-1)/2;
        for(int i=largest_non_leaf;i>=0;i--){
                max_heapify(arr,i);
        }
}

void delete_key(int arr[],int id){
        int del_ele=arr[id];
        arr[id]=arr[n-1];
        n--;
        printf("\n%d is deleted\n",del_ele);
        max_heapify(arr,id);   //no need to hepify from root as the last elem must smaller then lvl-root+1,root+2..root+(n-1) but so the sbtrees of index id need to heapify but that lat elem is already smaller than lvl:root,rot+1,root+(id-1),so no need to heapify from the 1st
}

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);
        }
}
```