//all op on max heap(giving random element & creating a new array(heap all time)

```
#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 create_mh(int arr[]);
void increase(int arr[],int id,int key);
void max_heapify(int arr[],int i);

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

	while(ch!=8){
		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]);
}
    create_mh(arr);
printf("the array(max 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]);
        }
        create_mh(arr);
        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]);
	}
	create_mh(arr);
	printf("the array(max 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]);
	}
	create_mh(arr);
	printf("the array(max 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]);
        }
        create_mh(arr);
        printf("the array(max 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]);
        }
        create_mh(arr);
        printf("the array(max 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]);
	}
	create_mh(arr);
	printf("the array(max 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 create_mh(int arr[]){
	int largest_non_leaf=(n-1)/2;
	for(int i=largest_non_leaf;i>=0;i--){
		max_heapify(arr,i);
	}
}

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);
        }
}
```