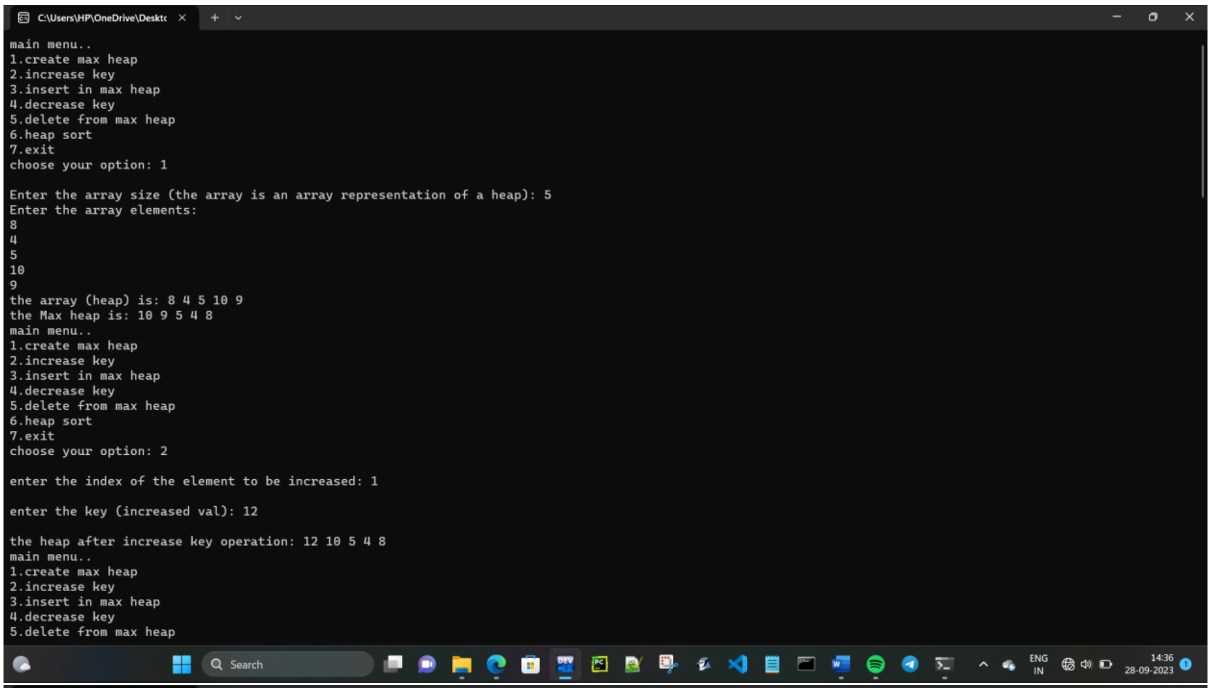
/*all operations on max heap*/

```
#include <stdio.h>

#include<stdlib.h>


int n = 0;


void create_max_heap(int arr[]);

void increase_key(int arr[]);

void insert_key(int arr[]);

void decrease_key(int arr[]);

void delete_key(int arr[]);

void heap_sort(int arr[]);

void display(int arr[]);

void max_heapify(int arr[],int id);

void increase(int arr[],int id,int key);


int main()

{

int ch=0,arr[20];


while(ch!=7){

printf("main menu..\n");

printf("1.create max heap\n2.increase key\n3.insert in max heap\n4.decrease key\n5.delete from
max heap\n6.heap sort\n7.exit\n");

printf("choose your option: ");

scanf("%d",&ch);

printf("\n");


switch(ch){

case 1:create_max_heap(arr);
break;
case 2:increase_key(arr);
break;
case 3:insert_key(arr);
break;
case 4:decrease_key(arr);
break;
case 5:delete_key(arr);
break;
case 6:heap_sort(arr);
break;
case 7:exit(0);
default:
printf("invalid choice\n");
}
}
return 0;
}

void create_max_heap(int arr[]) {
        printf("Enter the array size (the array is an array representation of a heap): ");
scanf("%d", &n);
printf("Enter the array elements:\n");
for (int i = 0; i < n; i++)
{
scanf("%d", &arr[i]);
}
printf("the array (heap) is: ");
display(arr);
int last_non_leaf = (n - 1) / 2;
for (int i = last_non_leaf; i >= 0; i--) {
```

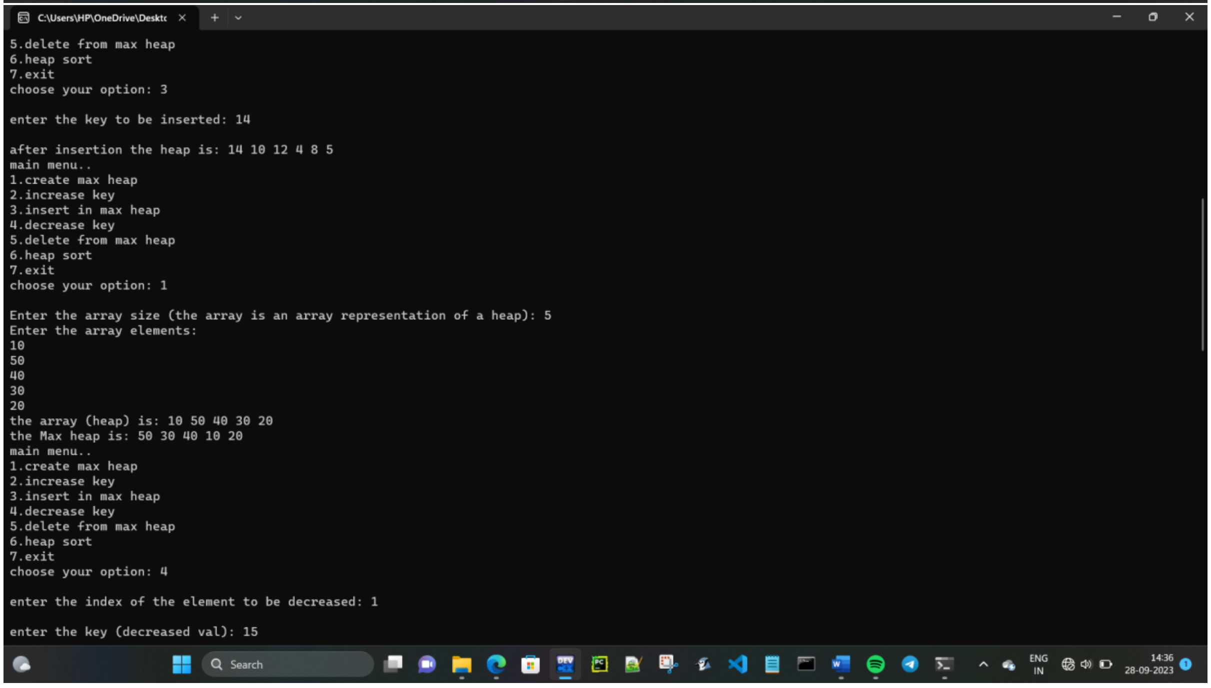
```
max_heapify(arr,i);
}
printf("the Max heap is: ");
display(arr);
}

void increase_key(int arr[]) {
        int id=0,key=0;
printf("enter the index of the element to be increased: ");
scanf("%d", &id);
printf("\nenter the key (increased val): ");
scanf("%d", &key);
printf("\n");
increase(arr,id,key);
printf("the heap after increase key operation: ");
display(arr);
}

void insert_key(int arr[])
{
        int key=0;
printf("enter the key to be inserted: ");
scanf("%d", &key);
printf("\n");
n++;
arr[n-1]=-99999;
int id=n-1;
increase(arr,id,key);
printf("after insertion the heap is: ");
display(arr);
}
```

```
void decrease_key(int arr[])
{
        int id=0,key=0;
printf("enter the index of the element to be decreased: ");
scanf("%d", &id);
printf("\nenter the key (decreased val): ");
scanf("%d", &key);
printf("\n");
if (arr[id] < key)
{
printf("ERROR: node value already lesser than key");
return;
}
arr[id] = key;
max_heapify(arr,id);
printf("the heap after decrease key operation: ");
display(arr);
}

void delete_key(int arr[])
{
        int id=0;
printf("enter the index of the key to be deleted: ");
scanf("%d", &id);
printf("\n");
int del_ele=arr[id];
arr[id]=arr[n-1];
n--;
max_heapify(arr,id);
printf("%d deleted\n",del_ele);
printf("after deletion the heap is: ");
display(arr);
}



void heap_sort(int arr[]) {
int copy = n; //n=size
//create_max_heap(arr);
for (int i = n - 1; i >= 1; i--) {
int max = arr[0];
arr[0] = arr[i];
arr[i] = max;
n--;
max_heapify(arr,0);
}
n=copy;
printf("After heap sort, the sorted array is: ");
display(arr);
}

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

void max_heapify(int arr[], int i) {
   int lc, rc, largest;
   lc = 2 * i + 1;
   rc = 2 * i + 2;
```

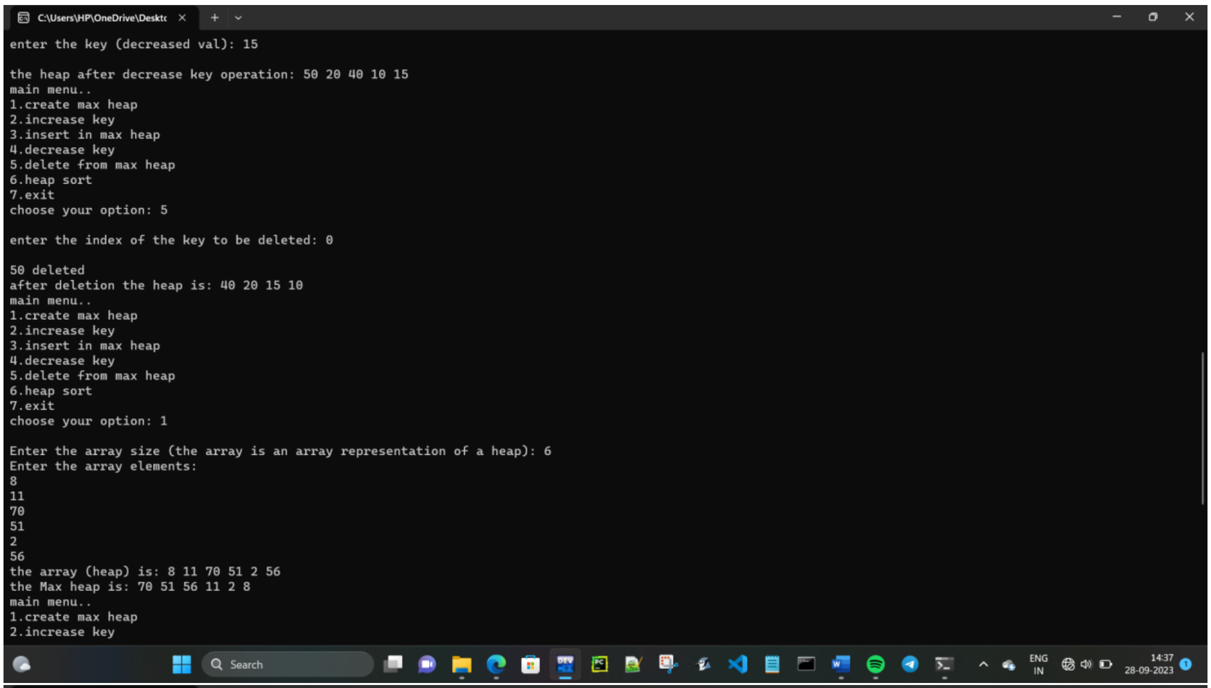
```
    if (lc < n && arr[lc] > arr[i]) {
        largest = lc;
    } else {
        largest = i;
    }
    if (rc < n && arr[rc] > arr[largest]) {
        largest = rc;
    }
    if (largest != i) {
        int temp = arr[i];
        arr[i] = arr[largest];
        arr[largest] = temp;
        max_heapify(arr,largest);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

void increase(int arr[], int id, int key) {
   if (arr[id] > key) {
        printf("ERROR: node value already greater than key");
        return;
    }
    arr[id] = key;
    while (id > 0 && arr[(id - 1)/ 2] < arr[id]) {
        int temp = arr[(id - 1)/ 2];
        arr[(id - 1)/ 2] = arr[id];
        arr[id] = temp;
        id = (id - 1)/ 2;
    }
}
```



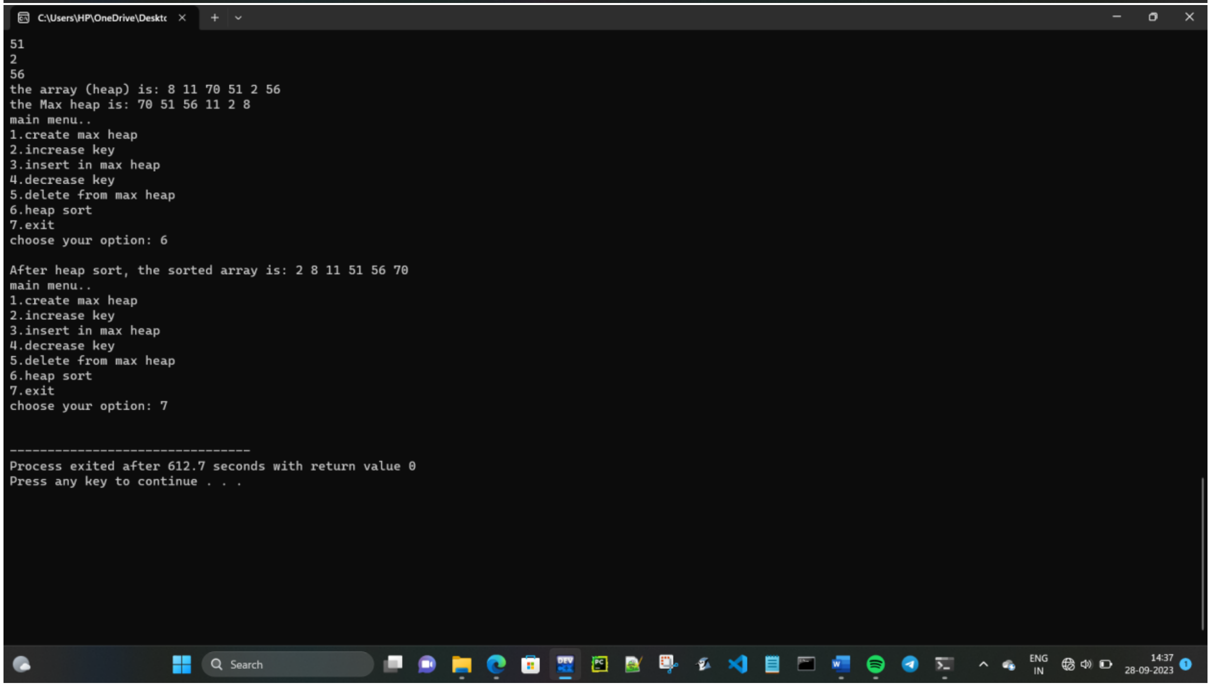
```
main menu..
1.create max heap
2.increase key
3.insert in max heap
4.decrease key
5.delete from max heap
6.heap sort
7.exit
choose your option: 1

Enter the array size (the array is an array representation of a heap): 5
Enter the array elements:
8
4
5
10
9
the array (heap) is: 8 4 5 10 9
the Max heap is: 10 9 5 4 8
main menu..
1.create max heap
2.increase key
3.insert in max heap
4.decrease key
5.delete from max heap
6.heap sort
7.exit
choose your option: 2

enter the index of the element to be increased: 1

enter the key (increased val): 12

the heap after increase key operation: 12 10 5 4 8
main menu..
1.create max heap
2.increase key
3.insert in max heap
4.decrease key
5.delete from max heap
```



```
5.delete from max heap
6.heap sort
7.exit
choose your option: 3

enter the key to be inserted: 14

after insertion the heap is: 14 10 12 4 8 5
main menu..
1.create max heap
2.increase key
3.insert in max heap
4.decrease key
5.delete from max heap
6.heap sort
7.exit
choose your option: 1

Enter the array size (the array is an array representation of a heap): 5
Enter the array elements:
10
50
40
30
20
the array (heap) is: 10 50 40 30 20
the Max heap is: 50 30 40 10 20
main menu..
1.create max heap
2.increase key
3.insert in max heap
4.decrease key
5.delete from max heap
6.heap sort
7.exit
choose your option: 4

enter the index of the element to be decreased: 1

enter the key (decreased val): 15
```



```
enter the key (decreased val): 15

the heap after decrease key operation: 50 20 40 10 15
main menu..
1.create max heap
2.increase key
3.insert in max heap
4.decrease key
5.delete from max heap
6.heap sort
7.exit
choose your option: 5

enter the index of the key to be deleted: 0

50 deleted
after deletion the heap is: 40 20 15 10
main menu..
1.create max heap
2.increase key
3.insert in max heap
4.decrease key
5.delete from max heap
6.heap sort
7.exit
choose your option: 1

Enter the array size (the array is an array representation of a heap): 6
Enter the array elements:
8
11
70
51
2
56
the array (heap) is: 8 11 70 51 2 56
the Max heap is: 70 51 56 11 2 8
main menu..
1.create max heap
2.increase key
```



```
51
2
56
the array (heap) is: 8 11 70 51 2 56
the Max heap is: 70 51 56 11 2 8
main menu..
1.create max heap
2.increase key
3.insert in max heap
4.decrease key
5.delete from max heap
6.heap sort
7.exit
choose your option: 6

After heap sort, the sorted array is: 2 8 11 51 56 70
main menu..
1.create max heap
2.increase key
3.insert in max heap
4.decrease key
5.delete from max heap
6.heap sort
7.exit
choose your option: 7


--------------------------------
Process exited after 612.7 seconds with return value 0
Press any key to continue . . .
```