```
#include <iostream>

#include <iomanip>

#include <assert.h>

#include <string>


using namespace std;

class Heap

{

	private:

		int *heap;

		int size,last;

		reheapUp(int);

		void reheapDown(int,int);

	public:

		Heap(int);

		void buildHeap();

		void createArr(int *);

		void printHeap();

		int insertHeap(int);

		int selectK(int);

		void heapSort();

		int deleteHeap();

};

Heap::Heap(int s)

{

	size=s;

	last=8;
```

```
	heap=new int[size];
}
void Heap::buildHeap()
{
	int walker=1;
	while(walker<=last)
	{
		reheapUp(walker);
		walker=walker+1;
	}
}
Heap::reheapUp(int newNode)
{
	int parent,temp;
	parent=(newNode-1)/2;
	if(newNode!=0)
	{
		if(heap[newNode]>heap[parent])
		{
			temp=heap[newNode];
			heap[newNode]=heap[parent];
			heap[parent]=temp;
			reheapUp(parent);
		}
	}
}
void Heap::reheapDown(int root,int last)
```

```
{
        int lowkey=0;
        int temp=0,rightkey=0,leftkey=0,largechildkey=0,largechildindex=0;
        if((root*2)+1<=last)
        {
                leftkey=heap[(root*2)+1];
                if((root*2)+2<=last)
                {
                        rightkey=heap[(root*2)+2];
                }
                else
                {
                        rightkey=lowkey;
                }
                if(leftkey>rightkey)
                {
                        largechildkey=leftkey;
                        largechildindex=(root*2)+1;
                }
                else
                {
                        largechildkey=rightkey;
                        largechildindex=(root*2)+2;
                }
                if(heap[root]<heap[largechildindex])
                {
                        temp=heap[root];
```

```
                        heap[root]=heap[largechildindex];
                        heap[largechildindex]=temp;
                        reheapDown(largechildindex,last);
                }
        }
}
int Heap::insertHeap(int data)
{
        //size++;
        if(last==size-1)
        {
                cout<<"\nHeap is Full"<<endl;
        }
        else
        {
                last=last+1;
                heap[last]=data;
                reheapUp(last);
        }

}
int Heap::deleteHeap()
{

        if(last==-1)
        {
                cout<<"\nHeap is Empty"<<endl;
```

```
                return 0;
        }
        else
        {

        heap[0]=heap[last];
        last=last-1;
        reheapDown(0,last);
}
}
void Heap::createArr(int *a)
{
        for(int i=0;i<size;i++)
        {
                heap[i]=a[i];
        }
}
void Heap::heapSort()
{
        int sorted,holdData;
        buildHeap();
        sorted=last;
        while(sorted>0)
        {
                holdData=heap[0];
                heap[0]=heap[sorted];
                heap[sorted]=holdData;
```

```
                sorted--;
                reheapDown(0,sorted);
        }
}
int Heap::selectK(int k)
{
        int holdout=0,i=0,temp=0,heapsize;
        if(k>last)
        {
                return 0;
        }
        else
        {
        i=1;
        heapsize=last;
        while(i<k)
        {
                temp=heap[0];
                deleteHeap();
                heap[last+1]=temp;
                i=i+1;
        }
        holdout=heap[0];
        while(last<=heapsize)
        {
                last=last+1;
                reheapUp(last);
```

```
        }
        last--;
        return holdout;
}
}
void Heap::printHeap()
{
        for(int i=0;i<=last;i++)
        {
                cout<<heap[i]<<" ";
        }
}
int main()
{
        /*int n=0;
        cout<<"\n\n\tEnter size of Heap Array : ";
        cin>>n;*/
        Heap h1(12);
        int ar[]={5,10,35,25,10,20,40,30,50};
        h1.createArr(ar);

        h1.buildHeap();
        cout<<"\n\nElements of the heap:"<<endl;
        h1.printHeap();

        cout<<"\n\nInsert 33,22,8 into heap"<<endl;
        h1.insertHeap(33);
```

```
h1.insertHeap(22);
h1.insertHeap(8);
h1.printHeap();

cout<<"\n\nAfter deleting heap:"<<endl;
h1.deleteHeap();
h1.deleteHeap();
h1.deleteHeap();
h1.printHeap();

cout<<"\n\nInsert 60,70,80 into heap"<<endl;
h1.insertHeap(60);
h1.insertHeap(70);
h1.insertHeap(80);
h1.printHeap();

int k=0,s=0;
cout<<"\n\nEnter which highest element to view: ";
cin>>k;
s=h1.selectK(k);
cout<<"\nThe "<<k<<" highest element is "<<s<<endl;
h1.printHeap();

h1.heapSort();
cout<<"\nElements after sorting: "<<endl;
h1.printHeap();
```

```
        return 0;

}
```

C:\Users\Vaidhai\Desktop\heapassignment.exe

```
Elements of the heap:
50 40 35 30 10 10 20 5 25

Insert 33,22,8 into heap
50 40 35 30 33 10 20 5 25 10 22 8

After deleting heap:
33 30 20 25 22 10 8 5 10

Insert 60,70,80 into heap
80 60 70 25 33 20 8 5 10 22 30 10

Enter which highest element to view: 3

The 3 highest element is 60
80 60 70 25 33 20 8 5 10 22 30 10
Elements after sorting:
5 8 10 10 20 22 25 30 33 60 70 80
--------------------------------
Process exited after 2.724 seconds with return value 0
Press any key to continue . . .
```