데이터 구조 과제

학번: 12170584

이름: 이진호

**Heap**

**문제정의:**

1. 서로 다른 정수로 구성되는 Max Heap을 처리하는 프로그램을 만든다.
2. stdin에서 command를 입력받으며, 각 command 수행 후 적절한 답을 stdout에 출력 한다.

**동작 설명**

I *x*: Max Heap에 *x*를 삽입[push] (Insert x into Max Heap)

D: 삭제[pop] (Delete root from Max Heap)

T: 가장 큰 값 출력[top] (Output max value, donot remove it)

D *i*: *i*번째 큰 값 삭제 (Delete ith max value from heap)

T *i*: *i*번째 큰 값 출력 (Output ith max value from heap,

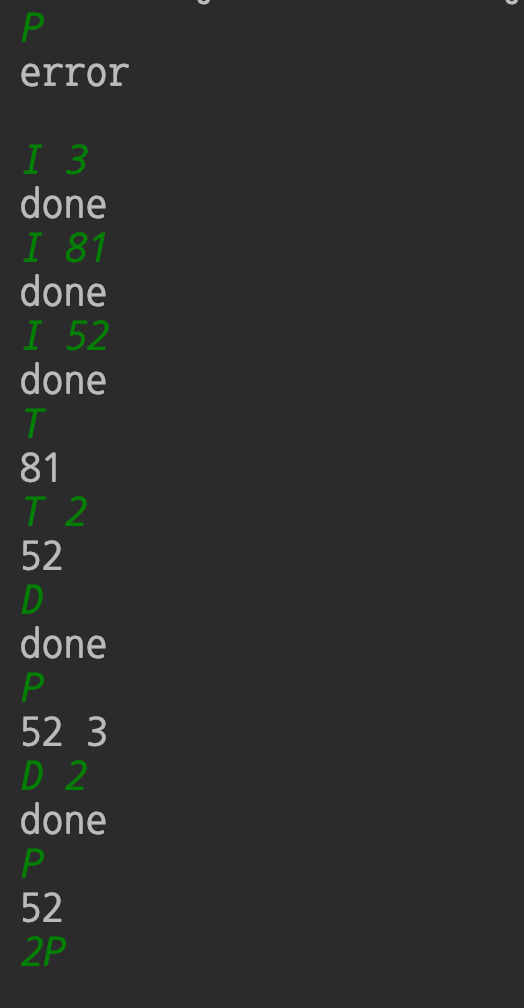
donot remove it)

P: Max Heap 내용 출력 (root부터 노드 순서 차례로)

**코드 설명:**

#include <iostream>  
#define MAX\_SIZE 100  
using namespace std;  
  
typedef int Compare(int a, int b);  
int arr[100] = {0,};  
  
//struct heap  
struct Heap{  
 Compare\* comp;  
 int num;  
 int heapArr[MAX\_SIZE];  
};  
//Maxheap initialize  
void heapInit(Heap\* h,Compare pc){  
 h->num=0;  
 h->comp=pc;  
}  
//checking Empty heap  
int isEmpty(Heap\* h){  
 if(h->num == 0){  
 return true;  
 }  
 else return false;  
}  
//get parent heap  
int getParent(int index){  
 return index/2;  
}  
//get leftchild heap  
int getLchild(int index){  
 return index\*2;  
}  
//get rightchild heap  
int getRchild(int index){  
 return index\*2+1;  
}  
//get bigger child heap  
int getHighChild(Heap\* h,int index){  
 if(getLchild(index)>h->num){  
 return 0;  
 }  
 else if(getLchild(index) == h->num){  
 return getLchild(index);  
 }  
 else{  
  
 if(h->comp(h->heapArr[getLchild(index)],h->heapArr[getRchild(index)])<0){  
 return getRchild(index);  
 }  
 else{  
 return getLchild(index);  
 }  
  
 }  
}  
  
//Maxheap Insert data  
void hInsert(Heap\* h,int data){  
 int index = h->num+1;  
  
 while(index!=1){  
 if(h->comp(data,h->heapArr[getParent(index)])>0){  
 h->heapArr[index] = h->heapArr[getParent(index)];  
 index = getParent(index);  
 }  
 else break;  
 }  
  
 h->heapArr[index] = data;  
 h->num+=1;  
}  
//Maxheap delete  
int hDelete(Heap\* h){  
 int data = h->heapArr[1];  
 int lastData = h->heapArr[h->num];  
  
 int parentIndex = 1;  
 int childIndex;  
  
  
  
 while(childIndex = getHighChild(h,parentIndex)){  
 if(h->comp(lastData,h->heapArr[childIndex]) >=0){  
 break;  
 }  
  
 h->heapArr[parentIndex] = h->heapArr[childIndex];  
 parentIndex = childIndex;  
 }  
  
 h->heapArr[parentIndex] = lastData;  
 h->num-=1;  
  
 return data;  
}  
  
//get Maximum data in Maxheap  
int hMax(Heap\* h){  
  
 return h->heapArr[1];  
}  
  
//Maxheap print  
void hPrint(Heap\* h){  
 if(isEmpty(h)){  
 cout << "error" << endl;  
 }  
 for(int i=1;i<= h->num;i++){  
 cout << h->heapArr[i] << " ";  
 }  
 cout << endl;  
}  
  
//sorting a heap  
void hSort(int arr[],int n,Compare pc){  
 Heap heap;  
  
 heapInit(&heap,pc);  
  
 for(int i=1;i<=n;i++){  
 hInsert(&heap,arr[i]);  
 }  
 for(int i=1;i<=n;i++){  
 arr[i] = hDelete(&heap);  
 }  
  
}  
  
//delete heap userlocation  
int hdeleteN(Heap\* h,int loc1){  
  
  
 int loc = 0;  
 for(int i=1;i<=h->num;i++){  
 if(h->heapArr[i] == arr[loc1]){  
 loc = i; break;  
 }  
 }  
 int data = h->heapArr[loc];  
 int lastData = h->heapArr[h->num];  
 int parentIndex = loc;  
 int childIndex;  
  
  
  
 while(childIndex = getHighChild(h,parentIndex)){  
 if(h->comp(lastData,h->heapArr[childIndex]) >=0){  
 break;  
 }  
  
 h->heapArr[parentIndex] = h->heapArr[childIndex];  
 parentIndex = childIndex;  
 }  
  
 h->heapArr[parentIndex] = lastData;  
 h->num-=1;  
  
 return data;  
}  
  
//compare function  
int compare(int a,int b){  
  
 return a-b;  
  
}  
  
  
//main  
int main(){  
 bool run = true;  
 Heap heap;  
 heapInit(&heap,compare);  
  
 while(run){  
 string in;  
 string in2 = "";  
  
 getline(cin,in);  
  
 char input1 = in[0];  
 int input2 = -1;  
  
 //exception  
 if(in[0] < 65 || in[0] > 90){  
 break;  
 }  
  
 bool err = false;  
 //parsing char and int  
 if(in[2]!=NULL){  
  
 for(int i=2;i<in.length();i++){  
 in2+=in[i];  
 if(in[i] < 48 || in[i] > 57){  
 err = true;  
 }  
 }  
 //exception  
 if(err){  
 break;  
 }  
  
 input2 = stoi(in2);  
 }  
  
  
  
 //sorting heap array  
 if(!isEmpty(&heap)){  
 for(int i=1;i<=heap.num;i++){  
 arr[i] = heap.heapArr[i];  
 }  
  
 hSort(arr,heap.num,compare);  
  
 }  
  
 //calling function  
 switch(input1){  
  
 case 'I':  
 hInsert(&heap,input2);  
 cout << "done" << endl;  
 break;  
 case 'D':  
 if(isEmpty(&heap)){  
 cout << "error" << endl;  
 }  
  
 else if(input2 == -1){  
 hDelete(&heap);  
 cout << "done" << endl;  
 }  
  
 else{  
 hdeleteN(&heap,input2);  
 cout << "done" << endl;  
  
 }  
 break;  
  
 case 'T':  
 if(isEmpty(&heap)){  
 cout << "error" << endl;  
 }  
  
 else if(input2 == -1){  
 cout << hMax(&heap) << endl;  
 }  
  
 else{  
 cout << arr[input2] << endl;  
 }  
  
 break;  
  
  
 case 'P':  
 hPrint(&heap);  
 break;  
  
 //exception  
 default:  
 run=false;  
 break;  
 }  
 }  
}

**출력 결과**



**결론- 힙을 이용하여 삽입, 삭제, 출력을 할 수 있었다.**