Algorithm HW #1

컴파일 환경: g++ (Ubuntu 7.4.0-1ubuntu1~18.04.1) 7.4.0

1-1. Heap sort

※ 파일 이름: 1-1HeapSort.cpp, 1-1HeapSort.out

1) 헤더 파일 목록

- <iostream>

2) Functions

function	설명
Parent(int x): int	index가 x인 node의 부모 index를 찾아서 return
LChild(int x): int	index가 x인 node의 왼쪽 자식 index를 찾아서 return
RChild(int x): int	index가 x인 node의 오른쪽 자식 index를 찾아서 return
swap(int* a, int* b): void	a와 b의 값을 서로 바꾼다.
Max_Heapify(int nodenum, int size, int* array): void	index가 nodenum인 node가 Heap의 규칙에 알맞도록 array를 정렬한다. 이 때 array의 크기는 size로 받는다.

3) main 함수 동작 과정

① stdin으로 n개의 자연수를 입력 받아 array에 저장한다.

② array의 가장 마지막 원소의 부모부터 루트 노드까지 Heap의 규칙에 알맞도록 정렬한다.

③ Heap의 가장 첫 노드를 꺼내어 출력하고 Heap을 다시 정렬한다.

④ 모든 원소를 출력할 때까지 과정 ③을 반복한다.

4) 실행 화면 캡처

- 주어진 예시

```
sp2019@sp2019-VirtualBox:~/algorithm$ g++ -o 1-1HeapSort.out 1-1HeapSort.cpp
sp2019@sp2019-VirtualBox:~/algorithm$ ./1-1HeapSort.out
Input numbers to sort:
9 45 871 23 13 13 88 46 12 51 99 -1

Result:
871 99 88 51 46 45 23 13 13 12 9
```

- n=99일 때

```
sp2019@sp2019-VirtualBox:~/algorithm$ ./1-1HeapSort.out
Input numbers to sort:
225 129 82 957 785 281 623 871 519 539 939 125 769 226 58 147 210 124 82 339 869
 290 446 226 103 341 473 474 683 199 426 177 794 78 781 695 369 537 22 40 145 53
5 275 223 203 939 300 894 25 287 847 981 176 329 134 70 44 423 219 510 45 615 49
7 794 334 993 564 908 468 884 656 294 224 289 639 378 598 976 505 806 918 955 21
8 96 335 574 652 97 992 862 883 235 513 344 843 435 755 581 719 -1

Result:
993 992 981 976 957 955 939 939 918 908 894 884 883 871 869 862 847 843 806 794
794 785 781 769 755 719 695 683 656 652 639 623 615 598 581 574 564 539 537 535
519 513 510 505 497 474 473 468 446 435 426 423 378 369 344 341 339 335 334 329
300 294 290 289 287 281 275 235 226 226 225 224 223 219 218 210 203 199 177 176
147 145 134 129 125 124 103 97 96 82 82 78 70 58 45 44 40 25 22
sp2019@sp2019-VirtualBox:~/algorithm$ diff -s output.txt test1.txt
Files output.txt and test1.txt are identical
```

(output.txt: Result로 만든 txt 파일, test1.txt: 엑셀을 통해 입력을 정렬한 txt파일, 둘의 내용이 동일한지 비교하여 정렬이 잘 되었는지 확인)

- n=1일 때

```
sp2019@sp2019-VirtualBox:~/algorithm$ ./1-1HeapSort.out
Input numbers to sort:
2 -1

Result:
2
```

1-2. 연습문제 6-5.9

※ 파일 이름: 1-2Practice.cpp, 1-2Practice.out

1) 헤더 파일 목록

- <iostream>, <fstream>, <sstream>

2) Struct

- struct nodestruct{

```
int num;          //보관할 숫자
int xaddress;     //해당 숫자가 있던 array의 x좌표
int yaddress;     //해당 숫자가 있던 array의 y좌표
```

};

- struct HeapStruct{

```
int size;         //Heap에 들어있는 element 수
node *list;       //Heap
```

};

- node: struct nodestruct의 포인터
- MinHeap: struct HeapStruct의 포인터

3) Functions

function	설명
Parent(int x): int	index가 x인 node의 부모 index를 찾아서 return
LChild(int x): int	index가 x인 node의 왼쪽 자식 index를 찾아서 return
RChild(int x): int	index가 x인 node의 오른쪽 자식 index를 찾아서 return
swap(int* a, int* b): void	a와 b의 값을 서로 바꾼다.
BuildHeap(int size, MinHeap H): MinHeap	Heap을 만들고 Heap 안에 size만큼의 빈 node 포인터를 만든다. Heap의 size(들어 있는 element 수)는 0으로 초기화한다. 새로 만든 Heap을 return한다.
Min_Heapify(int nodenum, MinHeap H): void	index가 nodenum인 node가 Heap의 규칙에 알맞도록 array를 정렬한다.
Insert(node X, MinHeap H): void	node X를 MinHeap H에 입력하고 Heap의 규칙에 알맞도록 정렬한다. H의 size를 하나 키운다.
Delete_Min(MinHeap H): node	MinHeap H에서 제일 작은 node인 root 노드를 빼낸 뒤 Heap의 규칙에 알맞도록 정렬한다. H의 size를 하나 줄인다. 빼낸 노드를 return한다.

4) main 함수 동작 과정

① input1-2.txt 파일을 통해 k개의 배열을 입력 받는다.

② 각 배열의 첫 번째 원소를 Heap에 넣고 정렬한다.

③ Heap의 가장 첫 노드를 꺼내어 값을 출력하고 Heap을 다시 정렬한다.

④ 꺼낸 노드가 있던 배열에서 다음 노드를 Heap에 넣고 정렬한다.

⑤ 모든 배열의 원소를 Heap에 넣을 때까지 ③, ④ 과정을 반복한다.

⑥ Heap이 빌 때까지 Heap의 가장 첫 노드를 꺼내어 값을 출력하고 Heap을 다시 정렬한다.

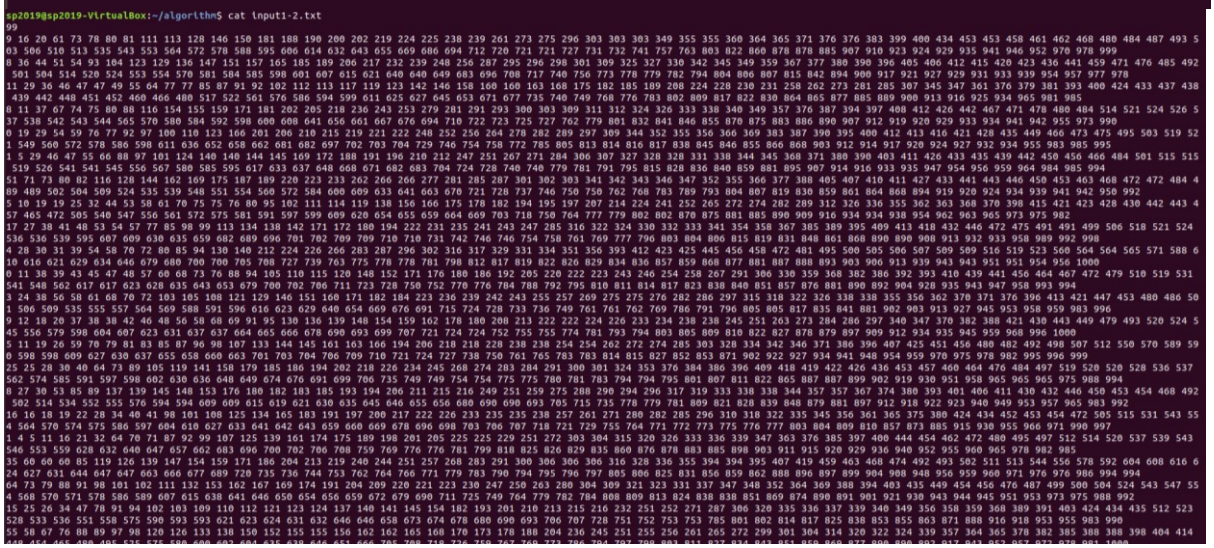
5) 실행 화면 캡처

- 주어진 예시

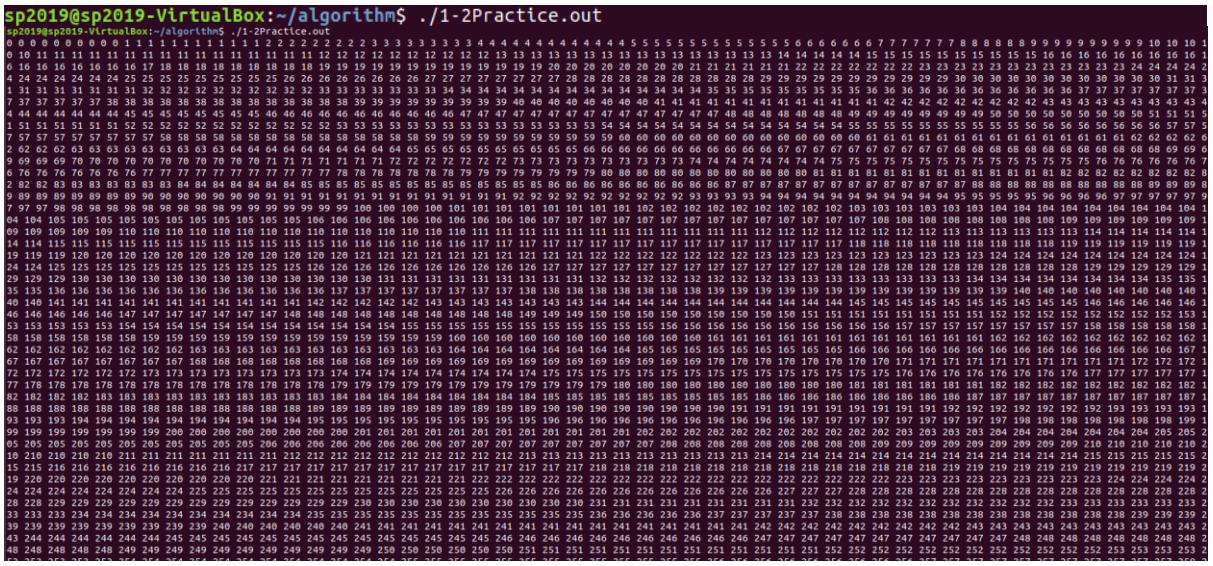
```
sp2019@sp2019-VirtualBox:~/algorithm$ g++ -o 1-2Practice.out 1-2Practice.cpp
sp2019@sp2019-VirtualBox:~/algorithm$ cat input1-2.txt
4
0 2 3 5 5 10 30
0 5 6 7 10 13 15
1 7 11
17 18 21 23
sp2019@sp2019-VirtualBox:~/algorithm$ ./1-2Practice.out
0 0 1 2 3 5 5 5 6 7 7 10 10 11 13 15 17 18 21 23 30
```

- 각 배열의 원소 개수가 99개, k=99일 때

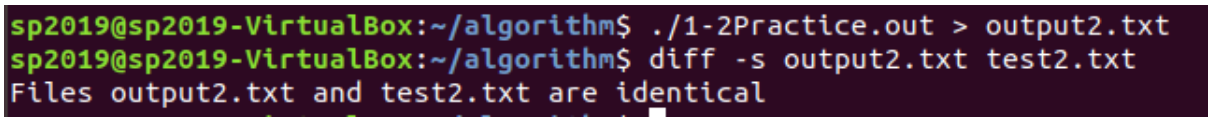
```
sp2019@sp2019-VirtualBox:~/algorithm$ cat input1-2.txt
99
```



Input1-2.txt 로 들어간 파일(너무 길어서 일부 생략)



1-2Practice.out 실행 화면(너무 길어서 일부 생략)



1-2Practice.out의 표준출력을 output2.txt에 저장 후 test2.txt와 비교한 화면

(output2.txt: Result로 만든 txt 파일, test2.txt: 엑셀을 통해 입력을 정렬한 txt파일, 둘의 내용이 동일한지 비교하여 정렬이 잘 되었는지 확인)

- 배열의 원소의 개수가 1개, k=1일 때