-처음 코드를 실행하면 데이터 값을 한개씩 입력, 여러개 동시에 입력을 선택할 수 있게 하였다.

-한개씩 입력에는 insert함수를, 여러개 동시에 입력에는 build heap 함수를 이용해 기능을 구현하였고 그외 extract min, delete heap, heapify, display등 기능은 모두 동일하다.

이때 기능 구현을 원활하게 하기 위해 shiftup, shiftdown함수를 따로 만들어서 사용했다.

-크기가 100인 배열을 만들어서 코드를 짰다.

>shiftup

index값을 받아 parent node값 보다 해당 index node의 값이 더 작을 경우 parent node 와 자리를 바꾸는 코드이다.

heapify, insert, build heap에서 이 기능을 사용했다.

>shiftdown

index, size의 값을 받아 내려가면서 값을 비교하여 child값이 더 작을 경우 위치를 바꾸는 코드이다.

작은 것의 index를 표현하기 위해 정수형 변수 small을 사용했고, 초기에는 index로는 선택 불가한 값 -1로 초기화해줬다.

두 child node 중 더 작은 값과 위치를 바꾼다.

extract min에서 이 기능을 사용했다.

>insert

배열의 크기를 넘어가는 값을 입력할 때는 “full”메세지를 반환하게 했다.

위에서 설명한 shiftup함수를 활용하여 매번 입력할때마다 heap성질을 만족하게 구현했다.

>display

현재 heap이 잘 구현되고 있는지 확인하기 위해 만들었다.

>exmin

extract min, 최솟값을 없애는 코드이다.

arr[0]의 값을 없애고 일단 마지막 노드의 값을 가져온다. 이때 마지막 노드를 구별하는 것은 size 변수를 활용했다.

마지막 노드의 값을 맨 위(root node)로 가져온 뒤 shiftdown 기능을 활용해 heap 성질을 만족시켰다.

heap구조에 데이터가 없는데 exmin을 수행하려 하면 “heap is empty”문구를 반환하게 만들었다.

>deleteheap

기존에 만들어진 heap을 삭제한다. heap에 저장된 데이터의 크기를 의미하는 size 값을 -1로 초기화 한 후에 다시 제 기능을 수행할 수 있도록 0으로 다시 맞춰줬다.

>buildheap

입력받고자 하는 데이터의 개수를 입력한다.

그 개수에 맞춰 데이터를 입력하면 heap구조를 만족하게 데이터를 가공한다.

이때 shiftup함수를 활용하였다.

>heapify

shiftup함수를 이용해 데이터가 heap 성질을 만족할 수 있도록 만들어준다.

다음은 컴파일 후 콘솔화면에서 실제로 구현해본 결과이다.

 