```
// p 부모, lastIdx 마지막 인덱스
int heapfy(int p,int lastIdx)
{
	// 인덱스 0부터
	// 오른쪽 자식 (n+1)*2
	// 왼쪽 자식 (n+1)*2 -1 왼쪽 자식
	// 부모는 (자식-1)/2

	//printf("heapify p:%dWn",p);
	int leftCIdx = (p+1)*2-1;
	int rightCIdx = (p+1)*2;

	//왼쪽 오른쪽 노드 모두 없는 경우
	if( (leftCIdx > (lastIdx) ) || (rightCIdx > (lastIdx) ) )
		return 1;

	int smallerIdx=0;
	//왼쪽만 있는 경우
	if( (leftCIdx <= lastIdx) && (rightCIdx > (lastIdx) ) && ( g_student[leftCIdx].score <
g_student[p].score )  )
	{
		smallerIdx = leftCIdx;
	}
	//양쪽이 있는 경우 1
	else if( (leftCIdx <= lastIdx) && (rightCIdx <= lastIdx) && ( g_student[leftCIdx].score
< g_student[rightCIdx].score ) )
	{
		smallerIdx = leftCIdx;
	}
	//양쪽이 있는 경우 2
	else if( (leftCIdx <= lastIdx) && (rightCIdx <= lastIdx) && ( g_student[leftCIdx].score
>= g_student[rightCIdx].score ) )
	{
		smallerIdx = rightCIdx;
	}

	if( g_student[smallerIdx].score != g_student[p].score)    //궂이 같으면 스왑안해도
	{
		STUDENT temp =  g_student[smallerIdx];
		g_student[smallerIdx] = g_student[p];
		g_student[p] = temp;
		heapfy(smallerIdx,lastIdx);
	}

	return 0;
}


main()
{
....

	for(i=0 ; i<N ; i++)
	{
		g_student[i].id = students[i].id;
		g_student[i].score = students[i].score;
	}

	//1.Build heap : heap 은 중간노드 까지만 자식이 존재한다 so,  0번노드부터 중간노드 까지
만 순회하면서 힙형태를 만든다.
	i = g_varN/2 - 1;
	for(i ; i>=0 ; i--)
	{
		heapfy(i,g_varN-1);
	}

	// 2.Heap sort : n번 반복하면서 heapify
	i = N-1;
```

```
	for(i ; i>=0 ; i--)
	{
		//root 원소를 가지고 링크드 리스트 만들기
		insertNode(g_student[0]);
		//맨뒤 노드를 루트노드 자리로 이동하고 전체 원소 개수를 하나 줄인다.
		g_student[0] = g_student[i];

		if( i >= 1)
			heapfy(0,i-1);
	}
}
```