Lab 10: Heaps

General Instructions:

1- Create a C project in eclipse called Lab10

2- Download Lab10.zip files and extract them to a folder in your machine. `lab10_output.txt`

3- Enter your credentials on top of the file `heap.c` file.

4- Your output after using your copy of "Lab10_test.c" file should produce output similar to that of "Lab10_output.txt".

5- You need to submit only "heap.c" file.

Task 1: Leaf Nodes in a Heap

Implement the function:

```
int get_leaves(Heap *h, int  *leaves_array);
```

The above function receives a heap, and search it for all nodes which are leaves. A leaf node is a node that does not have children (left or right).

The function stores the indices of the leaf nodes in the leaves_array, and returns the number of leaves in the heap.

Task 2: Max item in Heap

Implement the function:

```
Data* find_max_heap(Heap *h);
```

The function returns a copy of the maximum data item stored in the heap. If the heap is empty, the function prints an error message and returns NULL;

Task 3: Min item in Heap

Implement the function:

```
Data* find_min_heap(Heap *h);
```

The function returns a copy of the minimum data item stored in the heap. If the heap is empty, the function prints an error message and returns NULL;

Unlike find_max_heap, finding the minimum is not straight forward. Of course, you can do a linear_search in the heap, but that is a lazy programmer solution. We want something better. Think about a solution that does better than linear_search.

Task 4: Enhanced Heapify_up

Re-implement the function

```
void heapify_up(Heap *h, int i);
```

The solution provided in R11 produces the correct results, but is not the most efficient. The function keeps heapifying from the leaf node to the root.

There are scenarios where the heapifying should stop in between. Modify the function implementation to accommodate such scenarios.

Task 5: Ascending Heap Sort

Implement the function

```
void heap_sort2(Data *array, const int size);
```

The function sorts the data items using heap sort but in ascending order.