Task 1: Advanced Data Structures

Here is your task

Your task is to implement a novel data structure - your project lead is calling it a power of two max heap. The rest of your team is doing their best to come up with a better name. The requirements of the data structure are as follows:

- The heap must satisfy the heap property.
- Every parent node in the heap must have 2^x children.
- The value of x must be a parameter of the heap's constructor.
- The heap must implement an insert method.
- The heap must implement a pop max method.
- The heap must be implemented in Java.
- The heap must be performant.
- You must use a more descriptive variable name than x in your implementation.

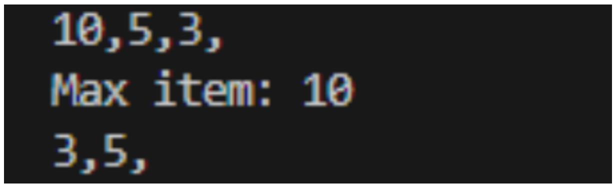
Think carefully about how you implement each method and manage the underlying data. Performance is critical, so keep cycles and memory usage to a minimum. Be sure to test your heap with very small and very large values of x. As always, keep a weather eye out for sneaky edge cases.

Solution

```
import java.util.Arrays; import
java.util.NoSuchElementException;
 public class PowerHeap {
private double x;
private int size;
private int[] heapArray;

    // Constructor     public
PowerHeap(double x, int capacity) {
this.size = 0;         heapArray = new
int[capacity + 1];         this.x = x;
        Arrays.fill(heapArray, -1);
    }      private int parent(int i) {
return (int) ((i - 1) / Math.pow(2, x));
    }
```

```
     public boolean isFull() {
return size == heapArray.length;
    }      public void insert(int value) {         if (isFull()) {
throw new NoSuchElementException("Heap is full, no space to insert new
element.");         } else {             heapArray[size++] = value;
heapifyUp(size - 1);
        }
    }      private void heapifyUp(int i) {
int tmp = heapArray[i];         while (i > 0 && tmp
> heapArray[parent(i)]) {             heapArray[i] =
heapArray[parent(i)];             i = parent(i);
        }
heapArray[i] = tmp;
    }      public int popMax() {
int maxItem = heapArray[0];
heapArray[0] = heapArray[size - 1];
heapArray[size - 1] = -1;         size--;
         int i = 0;
while (i < size - 1) {
heapifyUp(i);             i++;
        }
         return
maxItem;
    }      public void
print() {
        for (int i = 0; i < size; i++) {
            System.out.print(heapArray[i]);
            System.out.print(',');
        }
        System.out.println();
    }      public static void main(String[]
args) {          double x = 2; // Example
value for x          int capacity = 10; //
Example capacity

        PowerHeap heap = new PowerHeap(x, capacity);
heap.insert(5);          heap.insert(10);
heap.insert(3);

heap.print();

        int maxItem = heap.popMax();
        System.out.println("Max item: " + maxItem);

heap.print();
    }
}
```

Output



```
10,5,3,
Max item: 10
3,5,
```