Question 1:

For a d-ary heap the delMax( ) function has a time complexity of O(log *N*) * *d* in the worst case. The ***d*** is the number of children allowed by each parent, the *N* is the total number of elements in the array. In binary heap sort, where there are 2 children, the time complexity for worst case is O(log *N*) which implies that for a heap sort with ***d*** number of children there would be a coefficient added to the time complexity for binary heap.

```
public int delMax() {
		int max = 0;
		int maxKey = 0;
		int last = heapSize - 1;
		for(int i = 0; i < last; i++) {
			if(a[i] > max) {
				max = a[i];
				maxKey = i;
			}
		}
		exch(a, last, maxKey);
		heapSize--;
		sink(maxKey);
		a[last] = 0;
		return max;
	}

private void sink(int k) {

		int childSet = (k * child) + 1;
		if(childSet < heapSize) {
			int max = childSet;
			int counter = childSet + child;
			if(counter > heapSize) {
				counter = heapSize - 1;
			}
			for(int i = childSet; i < counter; i++) {
				if (a[i] > a[max]) {
					max = i;
				}
			}
			exch(a, max, k);
			sink(max);
		}

	}
```

So the running time of deleting a heap with *N* number of nodes is equal to the height of the heap. The height of a heap is equal to log(N + 1) approximately. So for delMax ( ) function for a d-ary heap it would be d * the height of the heap.

For a d-ary heap the sort function has a time complexity of O(log *N*) * *d* in the worst case as well where the *N* is the total number of elements in the array. For the binary heap the worst case is O(log *N*). Since the sort function also depends on the height of the heap the time complexity for the delMax( ) function and the daryHeapsort( ) is the same.

```
public int [] daryHeapsort() {
		int [] arraySort = new int [length];
		for(int i = 0; i < length; i++) {
			arraySort[i] = a[i];
		}
		int size = heapSize - 1;

		while(size > 2) {
			exch(arraySort, 0, size--);
			print(arraySort);
			sinkSorted(arraySort, 0, size);
		}

		for (int i = 1; i < child; i++) {
			if (arraySort[0] > arraySort[i]) {
				exch(arraySort, 0, i);
			}
		}

		return arraySort;
	}
```