1. What is the worst case asymptotic running time of isEmpty, size, insert, findMin, and deleteMin operations on all three of your heap implementations?

	isEmpty	size	insert	findMin	deleteMin
BinaryHeap	O(1)	O(1)	O(log(n))	O(1)	O(log(n))
ThreeHeap	O(1)	O(1)	O(log(n))	O(1)	O(log(n))
MyPQ	O(1)	O(1)	O(n)	O(1)	O(1)

2. Timing your code:

	N=10000	N=20000	N=50000	N=75000
Insert BinaryHeap	1 millisecond	3 milliseconds	4 milliseconds	1 millisecond
DeleteMin BinaryHeap	7 milliseconds	3 milliseconds	8 milliseconds	10 milliseconds
Total BinaryHeap	8 milliseconds	6 milliseconds	12 milliseconds	11 milliseconds
Insert ThreeHeap	2 milliseconds	3 milliseconds	3 milliseconds	2 milliseconds
DeleteMin ThreeHeap	7 milliseconds	7 milliseconds	6 milliseconds	11 milliseconds
Total ThreeHeap	9 milliseconds	10 milliseconds	9 milliseconds	13 milliseconds
Insert MyPQ	104 milliseconds	769 milliseconds	7966 milliseconds	20992 milliseconds
DeleteMin MyPQ	1 millisecond	1 millisecond	1 millisecond	1 millisecond
Total MyPQ	105 milliseconds	770 milliseconds	7967 milliseconds	20993 milliseconds

3. a. The asymptotic analysis was a good prediction of the actual run time. The insert and deleteMin methods for the binary and three heap ran about the same time and are much faster than the MyPQ class whose insert method takes much longer because it is O(n)
b. The prediction was pretty close, but in reality the binary and three heap ran faster because most of the time the worst condition is not present.

c. I would suggest using the binary heap because it runs about the same speed as the three heap and is much faster than the sorted linked list which I used for MyPQ. The reason to pick the binary heap over the three heap is because it is easier to implement. I would alternatively recommend MyPQ if the user did not care how long it took to insert the elements into the queue and wanted instant deletion.
4. To test the heaps I picked a random integer less than 100,000 and created an array of random doubles of that size. I then inserted all these doubles into the priority queue I was testing, sorted the double array, and deleted the minimum value of the priority queue until the queue was empty; all while making sure the value being deleted matched up with the value in the double array. Additionally, I did some basic logic testing to make sure the isEmpty, size, and findMin methods were working.
5. a. Binary: i*2, i*2+1
Three: i*3-1, i*3, i*3+1
Four: i*4-2, i*4-1, i*4, i*4+1
Five: i*5-3, i*5-2, i*5-1, i*5, i*5+1
b. d*i - (d - 2)