1.
- False. 2^{c * \sqrt(n)} = (2^c)^\sqrt(n). For example, let c = 2, then we have 4^\sqrt(n), which is not bounded by 2^\sqrt(n)

- True.

- True. big-O is an upper bound, but not a tight upper bound.

- False. The worst case running time is O(kn). (Note, this isn't the best worded question.)

- False. Let f = 2n. f is O(n), but 2^{2n} = 4^n is not O(2^n)

2.
a. \Theta(n \log n + k) = \Theta(n \log n) -- note k is always less than n so we can drop it.
b. \Theta(kn)
c. Unfortunately, selection is O(n^2) in the worst case. If you ignore this (technically, not correct), O(n + k \log k). Alternative answers would be to put in a heap: O(n + k \log n)

3.
call findShift(1, A.length, A)

```
findShift(s, e, A)
- mid = (e-s)/2
- if A[mid-1] > A[mid] -> return mid-1
- if A[s] > A[mid] -> return findShift(s, mid, A)
- else -> return findShift(mid, e, A)
```

4.
a. Master method
a = 3
b = 3
f(n) = log n

f(n) = O(n^0.9) so T(n) = \Theta(n)

b. Recursion tree method
T(n) = \sum_{i=1}^n n^d log n = n*n^d \log n = n^{d+1} log n