1. For each function $f(n)$ below, give an asymptotic upper bound using "big-Oh" notation. You should give the tightest bound possible (so giving $O(2^n)$ for every question is unlikely to result in many points). **You must choose your answer from the following list.** The list is not in any particular order and any item in the list could be the answer for 0, 1, or more than 1 question.

   $O(n), O(n^2), O(2^n), O(n^{1/2}), O(n^{1/4}), O(n^3), O(n^4), O(n^5), O(n^6), O(n^7), O(n^8), O(n^9),$
   $O(\log^3 n), O(n^2 \log n), O(\log^8 n), O(\log^4 n), O(n \log^3 n), O(\log^2 n), O(\log n), O(1), O(n^3 \log n),$
   $O(n^n), O(n \log \log n), O(n \log^2 n), O(\log \log n), O(n \log n)$

   (a) $f(n) = 100n^3 - 7n^3 + 14n^2$ _____

   (b) $f(n) = 100n^3 - 100n^3 + 7n^2$ _____

   (c) $f(n) = \log(7n^2)$ _____

   (d) $f(n) = 5 \log \log n + 4 \log^2(n)$ _____

   (e) $f(n) = .001n + 100 \cdot 2^n$ _____

   (f) $f(n) = n^3(1 + 6n + 2014n^2)$ _____

   (g) $f(n) = (\log n)(n + n^2)$ _____

Solution:

(a) $O(n^3)$
(b) $O(n^2)$
(c) $O(\log n)$
(d) $O(\log^2 N)$
(e) $O(2^n)$
(f) $O(n^5)$
(g) $O(n^2 \log n)$

2. Describe the worst case running time of the following code in "big-Oh" notation in terms of the variable `n`. You should give the tightest bound possible.

(a)
```
void f1(int n) {
  for(int i=0; i < n; i++) {
    for(int j=0; j < n; j++) {
      for(int k=0; k < n; k++) {
        for(int m=0; m < n; m++) {
          System.out.println("!");
  } } } }
}
```

(b)
```
void f2(int n) {
  for(int i=0; i < n; i++) {
    for(int j=0; j < 10; j++) {
      for(int k=0; k < n; k++) {
        for(int m=0; m < 10; m++) {
          System.out.println("!");
  } } } }
}
```

(c)
```
int f3(int n) {
  int sum = 73;
  for(int i=0; i < n; i++) {
    for(int j=i; j >= 5; j--) {
       sum--;
    }
  }
  return sum;
}
```

(d)
```
int f4(int n) {
  if (n < 10) {
    System.out.println("!");
    return n+3;
  } else {
    return f4(n-1) + f4(n-1);
  }
}
```

(e)
```
int f5(int n) {
  if (n < 10) {
    System.out.println("!");
    return n+3;
  } else {
    return f5(n-1) + 1;
  }
}
```

Solution:
(a) $O(n^4)$, (b) $O(n^2)$, (c) $O(n^2)$, (d) $O(2^n)$, (e) $O(n)$

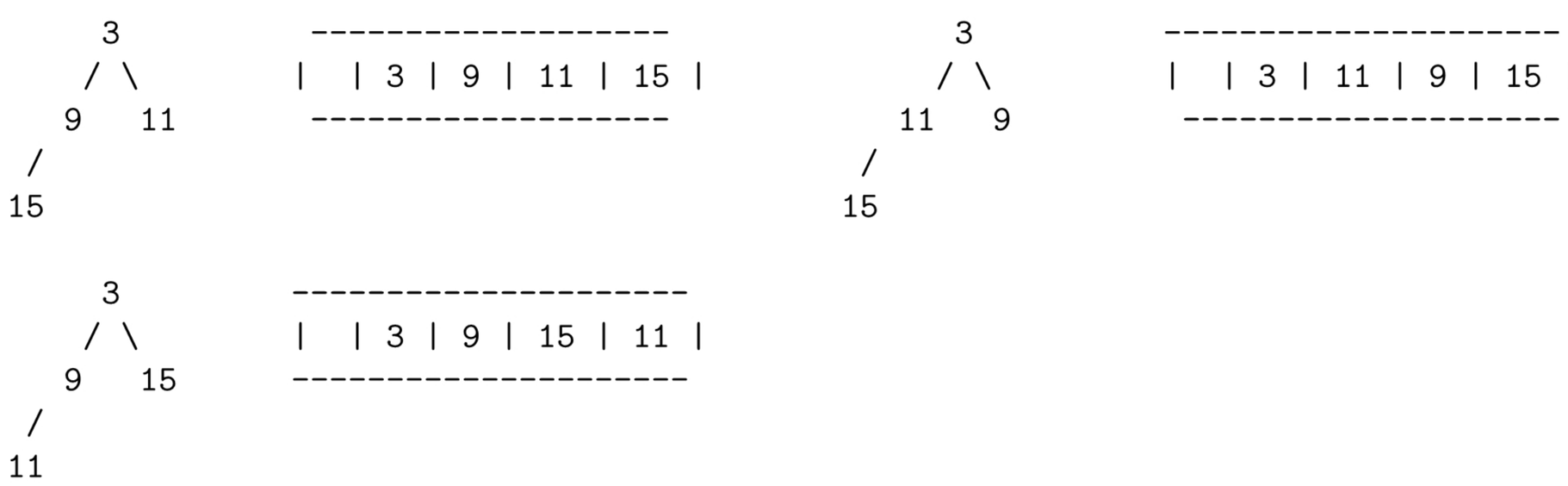
3. Suppose there is a binary min-heap with exactly 4 nodes, containing items with priorities 3, 9, 11, and 15.

   (a) Show **every** possible binary min-heap that could match this description. For each, draw the appropriate tree *and* the array representation. (You can show just the priorities, not the corresponding items.)

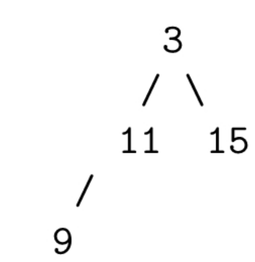
   (b) For **one** of your answers to part (a), show what happens with 4 `deleteMin` operations. Clearly indicate which heap you are starting with and show the heap after *each* `deleteMin`. You can just draw the tree (not the array) after each step.

Solution:

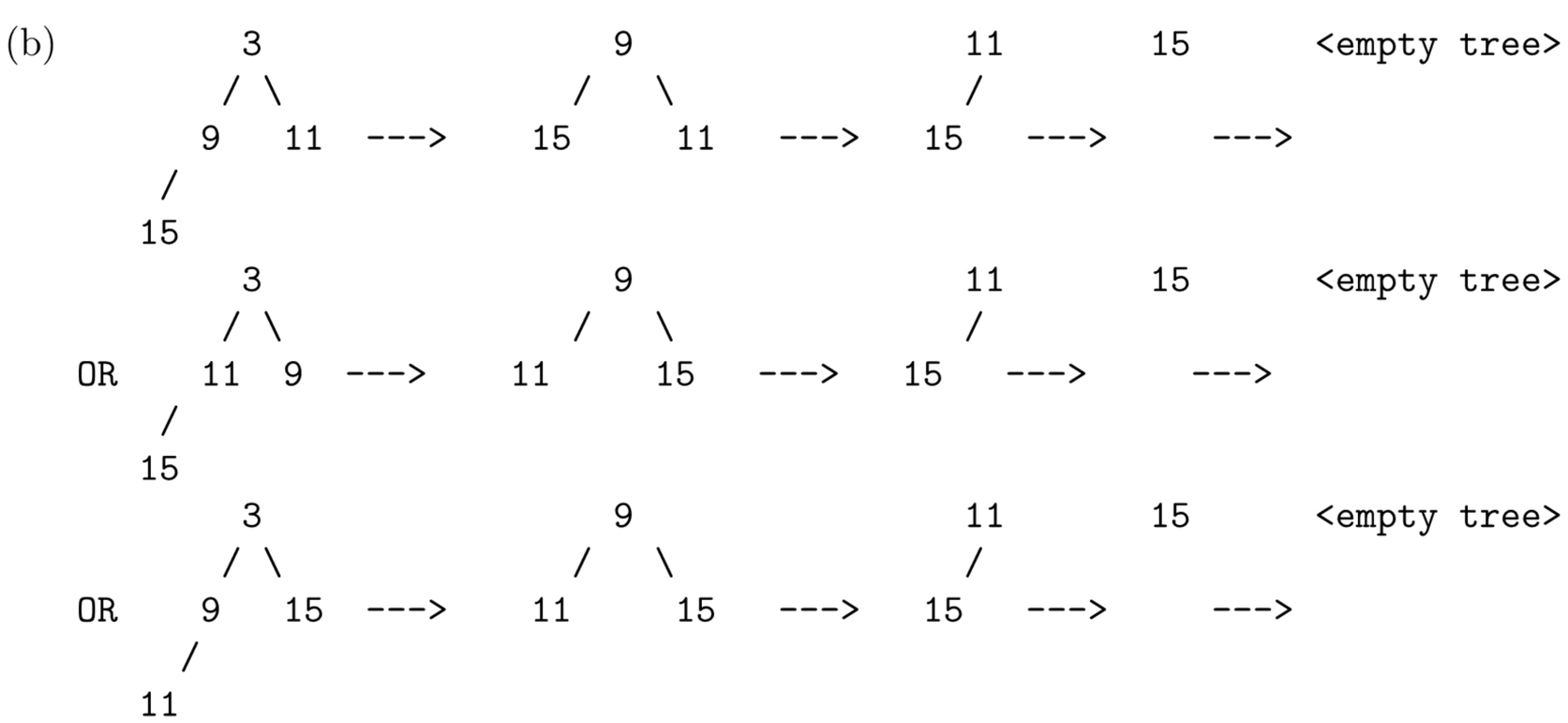
(a) Three possibilities:



Explanation: All 4-node heaps must have the shape of the 3 heaps in part (a). The minimum node, 3, must be at the root. The maximum node, 15, must be at a leaf. The only heap meeting these rules and not listed above does not satisfy the heap property because 11 is above 9:



(b)



4. **Big-Oh and Run Time Analysis:** Describe the worst case running time of the following pseudocode functions in Big-Oh notation in terms of the variable `n`. ***Showing your work is not required***. You MUST choose your answer from the following (not given in any particular order), each of which could be re-used (could be the answer for more than one of I. – IV.):

$O(n^2)$, $O(n^3 \log n)$, $O(n \log n)$, $O(n)$, $O(n^2 \log n)$, $O(n^5)$, $O(2^n)$, $O(n^3)$, $O(\log n)$, $O(1)$, $O(n^4)$, $O(n^n)$, $O(n^6)$, $O(n^8)$, $O(n^7)$

I.
```
void smiley (int n, int sum) {
    for (int i = 0; i < n * 100; ++i) {
       for (int j = n; j > 0; j--)
          sum++;
       for (int k = 0; k < i; ++k)
          sum++;
       }
    }
```

Runtime: $O(n^2)$

II.
```
int sunny (int n, int m, int sum) {
    if (n < 10)
      return n;
    else {
      for (int i = 0; i < n; ++i)
        sum++;
      return sunny (n - 2, m, sum);
    }
}
```

$O(n^2)$

III.
```
void funny (int n, int x, int sum) {
    for (int k = 0; k < n; ++k) {
       if (x < 10) {
          for (int i = 0; i < n; ++i)
             sum++;
       }
       for (int j = n; j > 0; j--)
          sum++;
       }
    }
```

$O(n^2)$

IV.
```
void happy (int n, int sum) {
    int j = n;
    while (j > 2) {
       sum++;
       j = j / 2;
    }
}
```

$O(\log n)$

5. **Big-Oh and Run Time Analysis:** Describe the worst case running time of the following pseudocode functions in Big-Oh notation in terms of the variable `n`. ***Showing your work is not required*** (although showing work ***<u>may</u>*** allow some partial credit in the case your answer is wrong – don't spend a lot of time showing your work.). You MUST choose your answer from the following (not given in any particular order), each of which could be re-used (could be the answer for more than one of I. – IV.):

$O(n^2)$, $O(n^3 \log n)$, $O(n \log n)$, $O(n)$, $O(n^2 \log n)$, $O(n^5)$, $O(2^n)$, $O(n^3)$, $O(\log n)$, $O(1)$, $O(n^4)$, $O(n^n)$, $O(n^6)$, $O(n^8)$, $O(n^7)$

I.
```
int happy (int n, int m) {
    if (n < 10) return n;
    else if (n < 100)
      return happy (n - 2, m);
    else
      return happy (n/2, m);
}
```

Runtime: $O(\log n)$

II.
```
void sunny (int n) {
    j = 0;
    while (j < n) {
       for (int i = 0; i < n; ++i) {
           System.out.println("i = " + i);
           for (int k = 0; k < i; ++k)
              System.out.println("k = " + k);
       }
       j = j + 1;
    }
}
```

$O(n^3)$

III.
```
void smiley (int n) {
    for (int i = 0; i < n * n; ++i) {
       for (int k = 0; k < i; ++k)
           System.out.println("k = " + k);
       for (int j = n; j > 0; j--)
           System.out.println("j = " + j);
    }
}
```

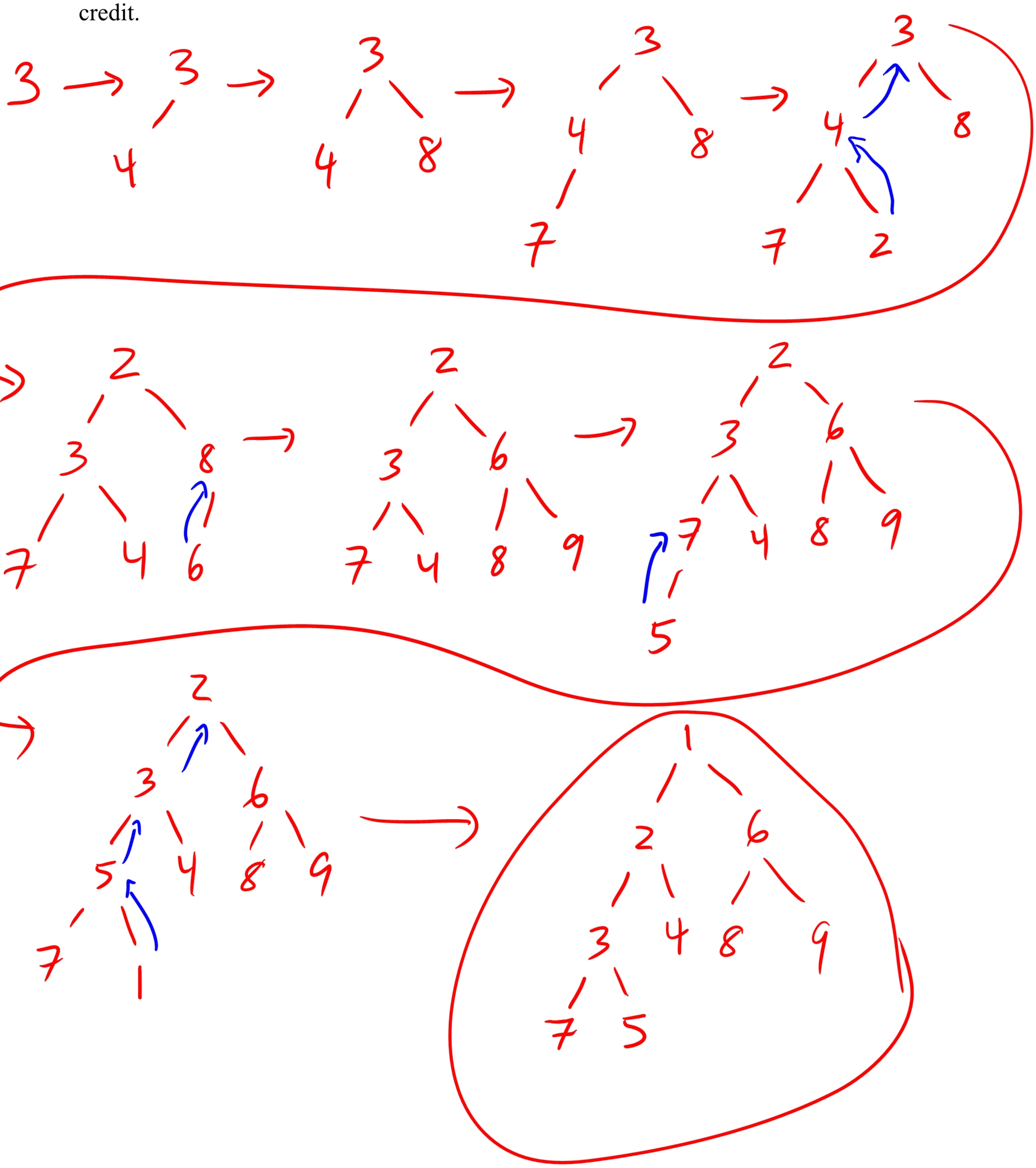
$O(n^4)$

IV.
```
void funny (int n, int x) {
    for (int k = 0; k < 100; ++k)
       if (x > 500) {
          for (int i = 0; i < n * k; ++i)
             for (int j = 0; j < n; ++j)
                System.out.println("x = " + x);
       }
    }
```

$O(n^2)$

6. **Binary Min Heaps**

(a) Draw the binary min heap that results from inserting 3, 4, 8, 7, 2, 6, 9, 5, 1 <u>in that order</u> into an initially empty binary min heap. *You do not need to show the array representation of the heap*. You are only required to show the final tree, although drawing intermediate trees may result in partial credit. If you draw intermediate trees, please **<u>circle your final result</u>** for any credit.



6. **(cont.)**

(b) Draw the result of one deletemin call on your heap drawn at the end of part (a).