**LAST NAME, FIRST NAME**

CS 4520/6520

**Submit TYPED, in PDF electronic format on or before February 6th, 5.30pm.**

No late submissions accepted, no other formats could be accessed.

Please do NOT change HW layout and leave 1 problem per 1 page.

This is needed for TA to use correctly with Gradescope system for grading, so it is important the format is the same for all students in class. Thank you!

**Homework #1**

**Problem 1 [15pts].** Recall how we solved recurrence relation to find the Big-O (first you need to find closed-form formula). Use same method (expand-guess-verify) to figure out Big-O of this relation. (You can skip last step “verify”, which is usually done by math induction).

**T (1) = 1**

**T(n) = T(n-1) + 5**

**Expand:**

**T(n) = (T(n-1)+5) +5**

**T(n) = ((T(n-3)+5)+5)+5**

**.**

**.**

**T(n) = T(n-k)+ k\*5**

**T(n) = T(1) +5n-5**

**O(n)**

**Problem 2 [30pts].** What are the time complexities of these methods?

Briefly explain how you counted, for all cases. ***Be careful***!

**a)**

int fun(int n)

{

    for (int i = 1; i <= 2n; i++)

       for (int j = 1; j < n; j += i)

            // printing stuff

}

O(2nlogn) = O(nlogn)

**b)**

|  |
| --- |
| int mystery (int n)  {    int count = 0;    for (int i = n; i > 0; i /= 2)       for (int j = 0; j < i; j++)          count += 1;    return count;  } |

O(n)

**c)** Be careful about power function. You may want to start with a few examples (k=1, k=2, k=3, and figure out the pattern). Explain your answer.

void magic(int n, int k)

{

    for (int i=1; i<=n; i++)

    {

      int p = pow(i, k);

      for (int j=1; j<=p; j++)

      {

          // printing stuff

      }

    }

}

O(n^(k+1))

**Problem 3 [20pts].** *Design algorithm! Critical thinking.*

Given an array representation of min-heap, convert it to max-heap. You don’t have to code it (although it is a good practice) but need to explain in detail your proposed algorithm.

Analyze the running time complexity of your solution (and always think - can you do better?)  
*Example:*

**Input**: arr[] = [3 5 7 6 8 20 10 12 18 9]

3

/ \

5 7

/ \ / \

6 8 20 10

/ \ /

12 18 9

**Output**: arr[] = [20 18 10 12 9 7 3 5 6 8] OR

[any max-heap formed from input elements]

20

/ \

18 10

/ \ / \

12 9 7 3

/ \ /

5 6 8

Important note: do not overthink this problem. All you are asked to do is build a max-heap.

**MaxHeap(A.i)**

**l = left(i)**

**r = right(i)**

**if i<A.heapsize and A[l]>=a[i]**

**large = l**

**else:**

**large = i**

**if r<A.heapsize and A[r] >= A[large]**

**large =r**

**if large !=i**

**swap a[i] and a[large]**

**MaxHeap (A,large)**

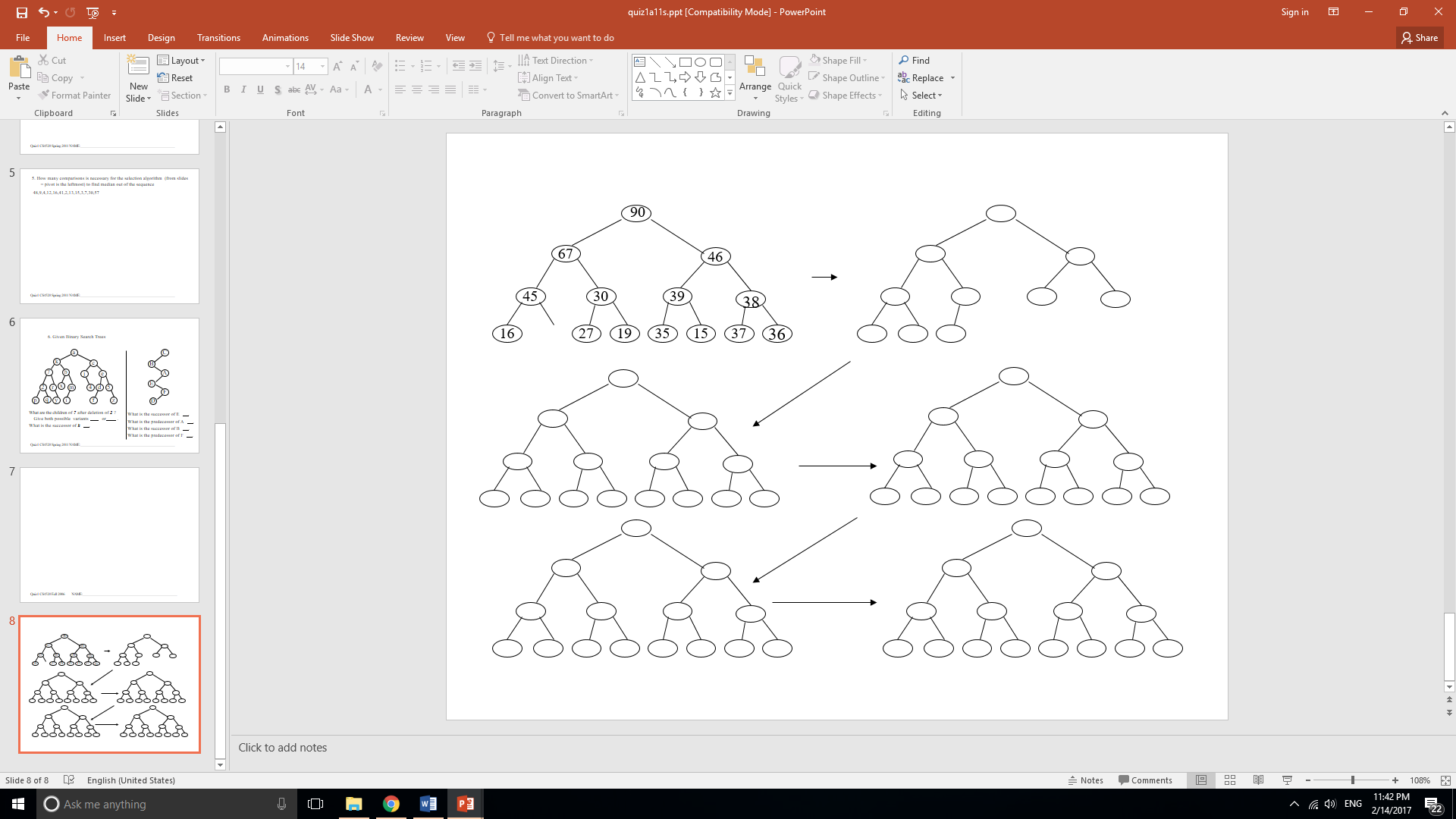
**Time Complexity: O(nlogn)**

**Problem 4 [15pts].**

**max**-**heap** is a binary tree in which the value in each internal node is greater than (or equal) to the values in the children of that node.

Do ***heapsort*** for the following max-heap. Sort from largest to smallest value. Show step-by-step process (you may copy-paste and change values of the nodes. Leave nodes blank when you don’t need them anymore). Stop if you run out of elements or after 12 steps (12 empty templates are given).

16



4

13

1

11

14

2

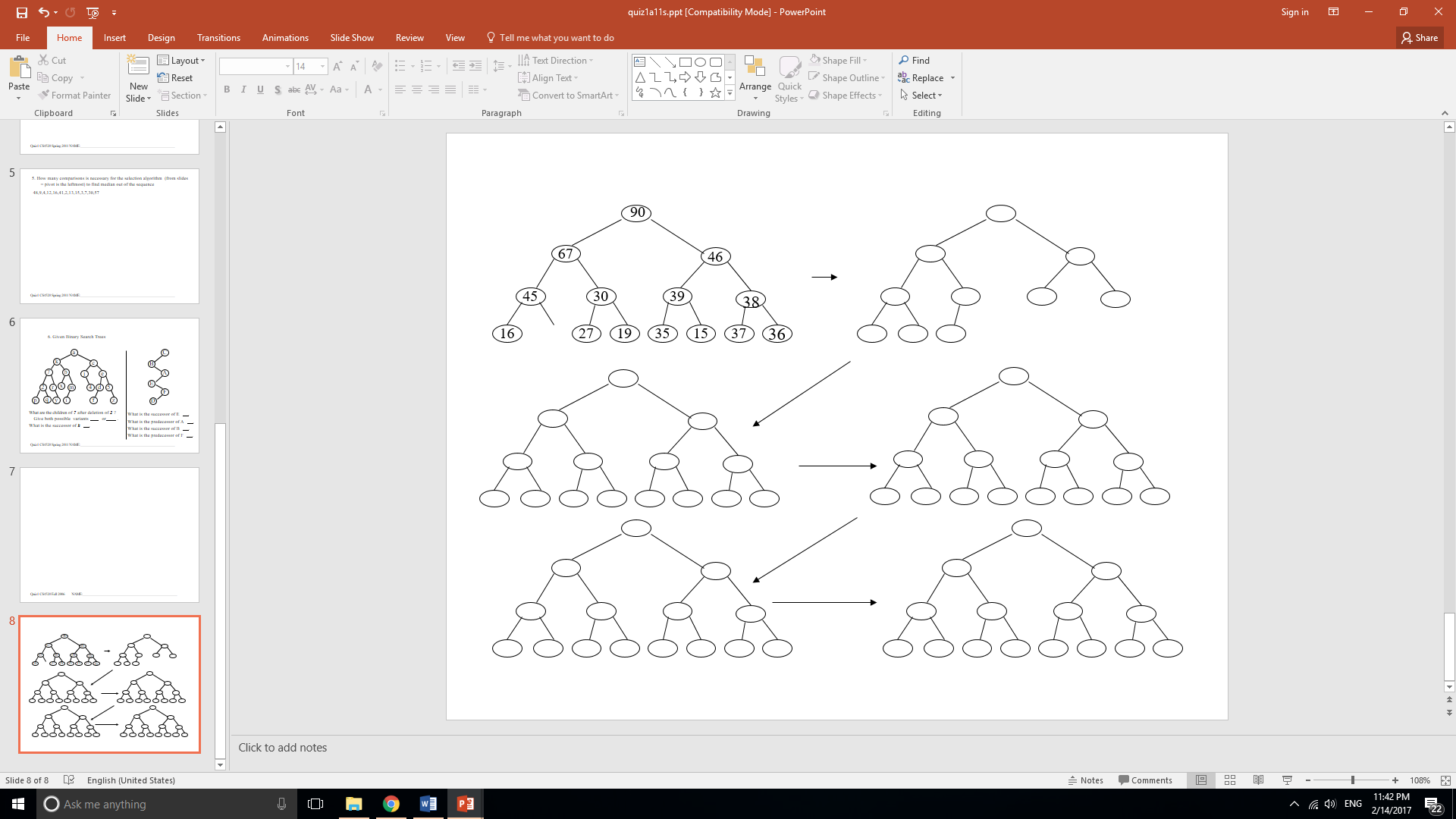
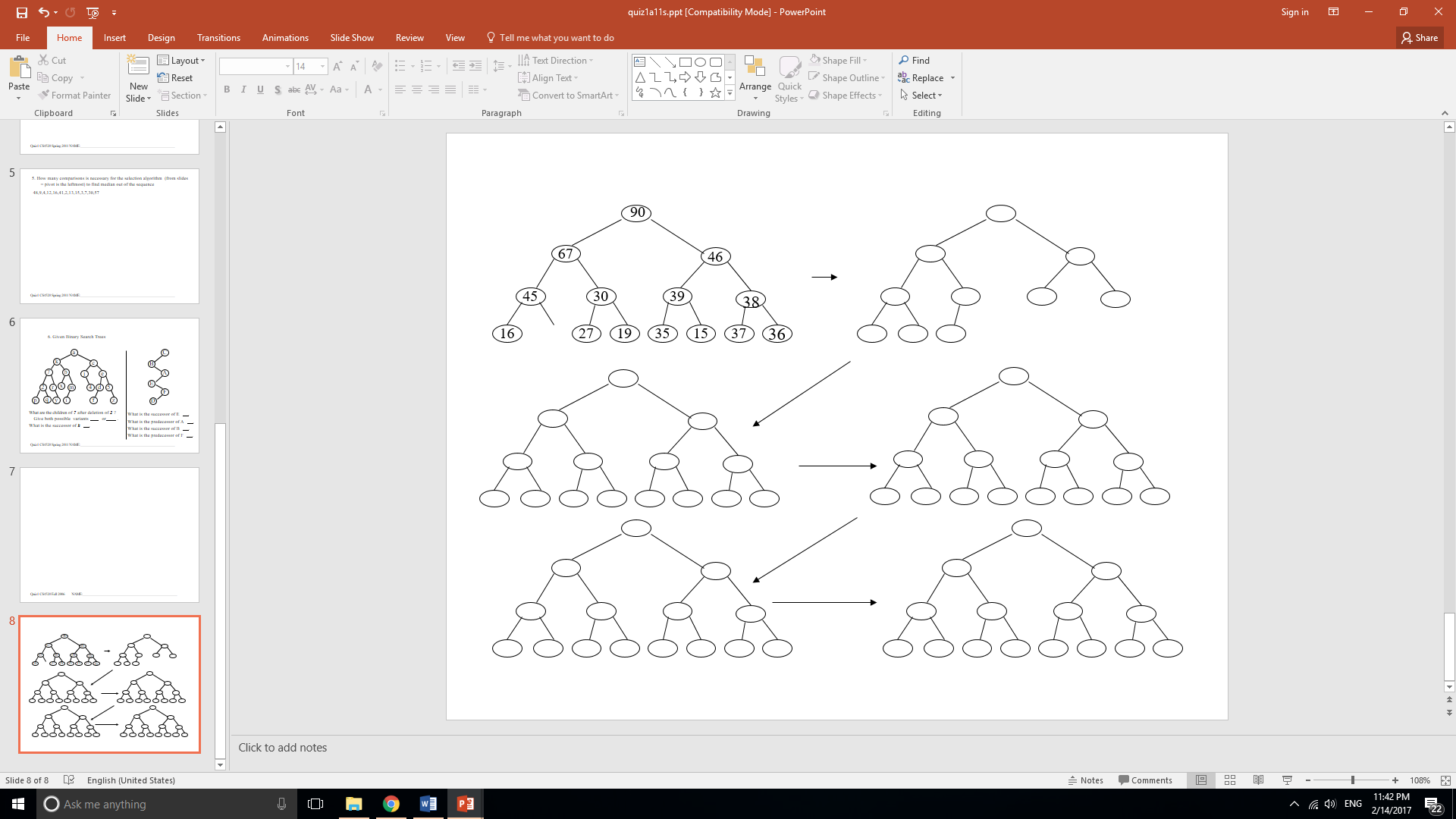
5

7

9

14

1



5

5

2

2

11

11

1

7

13

13

9

4

4

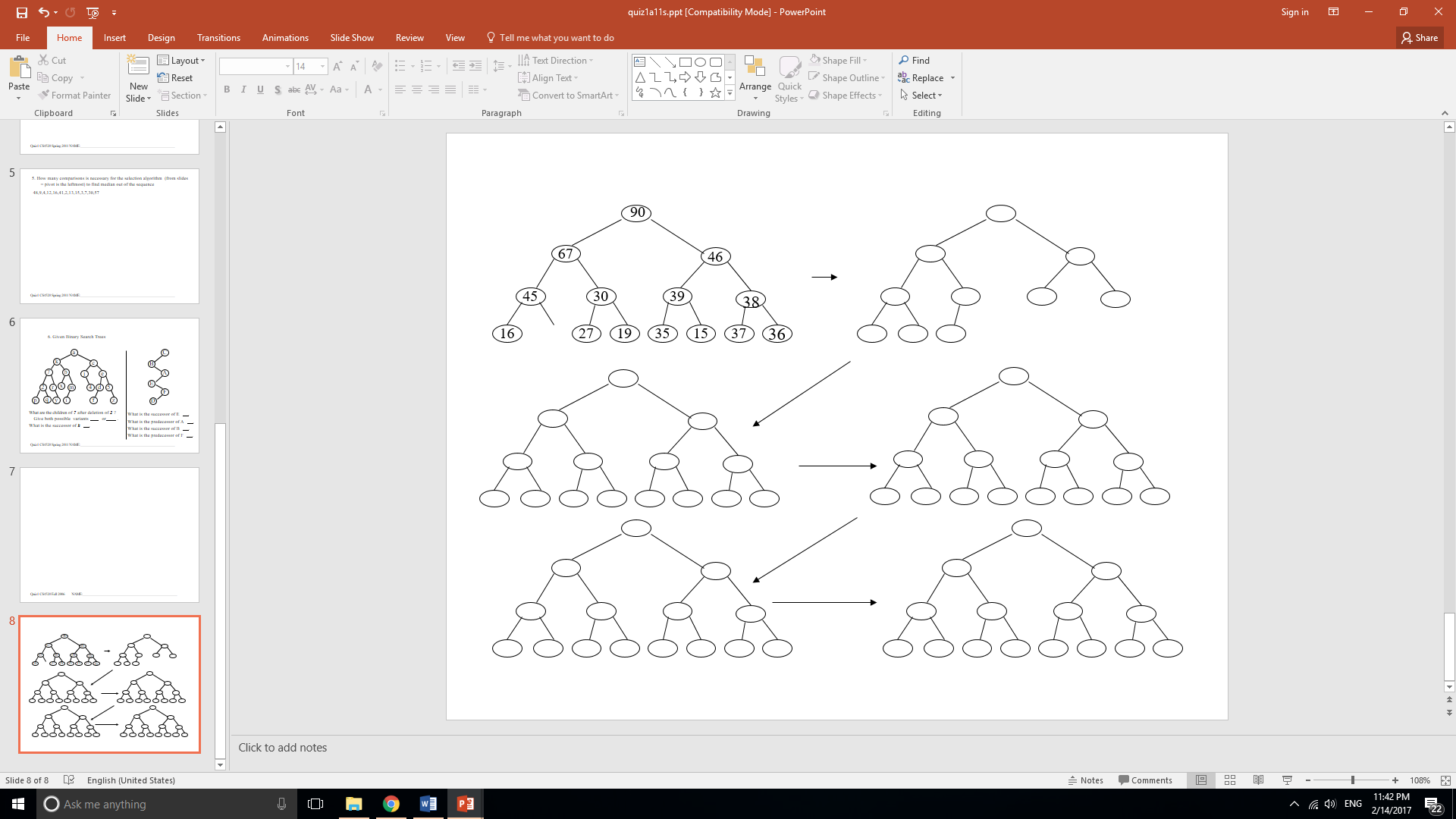
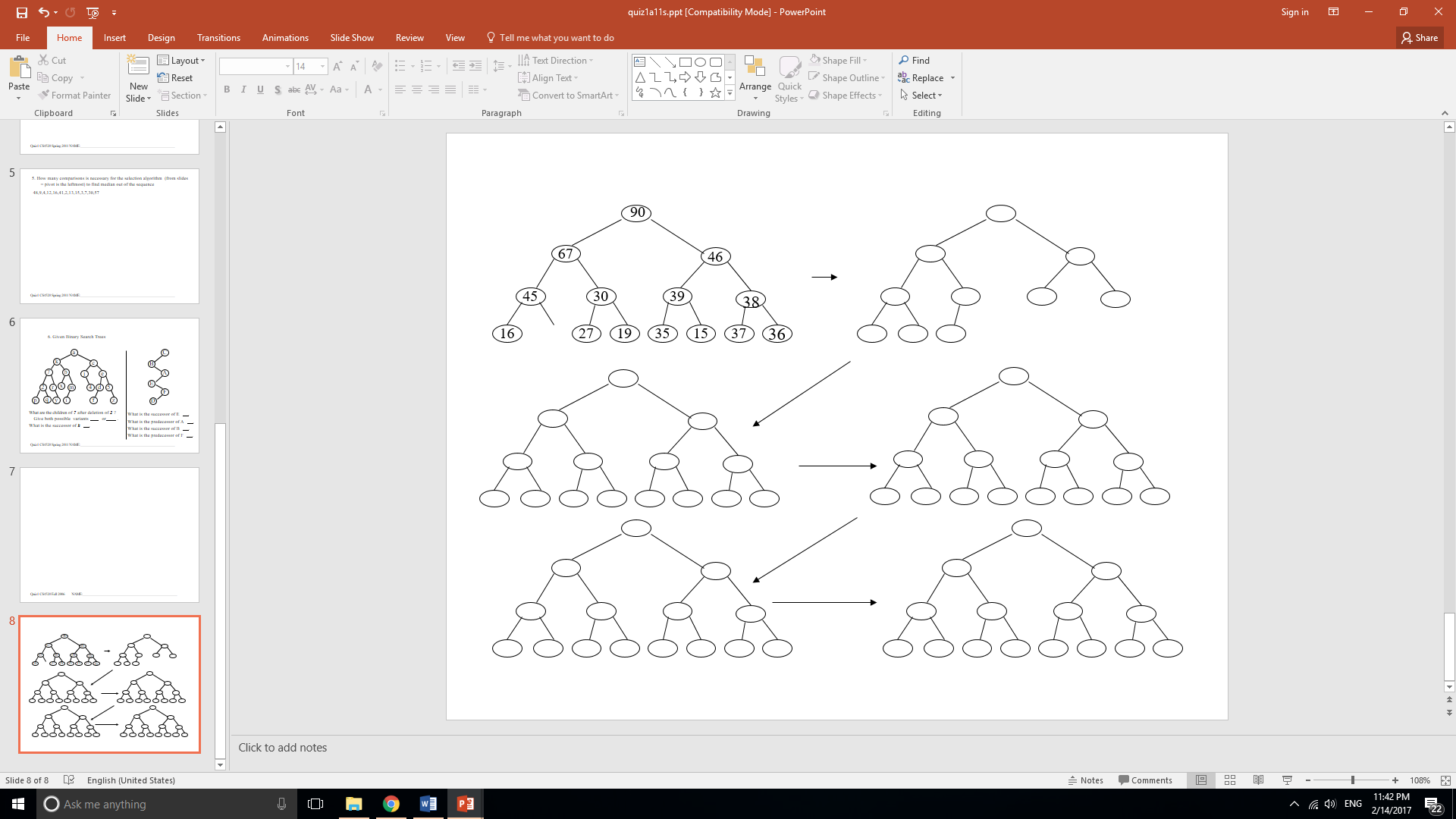
7

14

9

5

14



5

2

2

11

11

1

1

7

7

13

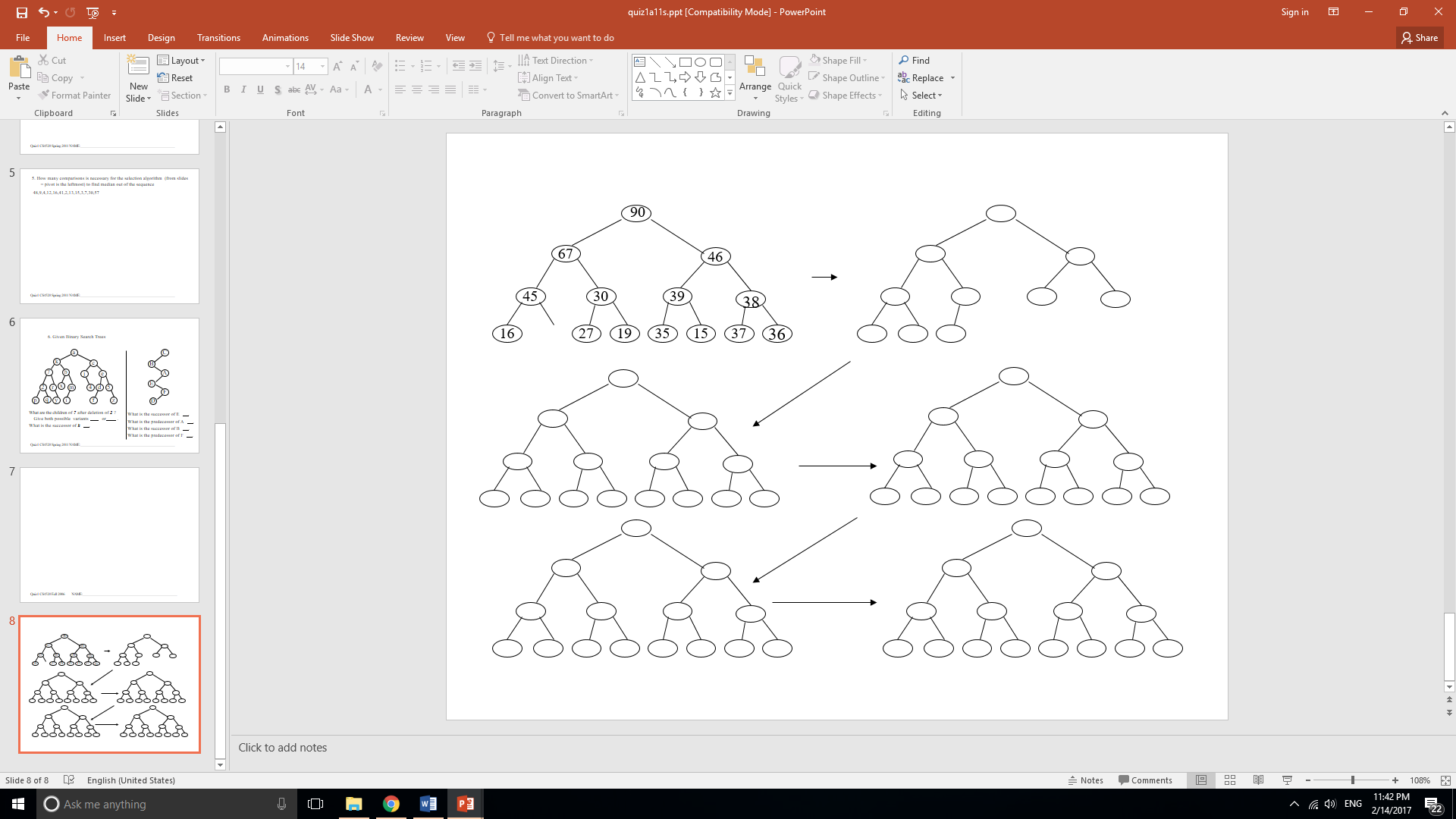
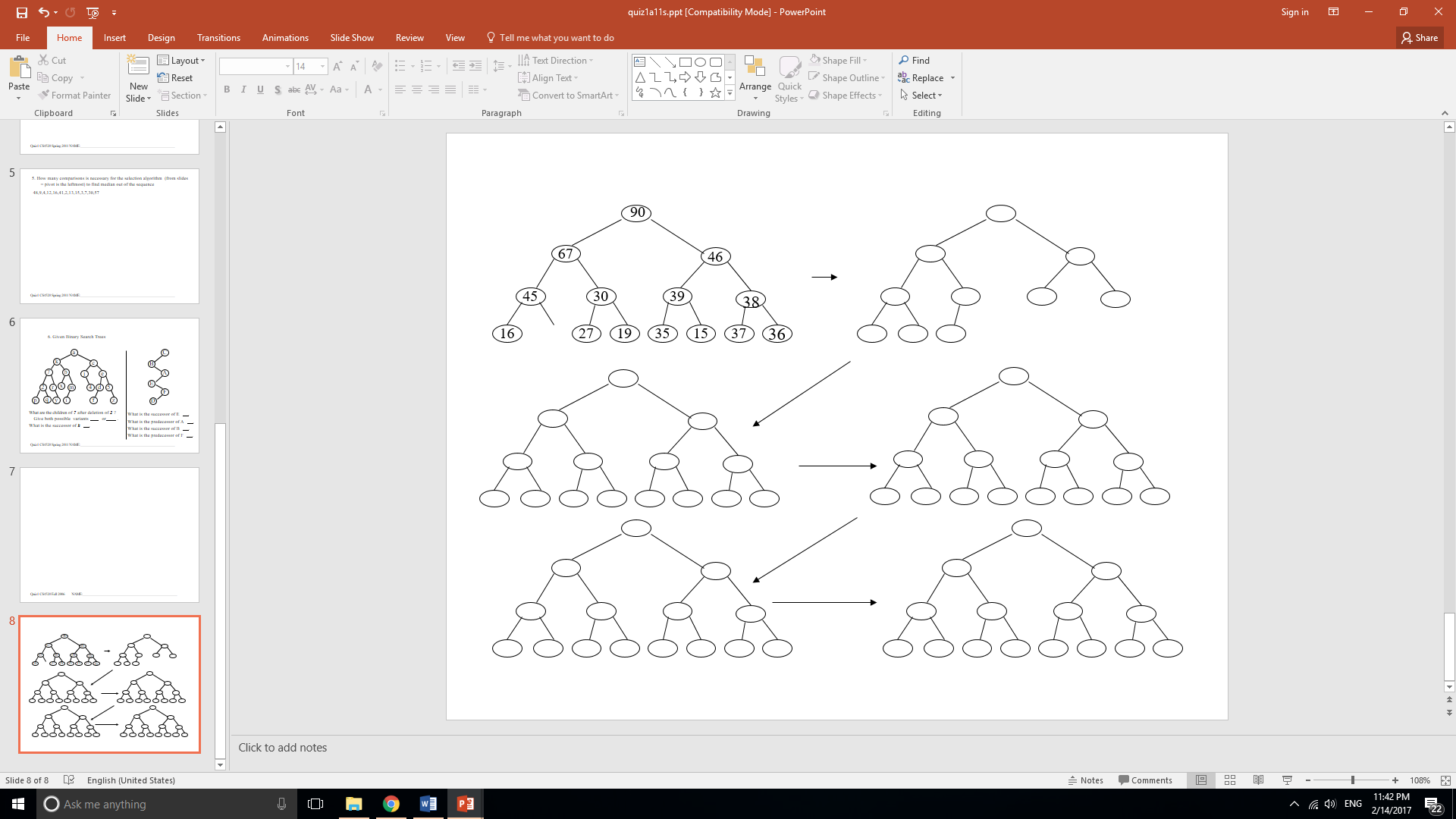
13

9

9

4

4



5

11

2

1

7

13

9

4

13

5

2

11

1

7

9

4

11

5

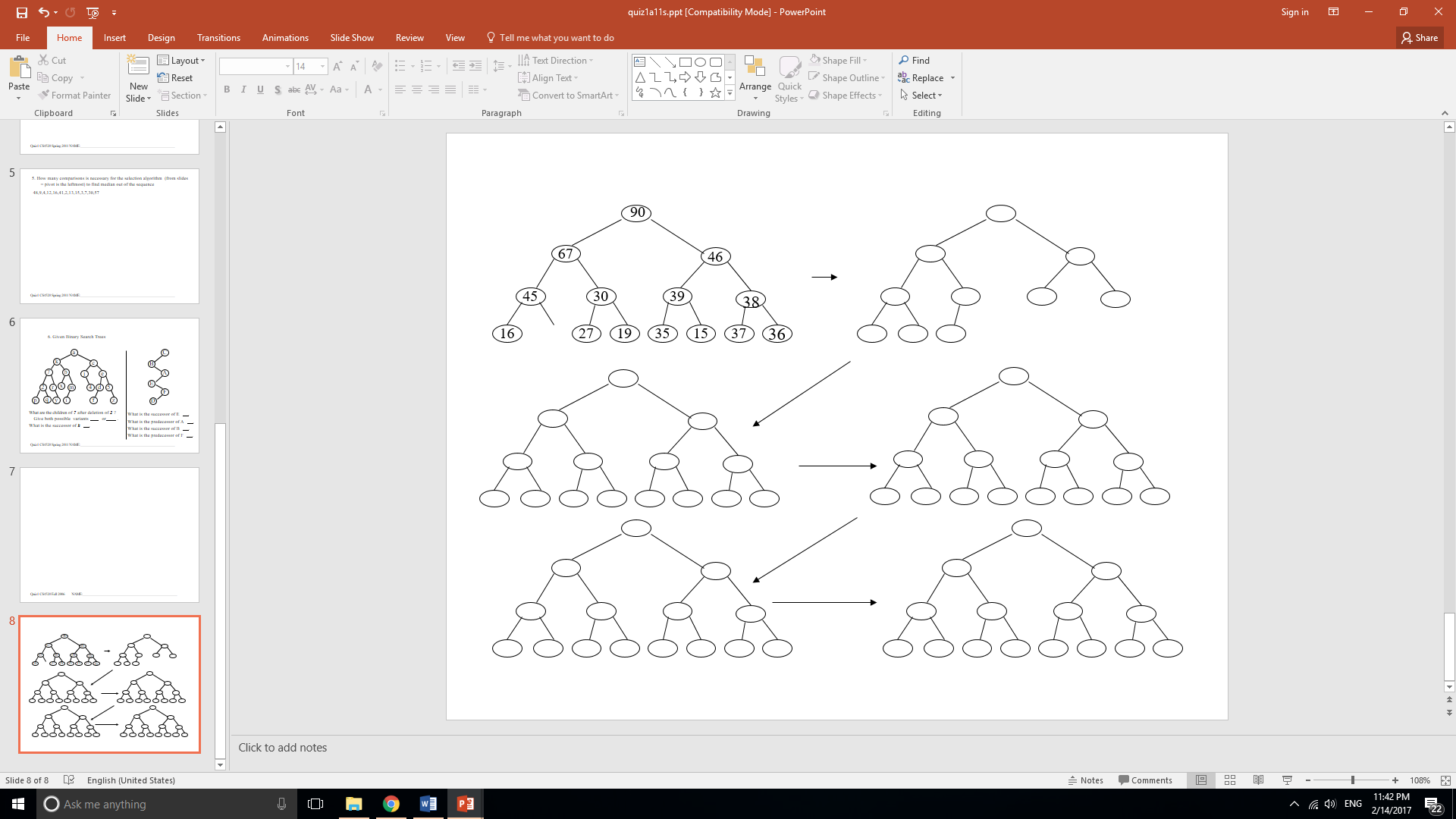
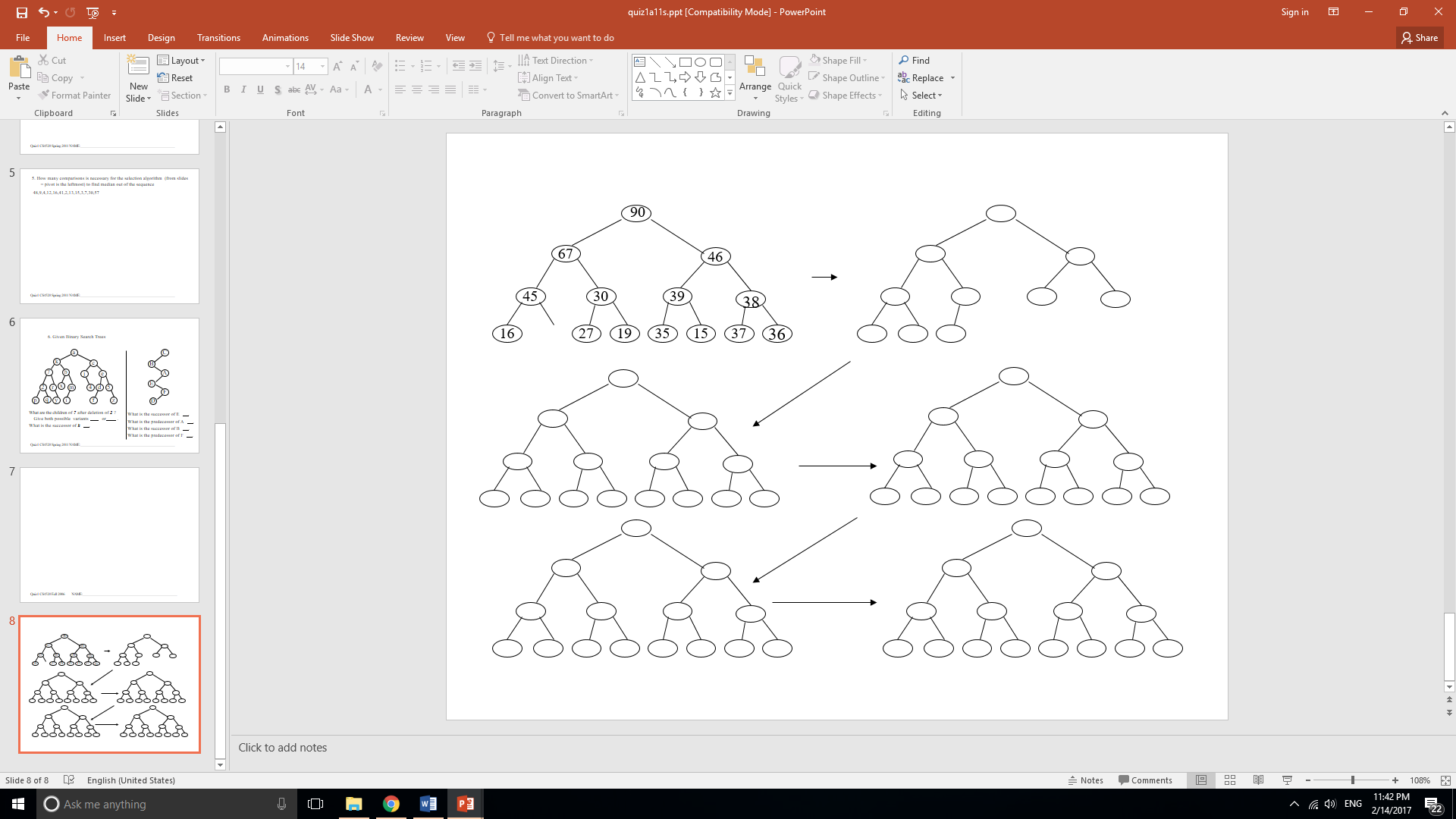
1

7

9

4

2

2

4

9

7

1

11

5

2

11

1

7

9

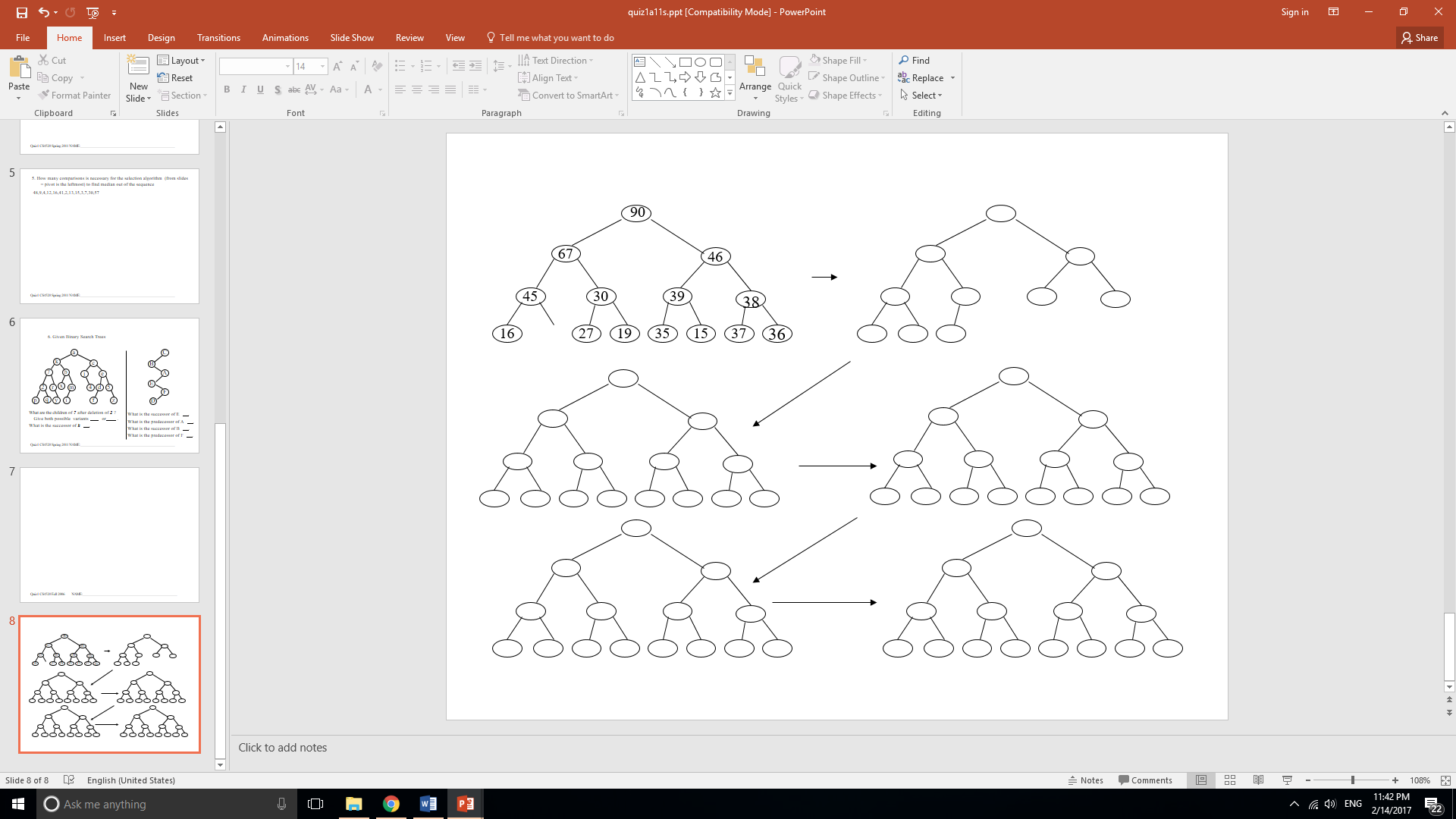
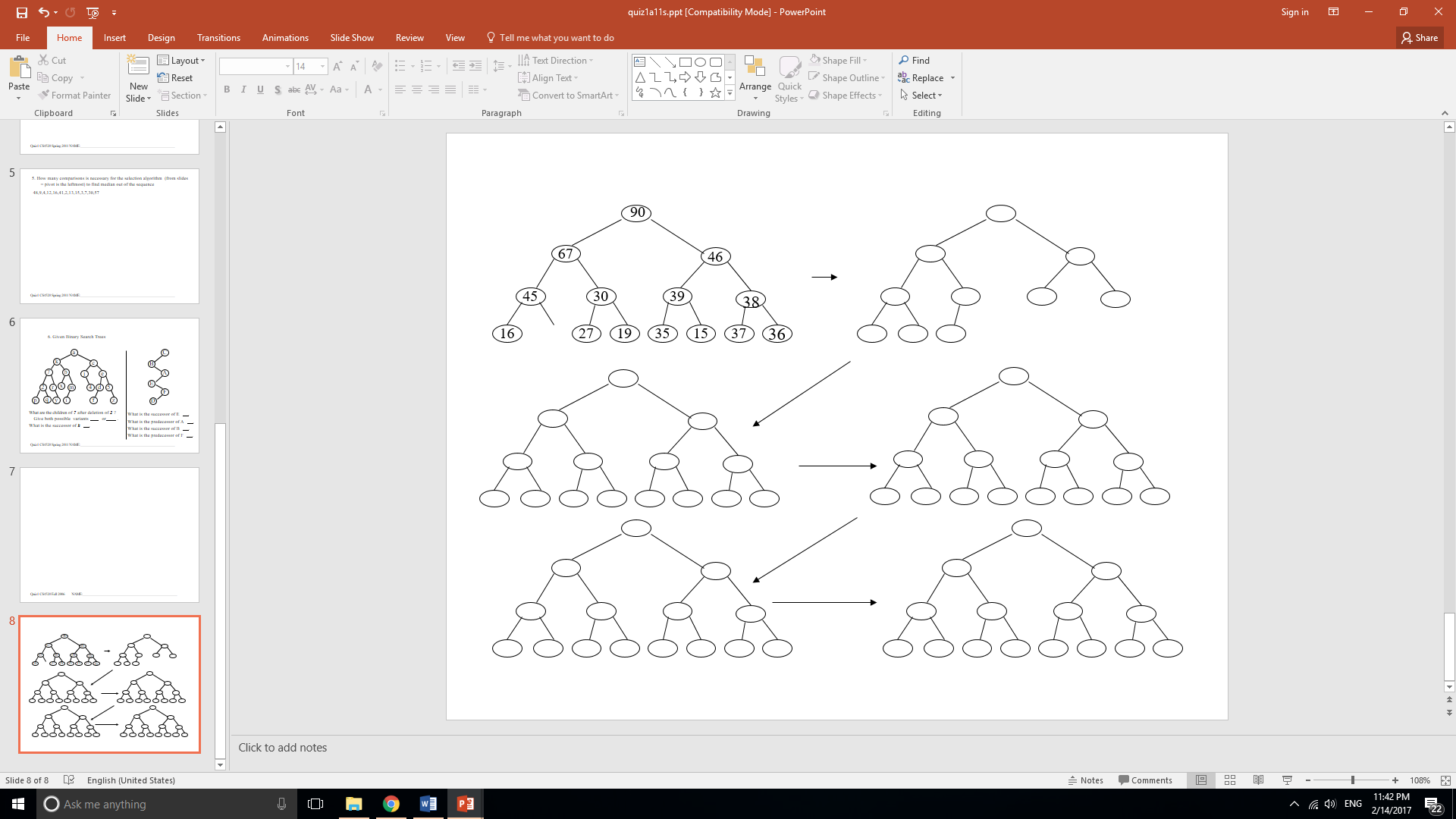
4

4

9

7

1

5

5

2

5

1

9

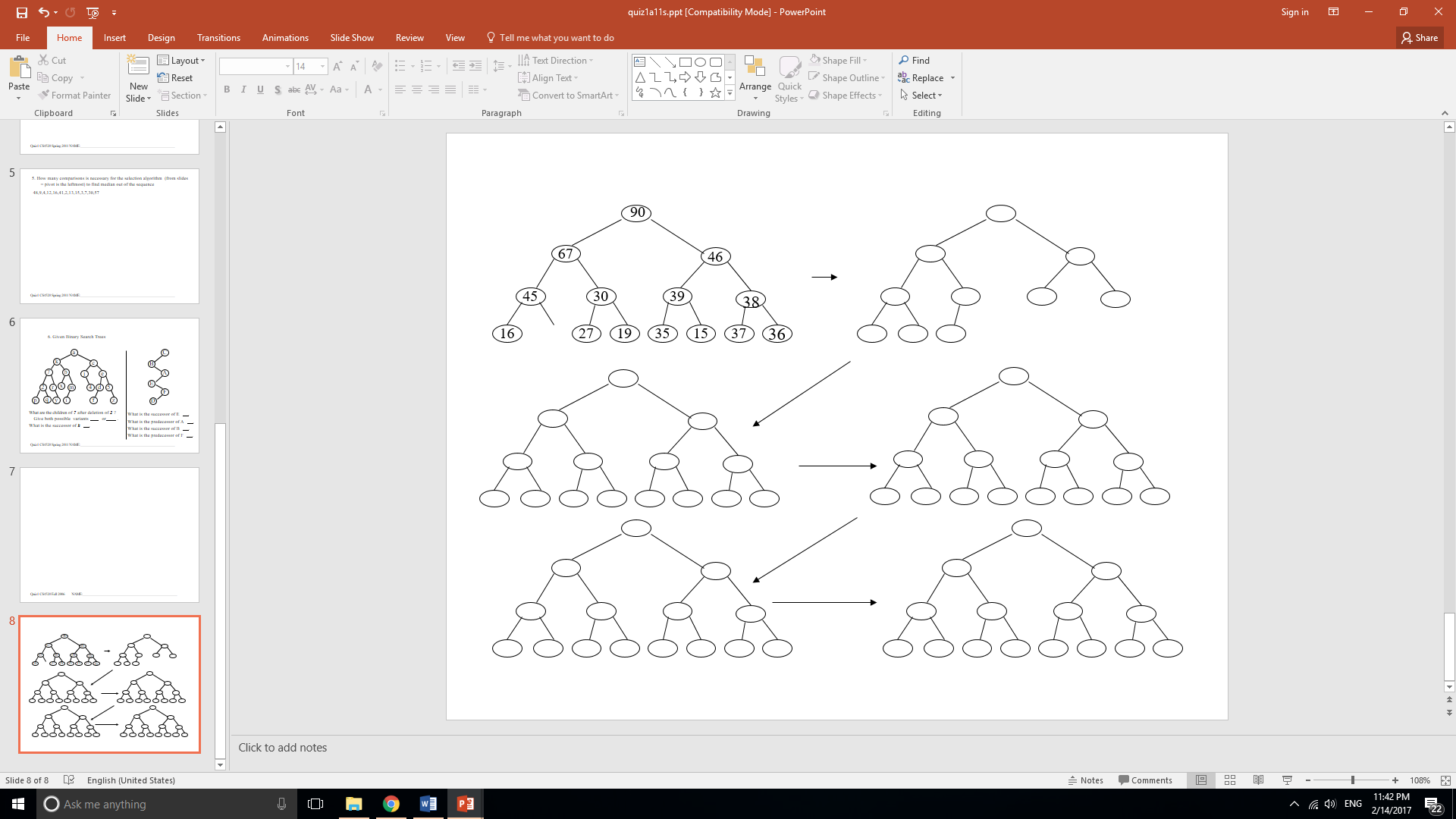
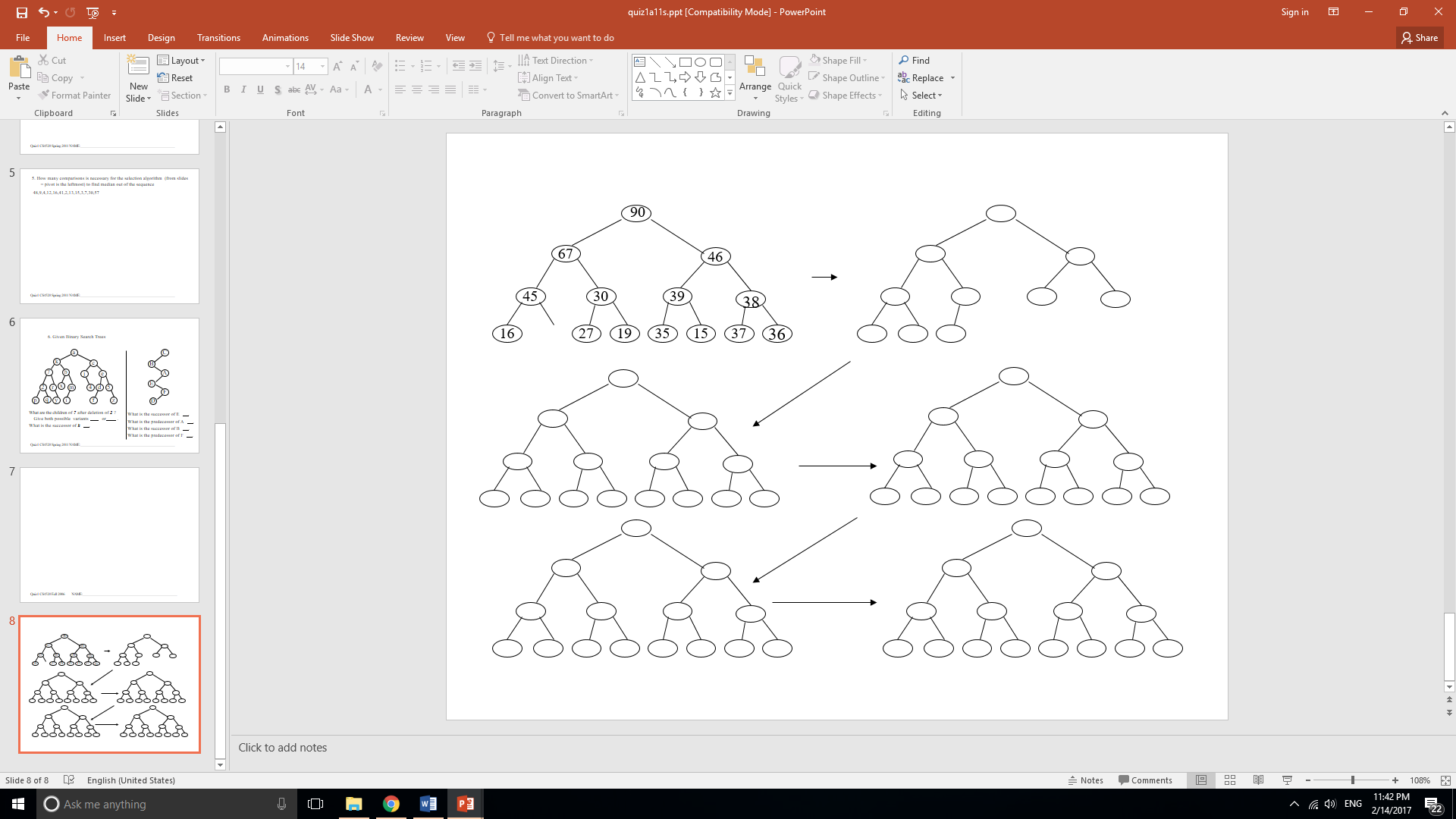
4

9

1

7

4

  Stop here.

7

2

2

5

**Problem 5. [20pts]** *Exploratory question (means google and figure out the answer!)*

What is the definition of **space complexity** of an algorithm?

Space complexity is ...

What would be space and time complexities of the following code snippets (in terms of Big-O)?

a)

int mystery(int x, int y, int z)

{

int r=x+y+z;

return r;

}

Time: O(1)

Space: O(1)

b)

void foo (int[] mylist){

for (int i=0; i<mylist.length; i++)

//print stuff

}

Time: O(n)

Space:O(1)

Does space and time complexities dependent on each other in some way? If so, how?

No

**Please go back to the beginning of this document and make sure it is consistent with requirements mentioned there: 1 problem per 1 page. Thank you!!**