JEOPARDY

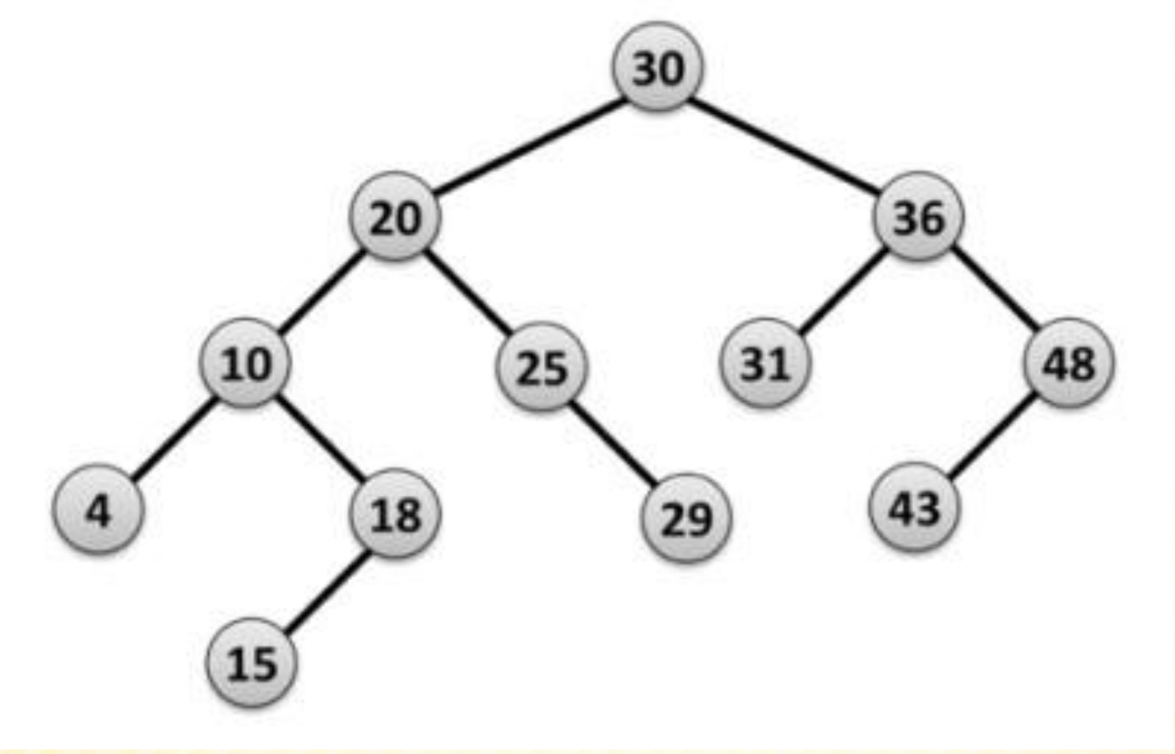
COP 3502

BSTs & AVL Trees – Q1

- Why would you use an AVL tree versus a Binary Search Tree?
  - Faster Search/Insert/Delete in a balanced tree versus an unbalanced tree.
  - In a balanced tree the Run-time of Search/Insert/Delete is O(log n)
    - but if a branch becomes deep the Run-time approaches O(n).

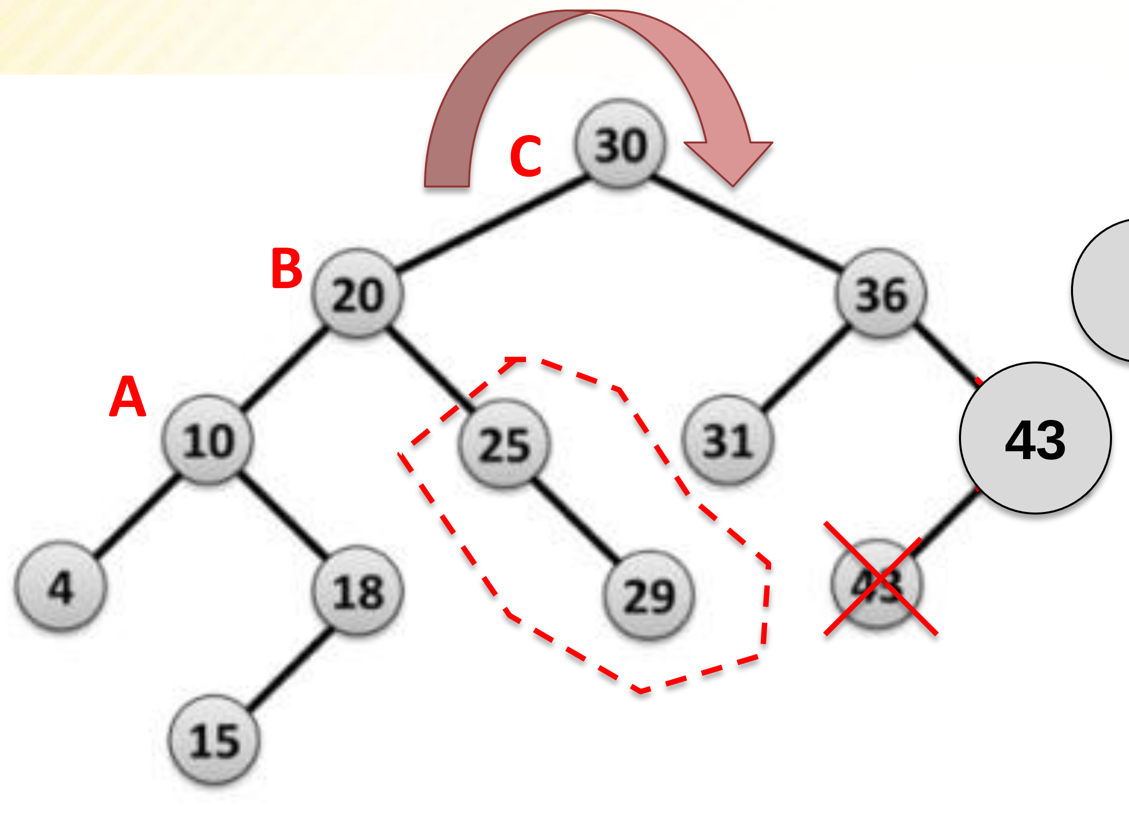
BSTs & AVL Trees – Q2

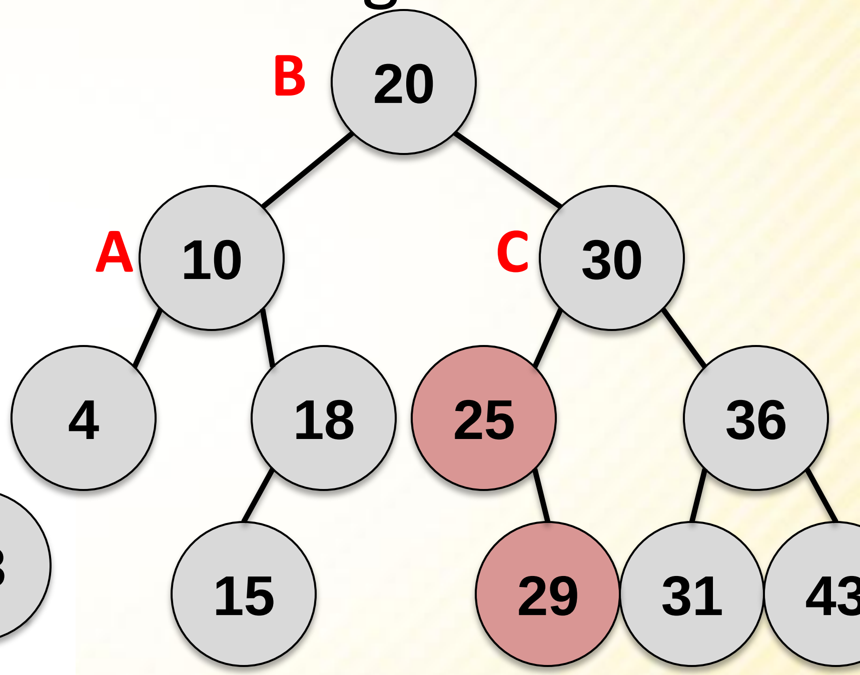
- Show the state of the AVL tree after deleting node 48 and doing any necessary rebalancing:



BSTs & AVL Trees – Q2

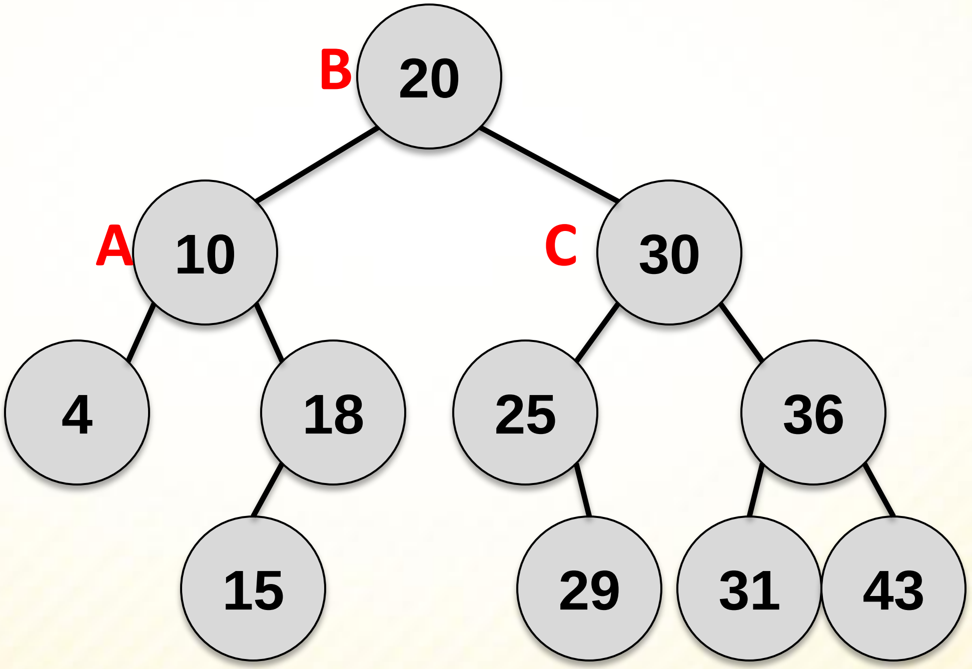
- Show the state of the AVL tree after deleting node 48 and doing any necessary rebalancing:





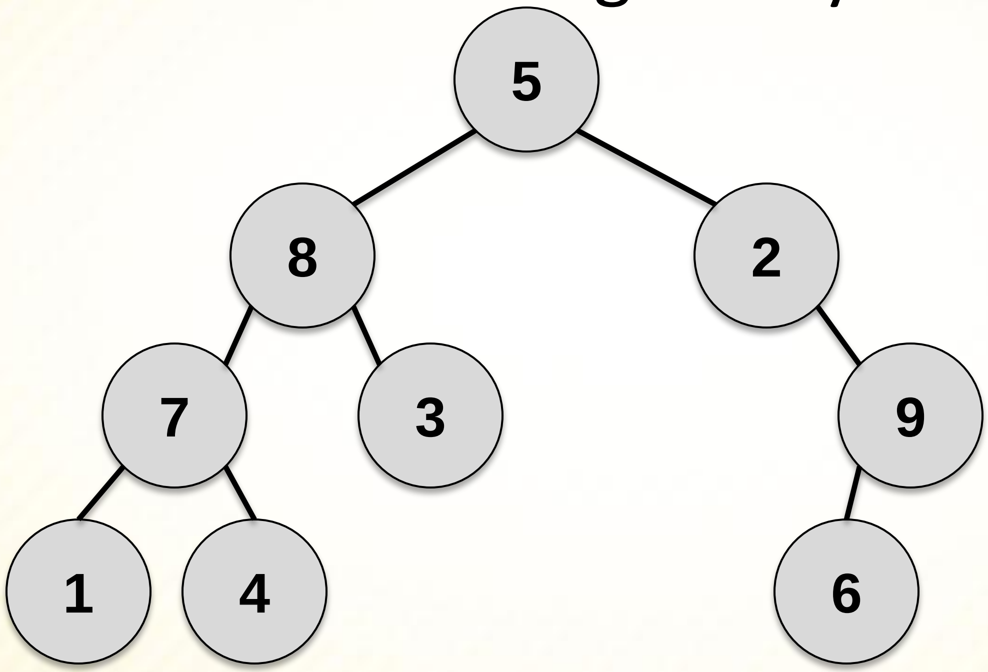
BSTs & AVL Trees – Q2

- Show the state of the AVL tree after deleting node 48 and doing any necessary rebalancing:



BSTs & AVL Trees – Q3

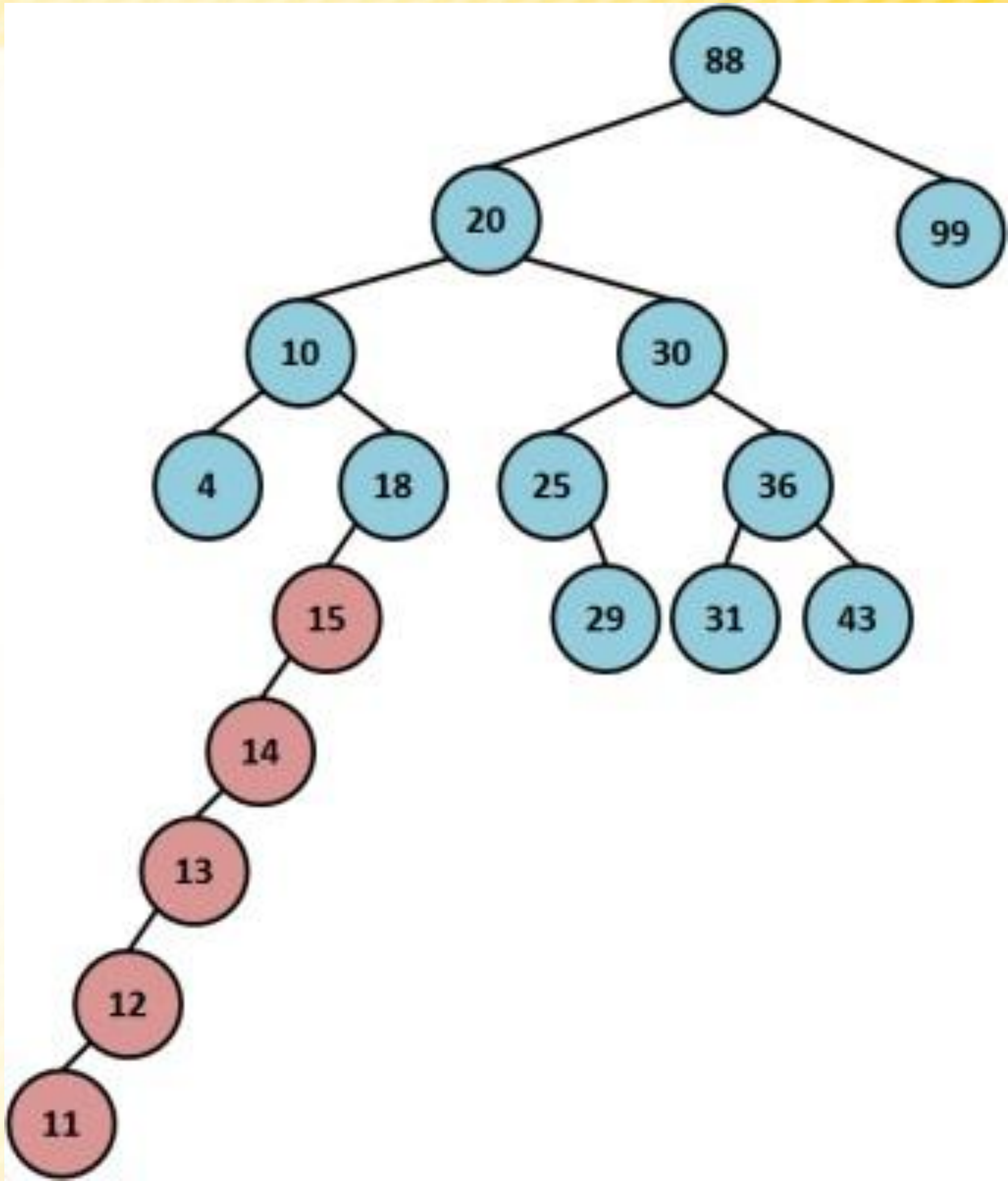
- What are the PreOrder, InOrder, and PostOrder traversals of the following Binary Tree?



- PreOrder: 5,8,7,1,4,3,2,9,6
- InOrder: 1,7,4,8,3,5,2,6,9
- PostOrder: 1,4,7,3,8,6,9,2,5

BSTs & AVL Trees – Q4

- What is the height of the following tree?
  - 8



BSTs & AVL Trees – Q5

- Write a recursive function to free the memory in a Binary Tree:

```
void FreeBST(node *root) {
    if (root != NULL) {
        FreeBST(root->left);
        FreeBST(root->right);
        free(root);
    }
}
```

Hash Tables & Heaps – Q1

- What index would 8 be inserted into in the following hash table using Quadratic Probing with the hash function ***x^2 + 7 % 13***:

index	0	1	2	3	4	5	6	7	8	9	10	11	12
val				3				0	1			2	

- 10

Hash Tables & Heaps – Q2

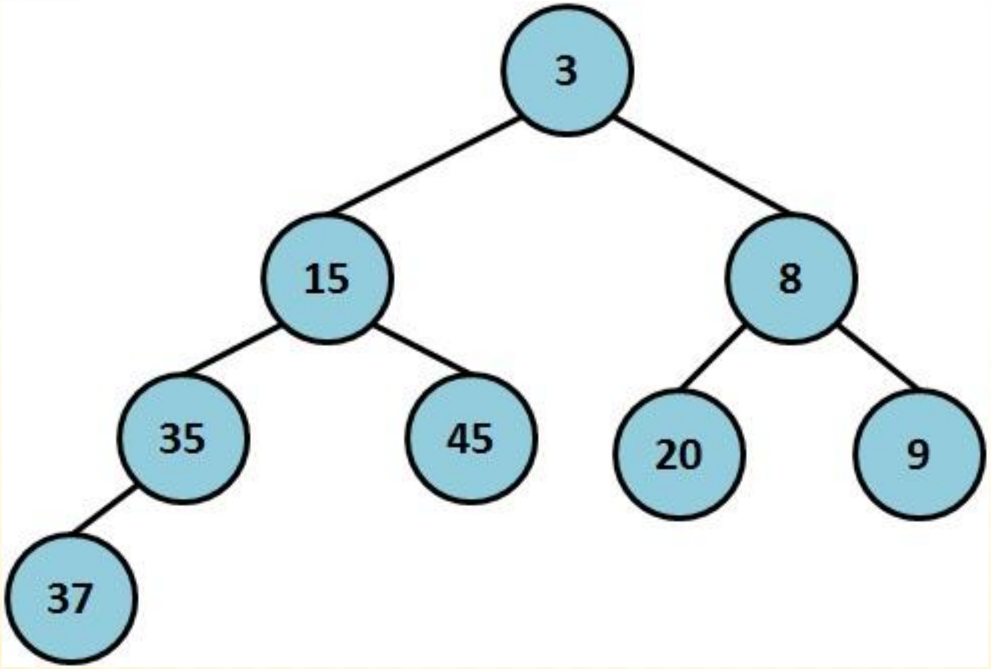
- What is the purpose of a hash table?
  - Very fast search, insert, and delete times: O(1) with a perfect hash function.

Hash Tables & Heaps – Q3

- What are the two uses for Heaps given in class?
  - Priority Queues and Heap Sort.

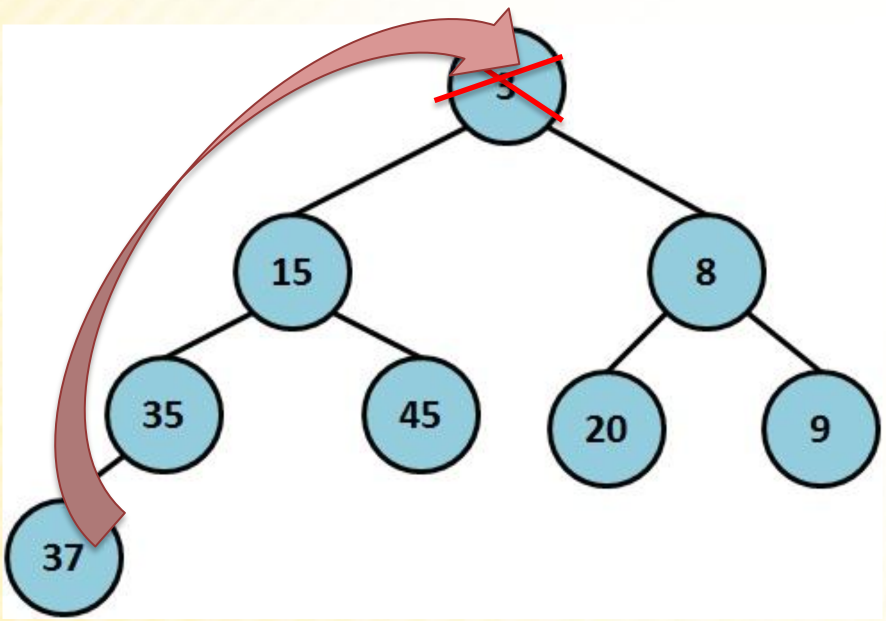
Hash Tables & Heaps – Q4

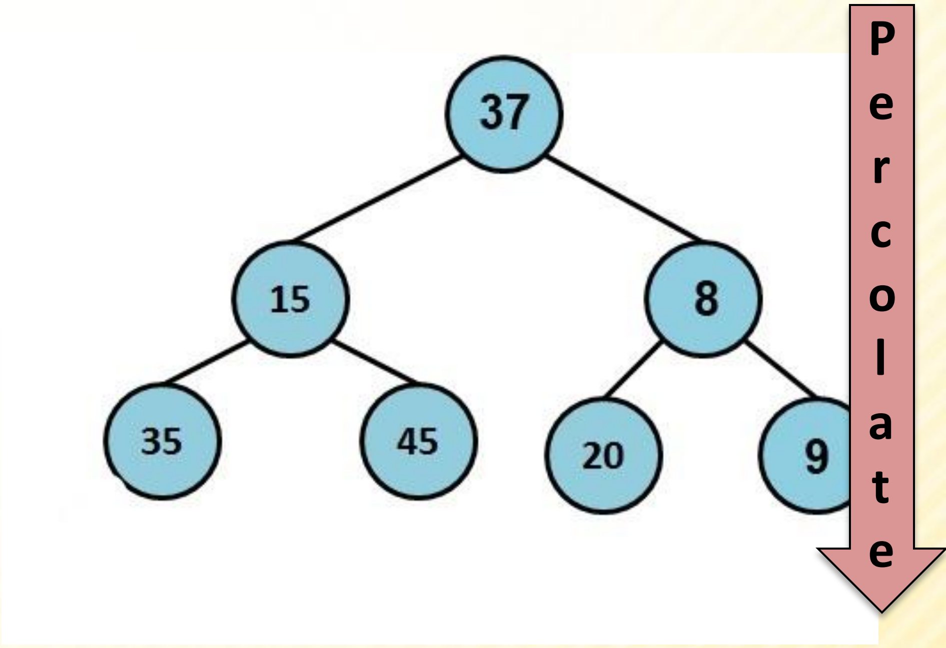
- What is the resulting heap after Deleting the Minimum element from the following heap?



Hash Tables & Heaps – Q4

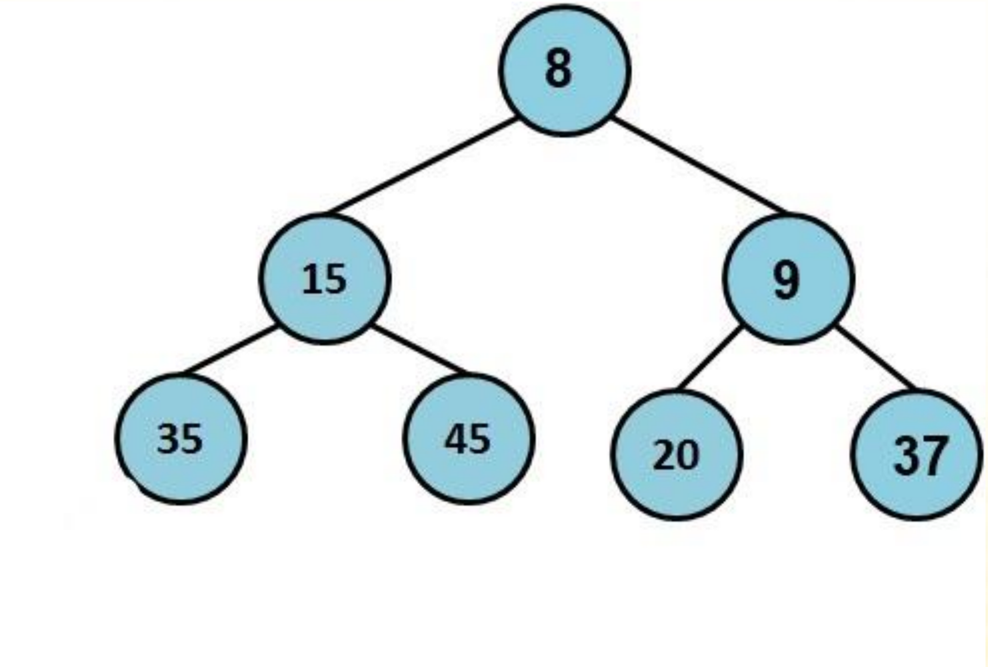
- What is the resulting heap after Deleting the Minimum element from the following heap?





Hash Tables & Heaps – Q4

- What is the resulting heap after Deleting the Minimum element from the following heap?



Hash Tables & Heaps – Q5

- Using Big-O notation, what is the run-time of:
  - (a) Inserting 10 items into an initially **empty** binary heap
  - (b) Inserting 10 items into a binary heap with ***n*** elements.

  - **O(1)**
  - **O(log n)**

Sorting – Q1

- Fill in the table to show the resulting array after each pass in Bubble Sort:

Initial	4	2	6	5	7	1	8	3
Sorted	1	2	3	4	5	6	7	8

Sorting – Q1

- Fill in the table to show the resulting array after each pass in Bubble Sort:

Initial	4	2	6	5	7	1	8	3
	2	4	5	6	1	7	3	8
	2	4	5	1	6	3	7	
	2	4	1	5	3	6		
	2	1	4	3	5			
	1	2	3	4				
Sorted	1	2	3	4	5	6	7	8

Sorting – Q2

- Show the result of running Partition on the array below using the leftmost element as the pivot element. Show the array after each swap.

Initial	4	2	6	5	7	1	8	3
Swap1								
Swap2								
Partitioned								

Sorting – Q2

- Show the result of running Partition on the array below using the leftmost element as the pivot element. Show the array after each swap.

Initial	**4**	**2**	**6**	**5**	**7**	**1**	**8**	**3**
Swap1	4	2	**3**					**6**
Swap2	4	2	3	**1**	7	**5**	8	6
Partitioned	**1**	2	3	**4**	7	5	8	6

Sorting – Q3

- Fill in the table to show the array after each call to the Merge function in Merge Sort.

Initial	5	2	6	4	7	1	8	3
Sorted	1	2	3	4	5	6	7	8

Sorting – Q3

- Fill in the table to show the array after each call to the Merge function in Merge Sort.

Initial	5	2	6	4	7	1	8	3
	2	**5**	6	4	7	1	8	3
	2	5	**4**	**6**	7	1	8	3
	2	**4**	**5**	**6**	7	1	8	3
	2	4	5	6	**1**	**7**	8	3
	2	4	5	6	1	7	**3**	**8**
	2	4	5	6	**1**	**3**	**7**	**8**
Sorted	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**

Sorting – Q4

- What is the Worst Case run-time of Insertion Sort, Selection Sort, and Bubble Sort respectively?
- What is the Best Case of each?
  - $O(n^2)$, $O(n^2)$, $O(n^2)$
  - $O(n)$, $O(n^2)$, $O(n^2)$

Sorting – Q5

- What is the Best Case and Worst Case for finding the kth smallest integer out of an unsorted array of n integers. (k <= n)

  - Best Case: O(n) , Worst Case: $O(n^2)$

Stacks & Queues – Q1

- What is the acronym for describing the push and pop rules for Stacks and what does it stand for?
  - LIFO – Last In, First Out.

Stacks & Queues – Q2

- Show the final contents of the Array-Implemented Queue, the index of front, and numElements – after running this code:

```
enqueue(Q1, 8);
enqueue(Q1, 3);
dequeue(Q1);
enqueue(Q1, 6);
enqueue(Q1, 7);
dequeue(Q1);
enqueue(Q1, 9);
```

Q1: elements:

front: ____________

numElements: ____________

Stacks & Queues – Q2

- Show the final contents of the Array-Implemented Queue, the index of front, and numElements – after running this code:

```
enqueue(Q1, 8);
enqueue(Q1, 3);
dequeue(Q1);
enqueue(Q1, 6);
enqueue(Q1, 7);
dequeue(Q1);
enqueue(Q1, 9);
```

FRONT: **-1 0 1 2**

NUMELEMENTS: **0 1 2 3**

Q1: elements:

8	3	6	7	9					

Stacks & Queues – Q3

- What two implementations of Queue's were used in HW #4? What was each one used for?
  - Array implementation – Router
  - Linked List implementation – each device's request queue.

Stacks & Queues – Q4

- What are the run-times of the following operations:
  - Stacks: Push and Pop
  - Queues: Enqueue and Dequeue
  - O(1) for all

Stacks & Queues – Q5

- Convert the following infix expression to postfix:
  - (A / (B – C) + D) * (E – F) + G * H
  - A B C – / D + E F – * G H * +

Algorithm Analysis – Q1

- What is the Big-O run-time of deleting one node from an AVL tree with ***n*** nodes?
- What is the Big-O run-time of deleting one node from an AVL tree with height ***h***?

- O(**log n**) and O(**h**)

Algorithm Analysis – Q2

- What is the Big-O solution to the following recurrence relation?
  - ***T(n) = 2T(n/2) + n, assume T(1) = 1***

- **O(n log n)**

Algorithm Analysis – Q3

- Determine a simplified closed-form solution for the following summation in terms of n:

$$\sum_{i=1}^{3n} \sum_{j=n+1}^{5n} (5i + 3j)$$

Algorithm Analysis – Q3

- Determine a simplified closed-form solution for the following summation in terms of n:

$$\sum_{i=1}^{3n}\sum_{j=n+1}^{5n}(5i+3j)$$

$$\sum_{i=1}^{3n}\sum_{j=n+1}^{5n}5i+\sum_{i=1}^{3n}\sum_{j=n+1}^{5n}3j$$

$$\sum_{i=1}^{3n}4n*5i+\sum_{i=1}^{3n}\sum_{j=n+1}^{5n}3j$$

$$4n*5(3n(3n+1)/2)+\sum_{i=1}^{3n}\sum_{j=n+1}^{5n}3j$$

$$90n^3+30n^2+\sum_{i=1}^{3n}(\sum_{j=1}^{5n}3j-\sum_{j=1}^{n}3j)$$

$$90n^3+30n^2+\sum_{i=1}^{3n}(\frac{3(5n)(5n+1)}{2}-\frac{3n(n+1)}{2})$$

$$90n^3+30n^2+3n(36n^2+6n)$$

$$198n^3+48n^2$$

Algorithm Analysis – Q4

- What is the Big-O running time of the following segment of code, it terms of ***n***.

```
int a = 1, b = n, sum = 0;
while (a < b) {
      sum++;
      a = a*2;
      b = b/2;
}
```

Algorithm Analysis – Q4

- What is the Big-O running time of the following segment of code, it terms of ***n***.

```
int a = 1, b = n, sum = 0;
while (a < b) {
      sum++;
      a = a*2;
      b = b/2;
}
```

- Consider the ratio b/a.
- The loop stops when this ration is 1. For each loop iteration the ratio decreases by a factor of 4. Let k be the number of loop iterations total. Then $1 = n/4^k$. Solving we get $k = \log_4 n$. → O(log n)

Algorithm Analysis – Q5

- If an $O(n^2)$ algorithm takes 40 ms to complete with an input size of n = 20,000, how much time will it take to complete on an input size of n = 50,000?
  - $c * n^2 = 40ms, c = 40/20{,}000^2 = 40 / 400{,}000$
  - $40 / 400{,}000 * (50{,}000^2) = 40 / 400{,}000 * (2{,}500{,}000)$
  - $= 10 * 25 = 250$ ms

Mixed Bag – Q1

- Fill in the blanks of the following recursive sorting function, which of the sorting algorithms that we have seen so far does this resemble?:

```
void sort(int *values, int length) {
  if (length > 1) {
    int maxIndex = 0;
    int i;
    for (i=1; i<length; i++)
      if ( ________(1)_________ )
        maxIndex = i ;
    int temp = values[length-1];
    values[length-1] = ____(2)___ ;
    ________(3)_______ = temp ;
    _____________(4)______________ ;
}
```

Mixed Bag – Q1

- Fill in the blanks of the following recursive sorting function, which of the sorting algorithms that we have seen so far does this resemble?
  - Selection sort.

```
void sort(int *values, int length) {
  if (length > 1) {
    int maxIndex = 0;
    int i;
    for (i=1; i<length; i++)
      if ( values[i] > values[maxIndex] )
        maxIndex = i ;
    int temp = values[length-1];
    values[length-1] = values[maxIndex];
    values[maxIndex] = temp ;
    sort(values, length - 1);
}
```

Mixed Bag – Q2

- In a binary search of the array below, which elements in the array are checked (and in what order) when a search is conducted for the number 17?

Index	0	1	2	3	4	5	6	7	8
Value	2	9	22	25	47	59	61	66	93

- 47, 9, 22

Mixed Bag – Q3

- Briefly explain what the function does AND what its return value means. (Using the typical tree node struct)

```
int mystery(struct node *root) {
    int retVal;
    if(root == NULL)
        return 0;
    retVal = mystery(root->left) +
             mystery(root->right);
    if(root->data % 2 == 1) {
        root->data -= 1;
        retVal ++;
    }
    return retVal;
```

Mixed Bag – Q3

- The function subtracts 1 from all nodes containing odd values
- The function returns the number of nodes altered by the function (# of odd nodes)

```
int mystery(struct node *root) {
    int retVal;
    if(root == NULL)
        return 0;
    retVal = mystery(root->left) +
             mystery(root->right);
    if(root->data % 2 == 1) {
        root->data -= 1;
        retVal ++;
    }
    return retVal;
}
```

Mixed Bag – Q4

- Imagine using a linked list of digits to store an integer. For example, a list containing 3, 6, 2, and 1, in that order stores the number 3621. Write an iterative function which accepts a linear linked list num that stores a number in this fashion and returns the value of the number. You may assume the list stores digits only and contains 9 or fewer nodes.

```
struct node{
    int data;
    struct node *next;
};

int getValue(struct node* num) {

    // Fill in code

}
```

Mixed Bag – Q4

- Imagine using a linked list of digits to store an integer. For example, a list containing 3, 6, 2, and 1, in that order stores the number 3621. Write an iterative function which accepts a linear linked list num that stores a number in this fashion and returns the value of the number.

```
int getValue(struct node* num) {
    int sum = 0;

    while (num != NULL) {
        sum = 10*sum + num->data;
        num = num->next;
    }

    return sum;
}
```

Mixed Bag – Q5

- What is the Big-O running time of the following segment of code, in terms of ***n***.

```
int i;
for (i=0; i<n; i+=2) {
      for (j=i; j>0; j--)
            printf("%d", j);
      printf("\n");
}
```

- The inner loop will run 0+2+4+…+ n times
- Since we know $0+1+2+3+\ldots+n = n(n+1)/2 = O(n^2)$
- We would have about ½ of $O(n^2)$ = $O(n^2)$