As you arrive, get a packet and prepare to do questions 1-3 of a practice test. Note that parts C & D of question 1 did not get in the packet and are on a separate page.

Here’s my solution to the prep work:

```java
int winAt21(int currentNum) {
    if(currentNum == 20)
        return -1;
    for(int i = 1; i <= 3; i++) {
        if (winAt21(currentNum + i) == -1)
            return i;
    }
    return -1;
}
```
Please work on questions 1-3 of the practice exam. The practice will stop at 10:45.

Note that parts C & D of question 1 did not get in the packet and are on a separate page.

I’ll be going over Q1 C&D, Q2 A, and Q3 A if you’re running out of time at least do those.

Some of these questions may be in your notes, but try to solve them from scratch.
Complete solutions are linked off the syllabus page
Question 1 Parts C & D

• Part C
  
  public int popMin()
  {
    // empty list
    if (start == null) return -1;
    // calculate the min
    int min = peekMin();
    // min at the front of the list
    if (start.value == min)
    {
      start = start.next;
      return min;
    }
    // min not the first node in the list
    Node prev = start;
    Node current = start.next;
    while (current.value != min)
    {
      current = current.next;
      prev = prev.next;
    }
    // delete the node
    prev.next = current.next;
    return min;
  }

• Part D

  7 9 4 8 (make sure you understand why this is the order)
Question 2: Part A

LEFT and RIGHT serve as maps of brace or parens to an integer, providing the same integer value for corresponding braces, e.g., ( maps to 2 and ) maps to 2. Using a stack ensures checking balance since it’s LIFO so () matches since ( is on top of stack when ) is seen. Similarly if we assume [ . . . ] works with . . . checked, then [ matches ] by the LIFO structure.

- You need to explain what’s going on with the stack of integers for full credit
public int groupSize(boolean[][][] grid, int row, int col) {
    if (row < 0 || row >= grid.length || col < 0 || col >= grid[0].length)
        return 0;
    if (!grid[row][col])
        return 0;
    grid[row][col] = false;
    return 1 + groupSize(grid,row-1,col) + groupSize(grid,row+1,col) +
                groupSize(grid,row,col-1) + groupSize(grid,row,col+1);
}
3 Questions You Should Definitely Be Able to Solve for the Exam

1. Questions that require you to manipulate the nodes in a linked list
2. Questions that require you to use stacks and queues
3. Questions that require you to use recursion (& potentially recursive backtracking)

Things not to worry about for the exam: Trees, Priority Queues, Heaps, Doubly Linked Lists, Big O for recursive functions