Problem #1

```
public boolean cigarParty(int cigars, boolean isWeekend) {
    return isWeekend ? cigars >= 40 : (cigars >= 40 && cigars <= 60);
}
```

Problem #2

```
public boolean lessBy10(int a, int b, int c) {
  return Math.abs(a - b) >= 10 ? true : Math.abs(b - c) >= 10 ? true :
Math.abs(a - c) >= 10 ? true: false;
}
```

The first problem was really straightforward, it would return true if the cigars were between 40 and 60 unless isWeekend was true in this case true vas only returned if cigars were more than 40, with no upper limit.

Although this problem was equally as straightforward, all I had to do was compare all values to each other, and if the difference was 10 or greater than 10 return true it took a bit longer because I had to figure out how to use the negative number that could some times appear when getting the difference that's when I remembered that I could get the absolute difference of these values, that would always be a positive number, now I had to figure out how to get absolute numbers in java after some research, I found this link.

Java Math abs() method with Examples - GeeksforGeeks

I found out that java has an absolute method in its math class, after finding this it was easy to just write the rest of my code and test it.

```java
public boolean firstLast6(int[] nums) {
  return nums[0] == 6 || nums[nums.length - 1] == 6;
}
```

```java
public int[] swapEnds(int[] nums) {

  int num = nums[0];
  int temp = nums[nums.length - 1];
  nums[nums.length - 1] = num;
  nums[0] = temp;
  return nums;
}
```

This week i decide to look at problems on coding bat related to arrays to try and get more familiar with arrays in java

The first problem asked me return true if the first item of my array was a 6 or if the last item was a 6, this wasn't a difficult problem to solve the only thing that i found a little difficult was how to specify the last index of the array without knowing its length, but then i remembers that java has the length method and by subtracting 1 from it i would the correct index of my array.

The second problem was a whole other can of worms, this problem asked me to swap the first and last items of an array, at first I was completely stumped on how to do this, but then I remembered that last semester for the C++ course I had to do something similar, so after a couple of minutes of searching in my old projects folder I found the function I needed, all I had to do was adjust it to work with my array which wasn't hard. In the end, this problem wasn't that difficult, but only because I had already done something similar in the past and was able to apply that knowledge to this current project.

```java
public int withoutDoubles(int die1, int die2, boolean noDoubles) {
  if(noDoubles){
    if(die1 == die2){
      die1++;
      if(die1 > 6){
        die1 = 1;
      }
    }
  }
  return die1 + die2;
}
```

```java
public String alarmClock(int day, boolean vacation) {
  if(vacation || (day == 0 || day == 6)){
    if(vacation && (day == 0 || day == 6)){
      return "off";
    }
    return "10:00";
  }
    return "7:00";
}
```

This week our code review had to be on Logic-1 in codingbat, the problem i choose at random seeing what name popped out at me, withouDoubles seemd intriguing so gave it a shot, in summary the program added up two six sided dice if both die where equal it would have to increment one of them by one, wrapping around to one if the die being incremented was already six, the one thing i thought to be challenging was figuring out how to wrap around from six back to one, but after some thought i figured that if i incremented the die by one and that made it larger than six i could just set it back to one. After doing this and testing the code it worked so i moved on to the second challenge.

To find the second challenge i decide to click the chance link on the top of the page, the first problem to appear was the alarm problem, this one wasn't as challenging as the on before, if the int day was a weekday 1-5 then the program would return "7:00", but if it was a weekend or vacation was true then it would return "10:00", and the final condition was if vacation was true and it was a weekend then it would return "off", i accomplished this by first cheking if vacation was true or if it was weekend end, if this condition was met i would then check to see if both were true if they were i would return "off", but if only one was then i would return "10:00", and if none of them were i would return "7:00".

```java
public String conCat(String a, String b) {
  if(!a.isEmpty() && !b. isEmpty()){
  if(a.charAt(a.length()-1) == b.charAt(0)){
      return a.concat(b.substring(1));
    }
  }
    return a.concat(b);
}
```

```java
public String left2(String str) {

    String[] b = {str.substring(0,2), str.substring(2)};
    String temp = b[0];
    b[0] = b[1];
    b[1] = temp;
    str = String.join("", b);
    return str;
}
```

This week we used problems from String-1 for our coding review, these problems all involved working and manipulating string, as the name states. Both problems I worked on were each challenging in their own way.

The First problem asks to take to string concatenate them, but if a double character is formed because of this, then it would have to remove one character from any of the two strings before concatenating them to check if a double character would be produced I would first compare the last character of the first string with the first character of the second string if they were the same I would concatenate to the first string a substring of the second with the first character removed. Finally, if any of the strings were empty, I would directly concatenate both strings and return that answer.

The second question was a bit tricky. It asked me to place the first two letters of a string at the end of said string. To accomplish this, I first divided my word into an array where the first element was the first two letters of the word and the second element was the rest of the word; I then proceeded to switch the element's places and join them back into a single word.

```java
class TriangleTester{
  public static boolean isTriangle(int a, int b, int c){
    if(((a+b) > c) && ((b+c) > a) &&((c+a) > b)){
      return true;
    }
    return false;
  }
}
```

```java
public class Kata {

  public static boolean feast(String beast, String dish) {

    return (beast.charAt(0) == dish.charAt(0)) &&
(beast.charAt(beast.length()-1)==dish.charAt(dish.length()-1));
  }

}
```

For this week we had to solve two problems from codewars, the problem i ended up solving werent particularly challenging, but each had their own charm.

The first problem asked to return true if its posible to form a triangle the integers it provided, at first i was confused on how to do this, but after some research i found the triangle inequality theorem, according to this theorem, for any triangle, the sum of lengths of two sides is always greater than the third side, by using this theorem, aslong as the sum of any two sides wasnt shorter than the remaining side a triangle is posible.

The second problem said that a beast had to bring a dish to a feast, the dish has start and end with the same character as the beasts name, this one was even easier. All i had to do was compare both the first and last character in both strings if they were equal then the beast can bring that dish, else the dish wasnt valid.

```java
public class Prime {
  public static boolean isPrime(int num) {
    if(num <= 1) return false;
    for(int i = 2; i <= num/2; i++){
      if(num % i == 0){
        return false;
      }
    }
    return true;
  }
}
```

```java
public static int countBits(int n){
        Stream<Character> st = Integer.toBinaryString(n).chars().mapToObj(i ->
(char)i);
        return st.filter(i -> i.equals('1')).mapToInt(i ->
Character.getNumericValue(i)).sum();
    }
```

For this code review i decided to do two Rank Up level challenges, this means that i completed challenges at or above my level in code wars.

The first challenge was to detect if a number was prime; the function would take in an int and then return a boolean answer. To start, I first had to figure out what exactly was a prime number; in essence, a prime number is only divisible by 1 and itself, so for starters, I would check to see if a number was less than or equal to 1 if so I would say that wasn't a prime number and return false if it was greater than one then I would iterate trough every number in between 2 and half of the original number(because the highest factor of a number is half of itself), if the original can be divided by that iteration and have 0 as the remainder then it wasn't a prime number, and I would return false, if by end of the sequence I hadn’t return false then the number was prime, and I could return true.

The next problem has been one of the most fun problems I've solved in a while, the core of the problem was to figure out how many 1’s were in the binary representation of a number to do this, I first took the number and created a Stream of chars based on the binary representation of the number, then i filtered these chars to leave only the ones, finally, I mapped the chars to ints and added them up, and that was my answer at first i tried doing this with an array and for loop, but after getting it working i knew i could do it fewer lines of could so i switched to stream, which i find to be very similar to ArrayList but with more methods, in the end, i went from about 10 lines of code to only 2, but once i submitted and saw the first answer, i facepalmed on how simple the answer was, turners out javas Interger object has a method for precisely the problem that i had to solve, so i could have done it in one line of code `Interger.bitCount(n);` that method specifically returns the number of 1’s in the binary representation of an int, which honestly just leads me to ask how often does this particular problem need to be solved for oracle to add this method to base java, just makes me laugh.

https://codingbat.com/done?user=jorge.fuerte001@mymdc.net&tag=6721633611

https://www.codewars.com/users/Jorjo_

The progress I have achieved isn't precisely what I would like, but I haven't had much time to sit and solve problems. I have been solving problems as they come, but I am still pleased with how much java I am assimilating. Java was one of the first languages I learned, but I felt like it never really stuck over the years. Still, thanks to this class, I feel competent at solving problems with java without needing to google every solution.