CS200: Programming I **Instructor:** Sam Vinitsky

P6: ReCURSE You!!!

(25 points)

Due: Monday, July 15 @ 11:59pm

Part 0: Introduction

Brrrring brriiiiiing the phone rings – it's Rose, local ghost! Rose has been dead for 10 years, and haunts a local astronomer name Dave, who wronged her while she was alive. Dave, who cannot see Rose's ghost, is a very stoic man, and it takes a lot to spook him. Rose needs your help to take her hauntings to the next level! She has asked that you write a series of programs to help her determine the best ways to haunt Dave, and enact vengeance for the years of suffering he inflicted on her.

Since she is stuck in an endless loop of haunting, **Rose is afraid of loops**. As such, your code **must not include any loops**. All of these problems can be solved recursively, using only conditionals and function calls. As usual, you may not use anything we haven't covered in class, such as: arrays, ArrayLists, String parsing, the "Math" library, etc. If you have questions about what you are/not allowed to use, please ask on Piazza.

Each program you turn in should include a comment at the top with (1) your full name, (2) your student ID number, (3) your netID, and (4) the name of anyone you discussed the homework with (excluding Sam and Alex).

As always: **start early, ask questions, and have fun!**

Part 1: Moving the honey jar [4 points]

Dave the astronomer loves honey. It's the only thing he eats, and he has a large jar that he eats from throughout the day. Rose likes to move the honey jar around, just to mess with him. Unfortunately, Dave has gotten used to it mysteriously moving once a day, so he is no longer bothered by it. As such, Rose would like to change the **number of times** that she moves the jar each day according to a strange pattern that will be hard for Dave to predict. She hopes that if she changes up the **number of times** she moves the honey jar, she will be able to surprise him and make him uncomfortable. The pattern she will use is:

- On day 1, she will move the jar twice.
- On day `n`, the number of times she will move the jar will be `2*n` greater than the number of times she moved it the previous day.[1]

Write a program called `Honey.java` that contains a function `numMoves` that takes as input a single integer `n`, and returns an integer representing the number of times Rose will move the jar on day `n`. **This function must be recursive, and cannot use any loops. If you use loops, you will receive a 0.** The signature of the function must be:

```
public static int numMoves(int n)
```

Make sure your function produces reasonable results for every `n` from 1 to 30. It is okay if your function is slow for larger numbers. Below is a **small sample** of the outputs you should expect from your function:

`numMoves(2)` should return `6`
`numMoves(3)` should return `12`
`numMoves(4)` should return `20`
`numMoves(5)` should return `30`
`numMoves(6)` should return `42`
`numMoves(7)` should return `56`
`numMoves(8)` should return `72`
`numMoves(9)` should return `90`
`numMoves(10)` should return `110`

Note: This function should not *print* anything, just *return* a value...

Part 2: A Higher Power [5 points]

Rose's strength[2] is growing! On the first day, her strength was a 2 and every day her power **doubles**. She hopes that one day her strength will be so great, that she can take over the entire astronomy department!!! Rose is impatient, and wants to know **when** she will be strong enough to do this. That's where you come in!

She needs you to write a program called `Power.java` that contains a single function named `power`. This function should take as input an int `day` representing the day, and returns an int equal to Rose's power on day number `day`. **You may not use loops**. The function signature must be:

```
public static int power(int day)
```

[1] This only holds when `n>1`
[2] Measured on a scale from 1 to 2147483647

Make sure your function produces reasonable results for every `day` from 1 to 30. It is okay if your function is slow for larger numbers. Below is a **small sample** of the outputs you should expect from your function:

`power(2)` should return `4`
`power(3)` should return `8`
`power(4)` should return `16`
`power(5)` should return `32`
`power(8)` should return `256`

Note: This function should not *print* anything, just *return* a value...

Part 3: The Telescope [6 points]

As mentioned above, Dave is an amateur astronomer. To mess with him, Rose swivels his telescope by a small number of degrees each day, so he has to re-adjust it to find what he was looking at. As before, she wants her swivels to follow a complicated pattern, so he is always surprised. The number of degrees she swivels the telescope by each day is determined by the following pattern:

- On day 1, she will swivel it 1 degree
- On day 2, she will swivel it 1 degree
- On day 3, she will swivel it 3 degrees
- On day `n`, she will swivel it a number of degrees equal to the sum of the number of degrees on the **previous** day and the number of degrees **three** days ago, plus 1.[3]

In a program called `Telescope.java`, write a function called `degrees` that takes an integer representing the day as input, and outputs an integer representing the number of degrees to swivel it on that day. **You may not use loops.** The function signature must be:

```
public static int degrees(int day)
```

Make sure your function produces reasonable results for every `day` from 1 to 30. It is okay if your function is slow for larger numbers. Below is a **small sample** of the outputs you should expect from your function:

`degrees(4)` should return `5`
`degrees(5)` should return `7`

[3] This only holds when `n>3`

`degrees(6)` should return `11`
`degrees(7)` should return `17`
`degrees(8)` should return `25`
`degrees(9)` should return `37`
`degrees(10)` should return `55`

Note: This function should not *print* anything, just *return* a value...

Part 4: Star ID's [7 points total]

Dave loves stars, which is why he became an astronomer. He spends his days recording the positions of a variety of stars in the sky. Each star is represented in his notes by an ID number. The ID numbers obey the following rules:

- Every `7` in the ID must be immediately followed by either a `6` or a `2`.
- Any `1` in the ID may **not** be immediately followed by a `7`.
- The digit `3` can never appear in the ID.

For example, `7655` would be a valid ID of length 4, but `7566` would not be (since the `7` is not immediately followed by a `2` or `6`). `666` would be a valid ID of length 3, but `333` would not be (since no `3`'s are allowed). `02102` would be a valid ID of length 5, but `02172` would not be (since `1` may not be followed by a `7`).

a) Counting the ID's (5 points)

Every day, Rose would like to modify the ID of the star Dave is currently tracking in his astronomy journal, just to mess with him. However, she is unsure as to how many days she will be able to do this (i.e. how many star ID's there are). Write a program called `IDs.java`. Include a **recursive** function called `countIDs` that takes as input an integer representing the length of the ID's, and returns **the number of ID's** there are of the given length that follow this rule. **You may not use loops.** The signature of your function must be as follows:

```
public static int countIDs(int numDigits)
```

Make sure your function produces reasonable results for every `numDigits` between 1 and 10. It is okay if your function is slow for larger numbers. Below is a **small sample** of the outputs you should expect from your function:

`countIDs(2)` should return `66`
`countIDs(3)` should return `542`
`countIDs(4)` should return `4454`
`countIDs(5)` should return `36598`

Hint: *This is very similar to a problem we did in class on Wednesday, July 10. I would recommend proceeding the same way: count the number of ID's that start with a 7, count the number that start with a 1, and count all the others. Then, add together these amounts. Each of the sub-counts will involve a recursive call.*

Hint 2: *As I mentioned in class, there is a single string of length 0... This should be one of your base cases...*

Note: This function should not *print* anything, just *return* a value...

b) Printing out (2 points)

In main, print out the number of IDs of each length from 1 to 10, formatted as follows:[4]

```
The number of 1-digit IDs is: 8
The number of 2-digit IDs is: 66
..
The number of 10-digit IDs is: 1370951286
```

Hint: *In order to print this out, you will need to call* `countIDs` *repeatedly from main...*

Part ∞: Feedback Form [3 points]

Fill out this feedback form **after you submit** this assignment. Completion of this will count towards your grade, but your responses themselves will not affect your grade in any way (so be honest!).

[4] Do not actually print the ellipses, those are just to indicate that there are numbers cut out...

What to turn in:

On Canvas, turn in a zip folder named `<your_net_id>_P6.zip` that contains the following files:

- `Honey.java` [4 points]
  - `int numMoves(int n)`
- `Power.java` [5 points]
  - `int power(int day)`
- `Telescope.java` [6 points]
  - `int degrees(int day)`
- `IDs.java` [7 points]
  - `int countIDs(int numDigits)`

Also complete the feedback form. [3 points]