2010Fa CS10 Online Final

Section 1

Instructions:
Save the file containing your answers with the name **`FinalYourfirstnameYourlastname.ypr`** (e.g., **`FinalBarackObama.ypr`**). You can assume that all the inputs that your program will ever be given are valid, so you do not have to perform any error checking. You will submit your solution on bSpace. When you are done, go to the `Assignments` tab and click on the `Online Final` assignment corresponding to your lab section. Upload your file, and *do not forget to click* `Submit`!

Take a deep breath and calm down before moving on: this exam is not worth having heart failure about. Good luck!

1. Create the block

   

   that takes in a minimum and a maximum, and reports a list of numbers from the minimum to the maximum (both inclusive). So, for example,

   

   would report the list

   

   You can approach this question using any of the techniques that you have learned in this class.

2. Create the block

   

   that takes in a word and reports a list of all of the characters in that word. So, for example,

   

   would report the list

   

   Again, you can approach this question using any of the techniques that you have learned in this class.

3. A number is called *pandigital* if it uses all of the digits from 1 to *n exactly once*, where *n* is the number of digits in the number. So, for example, the five-digit number 15432 is pandigital, but the five-digit number 11132 is not. Again, the four-digit number 4123 is pandigital, but the four-digit number 8312 is not. Finally, the one-digit number 1 is pandigital, but the one-digit number 3 is not.

   Create a predicate block

that checks if a number satisfies the definition above. The block should, for example, report `true` for the following:

and `false` for the following:

You should use higher-order functions in your solution. You cannot use explicit recursion or the yellow and orange loop constructs (such as `repeat`, `repeat until`, and `for`); you can, however, use the built-in list blocks in the `Variables` tab and the HOF blocks from ToolSprite. You will find the blocks that you created in earlier questions useful.

4. Using your block from question 3, have Alonzo say all of the pandigital numbers between 1 and 135 in a comma-separated sentence when the green flag is clicked. There are *five* pandigital numbers between 1 and 135, so Alonzo should say “1, 12, 21, 123, 132” when the green flag is clicked. This may take 1 - 2 minutes to compute, so don’t worry if Alonzo doesn’t speak up right away.

   You should use higher-order functions in your solution. You cannot use explicit recursion or the yellow and orange loop constructs (such as `repeat`, `repeat until`, and `for`); you can, however, use the built-in list blocks in the `Variables` tab and the HOF blocks from ToolSprite. You will find the blocks that you created in earlier questions useful.

2010Fa CS10 Online Final

Section 2

Instructions:
Save the file containing your answers with the name **`FinalYourfirstnameYourlastname.ypr`** (e.g., **`FinalBarackObama.ypr`**). You can assume that all the inputs that your program will ever be given are valid, so you do not have to perform any error checking. You will submit your solution on bSpace. When you are done, go to the `Assignments` tab and click on the `Online Final` assignment corresponding to your lab section. Upload your file, and *do not forget to click* `Submit`!

Take a deep breath and calm down before moving on: this exam is not worth having heart failure about. Good luck!

1. Create the block

   

   that takes in a minimum and a maximum, and reports a list of numbers from the minimum to the maximum (both inclusive). So, for example,

   

   would report the list

   

   You can approach this question using any of the techniques that you have learned in this class.

2. Create the block

   

   that takes in a word and reports a list of all of the characters in that word. So, for example,

   

   would report the list

   

   Again, you can approach this question using any of the techniques that you have learned in this class.

3. We will call a number *narcissistic* if it satisfies the following property: if the number has *n* digits, then it should be the sum of all of its digits, each raised to the *n*th power. So, for example, the number 153 is narcissistic, because 153 is a 3-digit number, and if you were to raise each digit to the third power -- 1^3, 5^3, 3^3 -- and add them, then you would get 153 back. In other words, $153 = 1^3 + 5^3 + 3^3$. 1634 is also a narcissistic number, since $1634 = 1^4 + 6^4 + 3^4 + 4^4$.

Create a predicate block

that checks if a number satisfies the definition above. The block should, for example, report `true` for the following:

You should use higher-order functions in your solution. You cannot use explicit recursion or the yellow and orange loop constructs (such as `repeat`, `repeat until`, and `for`); you can, however, use the built-in list blocks in the `Variables` tab and the HOF blocks from ToolSprite. You will find the blocks that you created in earlier questions useful.

You may also find it useful to write the block

that raises the first input to the second input (where both inputs are always positive integers). However, the same restrictions apply for this block: **you should use higher-order functions in your solution. You cannot use explicit recursion or the yellow and orange loop constructs (such as `repeat`, `repeat until`, and `for`); you can, however, use the built-in list blocks in the `Variables` tab and the HOF blocks from ToolSprite.** You will find the blocks that you created in earlier questions useful.

4. Using your block from question 3, have Alonzo say all of the narcissistic numbers between 1 and 160 in a comma-separated sentence when the green flag is clicked. There are *ten* narcissistic numbers between 1 and 160, so Alonzo should say "1, 2, 3, 4, 5, 6, 7, 8, 9, 153" when the green flag is clicked. This may take 1 - 2 minutes to compute, so don't worry if Alonzo doesn't speak up right away.

   You should use higher-order functions in your solution. You cannot use explicit recursion or the yellow and orange loop constructs (such as `repeat`, `repeat until`, and `for`); you can, however, use the built-in list blocks in the `Variables` tab. You will find the blocks that you created in earlier questions useful.

2010Fa CS10 Online Final

Section 3

Instructions:
Save the file containing your answers with the name **`FinalYourfirstnameYourlastname.ypr`** (e.g., **`FinalBarackObama.ypr`**). You can assume that all the inputs that your program will ever be given are valid, so you do not have to perform any error checking. You will submit your solution on bSpace. When you are done, go to the `Assignments` tab and click on the `Online Final` assignment corresponding to your lab section. Upload your file, and *do not forget to click* `Submit`!

Take a deep breath and calm down before moving on: this exam is not worth having heart failure about. Good luck!

1. Create the block

   

   that takes in a minimum and a maximum, and reports a list of numbers from the minimum to the maximum (both inclusive). So, for example,

   

   would report the list

   

   You can approach this question using any of the techniques that you have learned in this class.

2. Create the block

   

   that takes in a word and reports a list of all of the characters in that word. So, for example,

   

   would report the list

   

   Again, you can approach this question using any of the techniques that you have learned in this class.

3. A number is called a *factorion* if it is equal to the sum of the factorials of its digits. For example, 145 is a factorion since 145 = 1! + 4! + 5! (Remember that *n*! is the fancy, mathematical way of denoting the factorial of *n*, which is the product of all of the numbers from 1 to the number *n*.) Another, relatively simpler example of a factorion is 2, since 2 is equal to 2! (We're not exclaiming there.)

Create a predicate block

that checks if a number is a factorion. The block should, for example, report `true` for the following:

You should use higher-order functions in your solution. You cannot use either explicit recursion, or the yellow and orange loop constructs (such as `repeat`, `repeat until`, and `for`); you can, however, use the built-in list blocks in the `Variables` tab and the HOF blocks from ToolSprite. You will find the blocks that you created in earlier questions useful.

You may also find it useful to write the block

that finds the factorial of its input. However, the same restrictions apply for this block: **you should use higher-order functions in your solution. You cannot use explicit recursion or the yellow and orange loop constructs (such as `repeat`, `repeat until`, and `for`); you can, however, use the built-in list blocks in the `Variables` tab and the HOF blocks from ToolSprite.** You will find the blocks that you created in earlier questions useful.

4. Using your block from question 3, have Alonzo say all of the factorions between 1 and 150 in a comma-separated sentence when the green flag is clicked. There are *three* factorions between 1 and 150, so Alonzo should say "1, 2, 145" when the green flag is clicked. This may take 1 - 2 minutes to compute, so don't worry if Alonzo doesn't speak up right away.

   You should use higher-order functions in your solution. You cannot use explicit recursion or the yellow and orange loop constructs (such as `repeat`, `repeat until`, and `for`); you can, however, use the built-in list blocks in the `Variables` tab and the HOF blocks from ToolSprite. You will find the blocks that you created in earlier questions useful.