CSCI A201/A597 - Introduction to Programming I

Fall 2017

Homework 14

Due Date

- Thursday, November 30, 11:59pm on IU Canvas.

Work Policy

Homework for A201/A597 has to be completed individually: no group solutions or cooperative work. You may discuss the homework with other A201/A597 students, but you have to write the solutions and implement programming answers by yourself. If you discuss the homework with other A201/A597 students (or anyone else, for that matter!), you must acknowledge them by including their names in a comment at the top of the Python files you turn in for this homework. **Do not share any source code or homework answers with any students**.

Homework Tasks

[total: 100 points]

Topics: refer to Reading Assignment 14.

When you've completed and run your programming tasks on IDLE, save a transcript of your IDLE session to a file named `hw14-transcript.py`, and include that with your submission. Make sure that your submitted IDLE transcript includes interactions with running *all* of your homework scripts, otherwise your homework will be rejected.

Note about recursive functions: Any time we ask you to write a recursive function in this class, it must be a ***pure*** function with no side effects, meaning that the only result of the function is the returned value. In particular, you can't use `print()` or `input()`, you can't work with files, and you can't modify the arguments of the function.

Also, you are *not* allowed to use `for` or `while` loops on any of the tasks on this assignment. There are also some more specific requirements for certain tasks, as listed below, in order to use recursion. (yes, some of these tasks *could* be solved using other techniques, but then you wouldn't be learning about recursion, so follow the instructions).

Save all of the definitions you write for tasks B-D into a script called `hw14-recursive.py`.

A. Consider the following function definitions. For all of them, the signature is `Natural-Number -> Natural-Number` (where a *natural number* is any positive integer or zero).

```
def recOne(x):
    return recOne(x)

def recTwo(x):
    return recTwo(x-1)

def recThree(x):
    if x == 0:
        return recThree(x-1)
    else:
        return recThree(x-1)

def recFour(x):
    if x == 0:
        return 0
    else:
        return recFour(x-1)

def recFive(x):
    if x == 0:
        return 0
    else:
        return recFive(x-2)

def recSix(x):
    if x == 0:
        return 0
    else:
        return 2 + recSix(x-1)

def recSeven(x):
    if x == 0:
        return 0
    else:
        return x + recSeven(x-1)
```

For each of these functions, answer the following questions. Put all your answers into a text file called `hw14-answers.txt`.

1. What happens when you run the function with the input 4?
2. If it crashes, explain why. If it doesn't crash, how many times was the function called?
3. For which values will the function return an answer, and for which will it crash? For these functions, you should only worry about natural numbers: 0, 1, 2, 3, ... (Don't worry about negative numbers, non-integers, or things that aren't numbers.)
4. If there are values that the function won't crash for, explain what the output will be in all those cases.

Note: if you run the above examples in IDLE on a *macOS* system, it may take a while to run a function until it shows that it *"crashed"*. For example, running on recent hardware, one of the functions above may run for up to 50 seconds before IDLE shows that it crashed. During that time, you may think that the function is *"stuck"* in an infinite loop --- not even `Control`+`C` will interrupt it, since the delay is caused by IDLE very slow printing, not by Python being *"stuck"*. Please be patient.
If you don't want to wait that long, you can avoid slowdowns caused by the IDLE environment and test the same code directly in the Python command-line environment. For example, if you saved the above code in a file named `hw14-testing.py` on the "Desktop" on your computer system, you may try the following (these instructions are for *macOS* systems):
Start the *Terminal* (you can find *Terminal.app* in the *Applications→Utilities* folder on any computer running *macOS*), and in the *Terminal* window type `cd ~/Desktop; python3 -i hw14-testing.py` and press the `Return` key. In doing so, you will have a similar Python environment as in IDLE, with your `hw14-testing.py` code already executed. You may then try some of the functions above, e.g. try typing `recSeven(4)`, etc. You can exit the Python command-line environment by pressing `Control`+`D`, at which point you can either quit *Terminal.app*, or test your code again with `cd ~/Desktop; python3 -i hw14-testing.py`, etc.
If you are using IDLE on Windows or Linux, or if you're willing to wait a while, then you don't need to use the Python command-line environment like this.

B. Write a recursive function called `fastGrowing()` that takes a natural number and returns the product of all the natural numbers from 1 up to the given number. So `fastGrowing(5)` should return `120` (because $5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$). Remember that you can't use any loops anywhere in this assignment.

C. Write a function called `power_of_three()` that takes a natural number n and returns 3^n. So `power_of_three(5)` should return `243`. You are not allowed to use `**` or `pow()` for this task. In fact, the only math* operators or functions you need are multiplication `*`, subtraction `-`, and equality testing `==`.

*Of course, feel free to use non-math keywords like `def`, `if`, `else`, and `return`. (Hint: those are the *only* keywords you need for tasks C and D.)

D. Write a function called `power()` that takes a number *b* and a natural number *n* and returns b^n. So `power(3,4)` should return `81`. You are not allowed to use `**` or `pow()` for this task. In fact, the only math* operations or functions you need are multiplication `*`, subtraction `-`, and equality testing `==`.
*Of course, feel free to use non-math keywords like `def`, `if`, `else`, and `return`. (Hint: those are the *only* keywords you need for tasks C and D.)

Homework 14 Submission Instructions:

1. Your submission has to contain the files as listed above:
   `hw14-transcript.py`,
   `hw14-answers.txt`, and
   `hw14-recursive.py`.
   1. At the top of your files, include "*A201 / Fall 2017*" (or "*A597 / Fall 2017*"), "*Homework 14*", your *full name* and your *IU username* (i.e. your IU account) *in* the homework text. Note: in your Python files (file extension .py), kindly include this information at the top of the file, within Python `# comments`.
   2. Make sure that your plain-text documents (including all your `.py` files) are all clearly readable by anyone; plain-text documents need to use either:
      - ASCII encoding (7-bit, allows no accents/smart quotes/etc.), ← preferred for Python source code submissions that only contain English-alphabet characters & symbols.
      - UTF-8 encoding (capable of encoding all sorts of characters) ← only for any submissions that *require* non-English-alphabet characters & symbols.

      (...otherwise your output may end up garbled...)
   3. Do not use any other document file formats: submitted files *other* than plain text (file formats such as .pdf, .doc, .docx, .html, .xml, etc...) will be *rejected*. In other words, all submitted `.txt` and `.py` files need to be plain text.
2. Turn in your Homework 14 files by 11:59PM on Thursday, November 30 2017, on the IU Canvas *A201/A597 Fall 2017* page.
3. When you turn in your homework on IU Canvas, it is your responsibility to verify that your files have been uploaded, and that their content is visible on the IU Canvas server.
4. Please do not omit your IU username from the content.

Last updated: November 29, 2017
mitja⊛indiana·edu