Due: Wednesday Sep. 27th, 2017 at 10 am

Assignment Guidelines

- This assignment covers material in Module 02.
- Submission details:
  - Solutions to these questions must be placed in files `a02q1.py`, `a02q2.py`, `a02q3.py`, and `a02q4.py`.
  - You must be using Python 3 or higher. Do NOT use Python 2.
  - Download the interface file from the course Web page to ensure that all function names are spelled correctly and each function has the correct number and order of parameters.
  - All solutions must be submitted to MarkUs. No solutions will be accepted through email, even if you are having issues with MarkUs.
  - Verify using MarkUs and your basic test results that your files were properly submitted and are readable on MarkUs.
  - For full style marks, your program must follow the Python section of the CS116 Style Guide.
  - Be sure to review the Academic Integrity policy on the Assignments page
- Download the testing module from the course web page. Include `import check` in each solution file.
  - When a function produces a floating point value, you must use `check.within` for your testing. Unless told otherwise, you may use a tolerance of 0.001 in your tests.
- Restrictions:
  - Do not import any modules other than `math` and `check`.
  - Do not use Python constructs from later modules (e.g. loops and lists). Do not use any other Python functions not discussed in class or explicitly allowed elsewhere. See the allowable functions post on Piazza. You are always allowed to define your own helper functions, as long as they meet the assignment restrictions.
  - While you may use global *constants* in your solutions, do **not** use global *variables* for anything other than testing.
  - Read each question carefully for additional restrictions.

The solutions you submit must be entirely your own work. Do not look up either full or partial solutions on the Internet or in printed sources.

1. Write a Python function `parking_costs` that calculates and returns the cost of parking at an airport parking lot as a natural number.
   For short-term parking (less than one day), the cost is $4 for each 20 minutes used (full or partial), to a maximum of $28 for a day.
   For long-term parking, the cost is $28 per day, to a maximum of $120 for a week. Each additional week is charged the same way ($28 per day to a maximum of $120 for the week). Prices include all taxes.
   The function `parking_costs` consumes two parameters:
   i. `short_term`, a boolean that is `True` if you are calculating short-term parking costs and `False` if you are calculating a long-term cost; and
   ii. `time` (a natural number), the length of time your car is parked (if `short_term` is `True`, `time` is measured in full minutes, but if `short_term` is `False`, `time` is measured in full days).

   For example,
   - `parking_costs(True, 47) =>12`
   - `parking_costs(True, 500) =>28`
   - `parking_costs(False, 6) =>120`
   - `parking_costs(False, 29) =>508`

2. Write a Python function `interest_rate` that consumes for a user's account: the current balance (float number), the type of the account ("Personal" or "Business"), and the level ("Standard", "Gold", or "Platinum"). [Check the provided interface for parameter names]. Based on this information and the rate table below, the function returns the interest rate the user is receiving. Note that if the consumed balance is less than the minimum required then the Interest rate is 0. Also you may assume that the level is valid depending on the consumed type of account.

Type of account	Level	Minimum balance	Interest rate
Personal	Standard	0	1.2
Personal	Gold	2000 (and less than 5000)	1.9
Personal	Gold	5000	2.3
Business	Standard	2500	1.75
Business	Platinum	10000	2.6

   For example,
   - `interest_rate(4500, "Personal", "Gold") => 1.9`
   - `interest_rate(500.56, "Personal", "Gold") => 0`
   - `interest_rate(-6000, "Business", "Standard") => 0`

3. In the following table, an X indicates whether there is a direct transportation (one bus) between 7 sites in the region (named `"A"` … `"G"`). For example, there is a direct transportation from site `"C"` to site `"D"` where there is no direct transportation between `"C"` and `"F"`.

From / To	"A"	"B"	"C"	"D"	"E"	"F"	"G"
"A"	X						
"B"	X	X					
"C"	X		X				
"D"	X	X	X	X			
"E"	X			X	X		
"F"	X	X			X	X	
"G"	X	X	X	X	X	X	X

Write the function `direct_transportation` which consumes two strings (`orig` and `dest` representing two sites). The function returns `"Yes"` if there is a direct transportation from `orig` to `dest`, according to the table above, and `"No"` otherwise. For testing, for this question only, you may use white box testing rather than black box testing.

For example:

- `direct_transportation("F","F") => "Yes"`
- `direct_transportation("D","E") => "Yes"`
- `direct_transportation("F","A") => "No"`

4. Write a Python function `remainder`, that consumes `a` and `b` (two positive natural numbers) and returns the value of `a%b`. The catch? To receive any correctness marks for this question, you cannot use the (`%`) operator, and you must use recursion to perform repeated operation to calculate the result. The only allowed operations are `<, >` and − (You may NOT use `//`)

For example,

- `remainder(3,2) => 1`
- `remainder(164,10) => 4`
- `remainder(1,1) => 0`

As mentioned in class, there is a limit to how many times a Python function can recurse. For that reason, you may assume that (a//b < 1000).