Compsci 101: Test 1 Practice

September 26, 2012

Name: _____

NetID/Login: _____

Community Standard Acknowledgment (signature) _____

	value	grade
Problem 1	12 pts.	
Problem 2	12 pts.	
Problem 3	12 pts.	

In writing code you do not need to worry about specifying the proper `import statements`. Don't worry about getting function or method names exactly right. Assume that all libraries and packages we've discussed are imported in any code you write.

PROBLEM 1 : (*Play that Funky Music*)

Part A

A number is *abundant* if it is greater than the sum of its proper divisors, that is its divisors other than itself. For example 12 is abundant because $1 + 2 + 4 + 6 = 13 > 12$. The first 10 abundant numbers are 12, 18, 20, 24, 30, 36, 40, 42, 48, 54.

Write a boolean function `is_abundant` to return True if its parameter is abundant and False otherwise.

call	return value
`is_abundant(4)`	`False`
`is_abundant(12)`	`True`
`is_abundant(24)`	`True`
`is_abundant(28)`	`False`

```
def is_abundant(num):
    """
    return True if int parameter num is abundant and
    returns False otherwise
    """
```

Part C Write a function `abundant_count` that returns the number of abundant numbers between (and including) parameters first and last. You should call `is_abundant` and assume it works correctly.

call	**return value**
`abundant_count(1,11)`	0
`abundant_count(1,20)`	3
`abundant_count(20,30)`	3
`abundant_count(70,80)`	4

Part C

Sometimes the *beginning* of each word in a list of words combine to make something like an acronym for the list. For example, if we take the first part of each word up to the first vowel and including one letter after the first vowel in each word of a group of words we can form a word from a phrase. For example the phrase *probe lemming atrophy icing* leads to prob + lem + at + ic = `"problematic"`.

Write the function `combine` that creates a string from each of the strings in parameter `phrase`, a list of strings. In writing `combine` you can call the function `firstVowelIndex` given below that returns the index of the first vowel in a string. Every string in the list `phrase` will contain a vowel and the first vowel in each string will **not** be the last character of the string

call	**return value**
`combine(["money", "itch", "orange"])`	`"monitor"`
`combine(["creamery", "session"])`	`"creases"`

```
def firstVowelIndex(word):
    vow = "aeiou"
    for i,ch in enumerate(word):
        if vow.find(ch) >= 0:
            return i
    return len(word)


def combine(phrase):
    """
    return string formed by combining strings in the list
    phrase, each string contributes at least two letters, from the
    beginning of the string up to and including one letter after the first vowel.
    """
```

PROBLEM 2 : (***Phunkadelic (12 points)***)

Part A

A number is *square free* if it is not divisible by any perfect square greater than one. For example, 10 is square free since it is not divisible by four nor by nine, the two perfect squares less than 10. The number 100 is **not square free** since it is divisible by 25, and 25 is a perfect square and by 4 which is also a perfect square. Write the function `isSquareFree` to return True if its int parameter is square free, and false otherwise. For example:

call	**return value**
`isSquareFree(8)`	`False`
`isSquareFree(45)`	`False`
`isSquareFree(38)`	`True`
`isSquareFree(55)`	`True`

Hint: if you loop over 1,2,3,4, ... you can test divisors 1,4,9,16, ... by squaring each of the 1,2,3,4 being looped over.

```
def isSquareFree(num):
    """
    return True if int parameter num is square free and
    returns False otherwise
    """
```

Part B

Some words contain other words. For example, each of `"sublime"`, `"compliment"`, `"limerick"` and `"millimeter"` contains the word `"lime"`. Write the function `wordCount` that returns the number of strings in its list parameter `words` that contain the string `sub`.

call	**return value**
`wordCount(["sublime", "millimeter", "lemon"],"lime")`	2
`wordCount(["subtract", "assume", "consumer","presume","lime"],"sum")`	3
`wordCount(["apple", "banana", "lemon"],"meat")`	0

```
def wordCount(words,sub):
    """
    return the number of strings in string list words
    that contain string sub
    """
```

PROBLEM 3 : (*Genus, Order, Class, ...*)

Data is stored in a file in the format shown below. Each line contains data for one animal giving the animal's name (string), gestation period in days (int), and estimated longevity in years (int). The information on a line is delimited by commas as shown, for example the file below shows information for eight animals in the format used in this problem.

```
bear,180,15
cat,52,10
dog,53,10
hamster,15,2
elephant,510,30
hippopotamus,220,30
human,253,65
lion,106,10
```

Write the function `getAgeList` that returns a list of those animals whose estimated longevity is between the values given by its two int parameters: `low` and `high`. The name of the file holding the data to be read and processed is given by parameter `filename`.

For example, if `"data.txt"` is the name of the sample data file above, then the call `getAgeList("data.txt",15,30)` should return the list `["bear","elephant","hippopotamus"]`, the call `getAgeList("data.txt",1,8)` should return the list `["hamster"]` and the call `getAgeList("data.txt",70,100)` should return the empty list `[]`

```
def getAgeList(filename, low, high):
    file = open(filename);









    file.close()
```