CS 110 – Creative Problem Solving in Computer Science

Stevens Institute of Technology © 2016

Practice Exercises for Exam 3

Instructor: Adriana Compagnoni

Fall 2016

These are representative examples of the kind of exercises you might encounter in Exam 3. The exam will have between 3 and 5 exercises.

Exercises

1. Write a recursive Python function that returns the minimum number in a list.

   Test cases:

   ```
   >>> min([13, 21, 53, 4])
   4
   >>> min([12, -10, 16])
   -10
   >>> min([])
   "there is no minimum in the empty list"
   ```

2. Write a recursive Python function `succ`(n) that returns the successor of the natural number n.

   Test cases:

   ```
   >>> succ(23)
   24
   >>> succ(0)
   1
   ```

3. Write a recursive Python function `bump`(L) that adds 1 to each element of the list L.

   Test cases:

```
>>> bump([134,2,43,76,25])
[135, 3, 44, 77, 26]
>>> bump([])
[]
```

4. Write a recursive Python function `member`(x,L) that returns True if x is a member of L and False otherwise.

   Test cases:

   ```
   >>>
   >>> member(1,[3,6,1])
   True
   >>> member(24,[4,'honey',(2,5)])
   False
   >>> member(5,[])
   False
   >>>
   ```

5. Write an execution or hand-trace of `member`(4,[5, 12, 4]).

6. Write a recursive Python function `uniquify`(L) that removes all repetitions in the list L.

   Test cases:

   ```
   >>> uniquify([21,12,23,12,6,13,12])
   [21, 23, 6, 13, 12]
   >>> uniquify([12,6,5,12,'a','b','a'])
   [6, 5, 12, 'b', 'a']
   >>> uniquify([])
   []
   >>>
   ```

7. Write a recursive Python function `pal`(s) that returns True if string s is a palindrome and False otherwise.

   Test cases:

   ```
   >>> pal('')
   True
   >>> pal('pheromones')
   False
   >>> pal('nadia+aidan')
   True
   >>>
   ```

   Did you know that Nadia is Aidan backwards?

8. Write a recursive Python function `evens`(lst) that returns the list containing the even numbers in lst.

   Test cases:

```
>>> evens([10,3,5,6,23,7])
[10, 6]
>>> evens([])
[]
>>>
```

9. The American Heart Association recommends a maximum daily sugar intake of 20 grams for adult women, 36 grams for adult men, and 12 grams for children. To put that in perspective, a can of soda alone can have as many as 40 grams of sugar. Using the UseIt or LoseIt principle, write a recursive Python function `snacks`(target,L) that given a recommended maximum sugar intake target in grams, and a list L of sugar grams contained in snacks, it calculates the maximum number of snacks that a person can consume in one day without exceeding the target.

   Test cases:

```
>>> snacks(24,[])
0
>>> snacks(22,[10,4])
5
>>> snacks(33,[5,4,8,10])
8
```

10. Recall the following Python program `subset` from our lecture notes.

```
def subset(target,L):
    if target == 0: return True
    elif L == [] : return False
    elif L[0] > target: return subset(target, L[1:])
    else:
        useIt = subset(target - L[0], L[1:])
        loseIt = subset(target, L[1:])
        return useIt or loseIt
```

11. Write a memoized version of `subset`.

12. Write an execution or hand-trace of `subset(5,[2,4,3])` using the original version of `subset`.

13. Write an execution or hand-trace of `subset(5,[2,4,3])` using the memoized version of `subset`.