CS 110 – Creative Problem Solving in Computer Science

Stevens Institute of Technology © 2016

Lab Assignment

Instructor: Adriana Compagnoni

November 18, 2016

Remarks

- This assignment is open books.
- You are expected to use your laptop to implement your solution.
- You can ask the Teaching Assistants for help during lab.
- Please refrain from communicating with other students during this assignment.
- This assignment is due at the end of the Lab session.

Exercises

1. (Recursion) The list l contains the name and salary for employees at SmartSoftware LLC. The CEO, Emma Winter, announced yesterday that all salaries will be updated with a 2% inflation increase.

   - (40 points) Write a recursive Python function `salaryUpdate`(l) that takes the list l of tuples (name, salary) and returns a new list where the salaries reflect the CEO's announcement.

   Test cases:

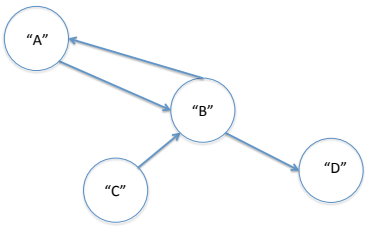
   ```
   >>> salaryUpdate([])
   []
   >>> salaryUpdate([("Jim", 70000), ("Amy", 80000)] )
   [('Jim', 71400), ('Amy', 81600)]
   >>>
   ```

   Remember that tuples are very similar to lists, except that they cannot be updated, but you can always make a new one using elements of the old one.

   Notice the following script where we use a list of tuples L, and we make a new tuple using the first element of L.

   ```
   >>> L = [('Jim', 70000), ('Amy', 80000)]
   >>> L[0]
   ('Jim', 70000)
   >>> L[0][0]
   'Jim'
   >>> L[0][1]
   70000
   >>> t = (L[0][0], L[0][1]+500)
   >>> t
   ('Jim', 70500)
   >>>
   ```

2. (Use-It-Or-Lose-It) (30 points) Consider a graph defined by a list of edges (a , b) connecting its nodes. Following the Use It or Lose It strategy, write a Python program `path(x,y,l)` that given two nodes x and y, and a list of edges l, it returns True if there is a path from x to y in the graph and False otherwise.



The list of edges describing this graph is [('A','B'), ('B','A'), ('C','B'), ('B','D')]. In this graph there is path from 'A' to 'D', but no path from 'D' to 'A'.

The path from 'A' to 'D' is composed by the edges ('A','B') and ('B','D').

Your code should work for any list l of edges, not just the one in the picture.

Hint: All the Use-It-Or-Lose-It algorithms on lists follow the same structure.

```
if "we are done": return "something"
elif "the list is empty, we run out of options": return "something else"
elif "the list is not empty, but the first argument is useless":
       return "go on with the rest of the list (LoseIt case)"
elif "the list is not empty but the first argument may be useful.
      Here we have two options":
      UseIt=
      LoseIt =
      return "something combining the results from UseIt and LoseIt"
```

Test cases:

```
>>> path('A','B',[])
False
>>> path('C','C', [('A','B'), ('B','A'), ('C','B'), ('B','D')])
True
>>> path('D','A', [('A','B'), ('B','A'), ('C','B'), ('B','D')])
False
>>> path('A','D', [('A','B'), ('B','A'), ('C','B'), ('B','D')])
True
```

3. Consider the code for the function `subset` that we saw in class.

```
def subset( target, numberList ):
    ''' Returns True if there exists a subset of numberList that adds
    up to target and returns False otherwise.'''
    if target == 0:
        return True
    elif numberList == ():
        return False
    if numberList[0] > target:
        return subset(target, numberList[1:])
    else:
        useIt = subset(target - numberList[0], numberList[1:])
        loseIt = subset(target, numberList[1:])
    return useIt or loseIt
```

```
>>> subset(25, (17, 20, 2, 15))
False
>>> subset(10, (2, 2, 3, 12, 3))
True
>>>
```

- (30 points) Write a memoized version of `subset`.
  Hint: remember that lists cannot be keys, so as we did in class with `memochange`*, the list of numbers has to be represented with tuples.*