Homework Assignment 6: Trees and Graphs

Problem 1: Expression Trees (15 points)

Use the expression tree class given below and in hw6_tools.py to create the expression tree for the following fully parenthesized expression.

```
((3+(1/2))*(5-2))
```

Your expression tree will be named `expTree`. You should create it “manually” using series of Python statements.

```
class ExpTree:
    def __init__(self,key):
        self.key = key
        self.leftChild = None
        self.rightChild = None
    def getRightChild(self):
        return self.rightChild
    def getLeftChild(self):
        return self.leftChild
    def setRootVal(self,obj):
        self.key = obj
    def getRootVal(self):
        return self.key
```

Problem 2: Expression Tree Traversal (20 points)

Implement the three tree traversal functions given in chapter 6.7 your book so that each of the following function calls prints the keys at each node in the order they are evaluated. Start with the code in the book for these three functions, some of the listings may need adapting so that they work as a function. Each function definition takes a single argument being the expression tree. To make your output look like the example below, use `print(key,end=' ')`, where key is the key you want to print. This uses a space instead of a newline at the end of each print statement.

You will receive half credit for this problem if your functions simply print the correct list of tokens for the expression tree from problem 1, and full credit if your functions work on any expression tree.

Example output for the expression tree corresponding to 1 + 1 is given for each function call on the next page.

```
postorder(expTreeTwo)
1 1 +
preorder(expTreeTwo)
+ 1 1
inorder(expTreeTwo)
1 + 1
```

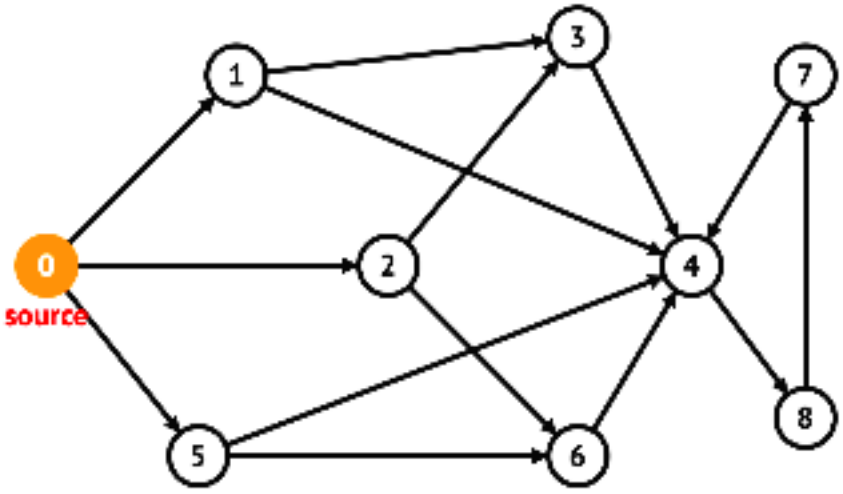
Problem 3: Printing Expression Trees (15 points)

Modify the function printexp given in chapter 6.7 your book (and hw6_tools.py) so that the parentheses are not printed around an operand if the operand consists only of a number. Note that the function printexp doesn't actually print the expression, it returns a string. The following function call on the expression tree created for problem 1 should result in the exact fully parenthesized expression we gave you.

```
print(printexp(expTree))
((3+(1/2))*(5-2))
```

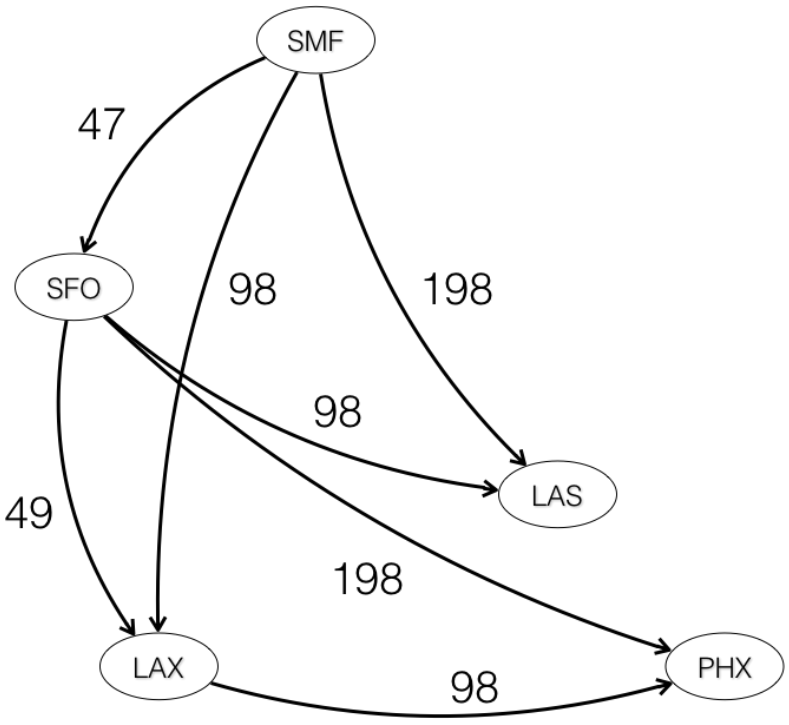
Problem 4: Graph Traversal (25 points)

Traverse the following graph, starting at vertex (node) 0, so that as each vertex is is added to a Python list in the order that it is processed, or visited, using **breadth first search** (chapter 7.9) and **depth first search** (chapter 7.15). Each vertex should only appear once in the Python list as an integer and the first entry in each list should be a 0. Assume in the event of a tie that vertices are visited in increasing numerical order. This should require no programming. We will cover these traversal methods in lecture.



Problem 5: Shortest Path (25 points)

Create a Graph object that encodes the following directed weighted graph. You can use a series of manual Python statements to encode the graph. To add directed weighted edges to the Graph object, you can use the `addEdge` method which takes three parameters from, to, and cost.



Next use the included function `dijkstra` to find and print the shortest path from Sacramento to all of the other airports in the list. The function works by adding distances to each of the vertices. After the function executes you can get the distances using the method getDistance. You'll need to study the code in hw6_tools.py a bit to see how to use it.

Your printed output should look like this

```
From To Cost
SMF SFO 47
SMF LAX 96
SMF LAS 145
SMF PHX 194
```