<table>
<thead>
<tr>
<th>Problem</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>15</td>
</tr>
<tr>
<td>Problem 2</td>
<td>5</td>
</tr>
<tr>
<td>Problem 3</td>
<td>10</td>
</tr>
<tr>
<td>Problem 4</td>
<td>10</td>
</tr>
<tr>
<td>Problem 5</td>
<td>10</td>
</tr>
<tr>
<td>Problem 6</td>
<td>10</td>
</tr>
<tr>
<td>Problem 7</td>
<td>10</td>
</tr>
<tr>
<td>Problem 8</td>
<td>5</td>
</tr>
<tr>
<td>Problem 9</td>
<td>10</td>
</tr>
<tr>
<td>Problem 10</td>
<td>15</td>
</tr>
</tbody>
</table>
1 What do they Do?

(a) Complete the specification in the following

```python
def f(s):
    """
    PreC: s is a string.
    """
    t = s
    nullstring = ''
    for c in s:
        if s.count(c)>1:
            t = t.replace(c,nullstring)
    return t
```

5 points:

Returns a string obtained from s by deleting all characters that appear more than once

-1 if "return" is not mentioned

(b) What is the output of the call F([30,40,10,20])?

```python
def F(x):
    """
    PreCondition: x is a nonempty list of distinct ints
    """
    n = len(x)
    for k in range(n-1):
        if x[k]>x[k+1]:
            t = x[k]
            x[k] = x[k+1]
            x[k+1] = t
    print x
```

5 points:

30 40 10 20
30 10 40 20
30 10 20 40

4 points

30 40 10 20
30 10 40 20

4 points

30 10 20 40

2 points for these 1-liners

30 10 40 20
30 10 10 20
10 20 30 40
40 30 20 10
(c) The following code displays a 10,000 non-intersecting randomly colored disks. Comment on the expected number of displayed red disks, the expected number of displayed white disks, and the expected number of displayed blue disks. FYI, `randu(a,b)` returns a float that is randomly chosen from the interval $[a, b]$.

```python
from random import uniform as randu
from simpleGraphics import *
MakeWindow(101)
r = 0.3
for i in range(100):
    for j in range(100):
        x = float(i)
        y = float(j)
        p = randu(0,1)
        if p <= .1:
            DrawDisk(x,y,r,RED)
        elif p <= .4:
            DrawDisk(x,y,r,WHITE)
        else:
            DrawDisk(x,y,r,BLUE)
ShowWindow()
```

5 points

1000 Red
3000 White
6000 Blue

5 points

10%
30%
60%

3 points

1000 Red
4000 White
5000 Blue
2 Functions and Lists

Complete the following function so that it performs as specified

```python
def Trim(L):
    """ Returns a list of strings K that has four properties:
    (1) every entry in K is in L
    (2) every entry in L is in K
    (3) no entry in K is repeated
    (4) K is sorted.
    L is not modified.
    """
    PreC: L is a nonempty list of strings

    Thus, if L = ['a', 'c', 'a', 'b', 'h', 'a', 'c']
    then ['a', 'b', 'c', 'h'] is returned.

    5 point solution:
    K = []
    for s in L:
        if s not in K:
            K.append(s)
    K.sort()
    return K

    5 point point solution:
    K = []
    for k in range(len(L)):
        if L[k] not in K:
            K.append(L[k])
    K.sort()
    return K

    3 point solution:
    K = L
    for s in L:
        if s not in K:
            K.append(s)
    K.sort()
    return K
```

5 point solution:

```python
K = []
for s in L:
    if s not in K:
        if K.count(s) == 0:
            K.append(s)
        else:
            print("-1 for using find on a list")
        K.sort()
    return K
```

5 point point solution:

```python
K = []
for k in range(len(L)):
    if L[k] not in K:
        if K.count(s) == 0:
            K.append(L[k])
        else:
            print("-1 for using find on a list")
    K.sort()
    return K
```

3 point solution:

```python
K = L
for s in L:
    if s not in K:
        if K.count(s) == 0:
            K.append(s)
    K.sort()
return K
```
3 Boolean Operations

(a) Implement the following function so that it performs as specified.

```python
def Q1(s1, s2, s3):
    
    """ Returns True if s1, s2, and s3 have a character in common and False otherwise.

    PreCondition: s1, s2, and s3 are nonempty strings
    ""

    for c in s1:
        for c1 in s1:
            if c in s2 and c in s3:
                for c2 in s2:
                    return True
                return False
        return False
    return True
```

5 point solutions

```python
for c in s1:
    if c in s2 and c in s3:
        return True
    return False
```

-2 if "or" instead of "and". -2 if "True" part is right but "False" part is not. And vice versa.

3 point solution:

```python
for c in s1:
    if c in s2 and c in s3:
        return True
    return False
```

1 point
No loop but some relevant Boolean expression
Note: It is possible to do this problem using find
(b) Assume that $B_1, B_2, B_3, B_4$, and $B_5$ are initialized Boolean variables. Rewrite the following code so that it does not involve any nested ifs. The rewritten code must be equivalent to the given code, i.e., it must render exactly the same output no matter what the value of the five initialized Boolean variables.

```python
if B1:
    if B2:
        print 'A'
    elif B3:
        print 'B'
else:
    if B4 or B5:
        print 'C'
    else:
        print 'D'
```

5 points

```python
if B1 and B2:  # 3 points for printing A and B correctly
    print 'A'
elif B1 and B3:
    print 'B'
elif (not B1) and (B4 or B5):  # 2 points for printing C and D correctly
    print 'C'
elif (not B1):  # -2 if the not B1 is missing
    print 'D'
```

3 points

```python
if B1 and B2:
    print 'A'
elif B1 and B3:
    print 'B'
elif (B4 or B5):
    print 'C'
else:
    print 'D'
```

Typical 1 point solution

```python
if B1 and B2
    print 'A'
if B1 and B3:
    print 'B'
if B4 or B5:
    print 'C'
else:
    print 'D'
```
4 While Loops

(a) Rewrite the following code so that it does the same thing but with while-loops instead of for-loops.

```python
s = 'abcdefghijklmnopqrstuvwxyz'
for i in range(26):
    for j in range(0, i-1):
        for k in range(j, i):
            print s[k] + s[j] + s[i]
```

5 points

```python
s = 'abcdefghijklmnopqrstuvwxyz'
i = 0
while i < 26:
    j = 0
    while j < i-1:
        k = j
        while k < i:
            print s[k] + s[j] + s[i]
            k += 1
        j += 1
    i += 1
```

3 points

```python
s = 'abcdefghijklmnopqrstuvwxyz'
i = 0
j = 0
k = 0
while i < 26:
    while j < i-1:
        while k < i:
            print s[k] + s[j] + s[i]
            k += 1
        j += 1
    i += 1
```

2 points

```python
s = 'abcdefghijklmnopqrstuvwxyz'
i = 0
j = 0
k = 0
while i < 26:
    while j < i-1:
        while k < i:
            print s[k] + s[j] + s[i]
        j += 1
    i += 1
```

1 point same as preceding but no initialization
1 point same as preceding but no updates
(b) Implement the following function so that it performs as specified.

```python
def OverBudget(A, M):
    """ Returns the smallest k so that \( \text{sum}(|A[0:k,0]|) \geq M, \text{sum}(|A[0:k,1]|) \geq M, \text{and} \text{sum}(|A[0:k,2]|) \geq M \). If no such k exists, returns 0.
    PreC: A is an n-by-3 numpy array of ints. M is an int.
    """

To illustrate, suppose

\[
A = \begin{bmatrix}
2 & 7 & 1 \\
1 & 0 & 4 \\
3 & 2 & 5 \\
0 & 1 & 4 \\
4 & 0 & 6
\end{bmatrix}
\]

If \( M = 3 \), then the value returned should be 2. If \( M = 10 \), then the returned value should be 5. If \( M = 100 \), then the returned value should be 0. You are not allowed to use the built-in function `sum` or `for-loops`.

5 point solution:

```python
k = 0
s0 = 0
s1 = 0
s2 = 0
(m, n) = A.shape
while k < m:
    s0 += abs(A[k,0])
    s1 += abs(A[k,1])
    s2 += abs(A[k,2])
    k += 1
    if s0>=M and s1>=M and s2 >= M:
        return k
return 0
```

5 point solution:

```python
k = 0
s0 = 0
s1 = 0
s2 = 0
(m, n) = A.shape
OneSumShort = (s0<M or s1<M or s2<M)
while k < m and OneSumShort:
    s0 += abs(A[k,0])
    s1 += abs(A[k,1])
    s2 += abs(A[k,2])
    OneSumShort = (s0<M or s1<M or s2<M)
    k += 1
if not OneSumShort:
    return k
else:
    return 0
```
5 Recursion

Binary search is a divide and conquer process that can be used to determine whether or not a given value is an entry in a sorted list. Here is an informal, recursive illustration of the process applied to finding a name in a phone book assuming that there is one name per page:

Look-Up Process:
- if the phone book has one page
  - Report whether or not the name is on that page
- else
  - Tear the phone book in half
  - Apply the Look-Up Process to the relevant half-sized phonebook

Develop a recursive binary search implementation of the following function so that it performs as specified. You are not allowed to use the "in" operator.

```python
def BinSearch(x, a, L, R):
    """Returns True if x in a[L:R+1] is True and False otherwise.
    Precondition: a is a length n-list of distinct ints whose entries are sorted from smallest to largest. L and R are ints that satisfy 0<=L<=R<n. x is an int with the property that a[L]<=x<=a[R]."
    if R==L:
        return x = a[L]  # -2 for len(a)==1 instead of R==L
        -1 for x in a[L,R+1]
        -1 for return L
    else:
        mid = (L+R)/2  # -2 for (a[L]+a[R])/2
        -2 for len(a)/2
        if x <= a[mid]:  # -1 for x <= mid
            return BinSearch(x, a, L, mid)  # 2 points
        else:
            return BinSearch(x, a, mid+1, R)  # 2 points
    # -1 if "mid" and not "mid+1"
```

For very wrong solutions,
- 1 point for a single if-else
- 1 point if the if-part tries to deal with the base case
- 1 point if the else part tries to come up with a half-sized problem
- 1 point if there is a recursive Binsearch call and it recognizes that BinSearch returns a Boolean value

Better solution
```python
def BinSearch(x, a, L, R):
    if R==L:
        return True
    else:
        m = (L+R)/2
        if a[L]<=x<=a[m]:
            return BinSearch(x, a, L, m,)
        elif a[m+1]<=x<=a[R]:
            return BinSearch(x, a, m+1, R)
        else:
            return False
```

Better solution
```python
def BinSearch(x, a, L, R):
    if R==L:
        return True
    else:
        m = (L+R)/2
        if a[L]<=x<=a[m]:
            return BinSearch(x, a, L, m,)
        elif a[m+1]<=x<=a[R]:
            return BinSearch(x, a, m+1, R)
        else:
            return False
```
6 Function Execution

What is the output if the following Application Script is executed?

```python
def F(a):
    b = True
    for k in range(len(a)):
        b = D(a,k) and b
    return b

def D(a,k):
    a[k] = a[k]-1
    return a[k] >= 0

if __name__ == '__main__':
a = [1,2,3,4]
print F(a)
print a
print F(a)
print a
```

Fact: D(a,k) subtracts 1 from a[k] and returns True iff the modified a[k] is nonnegative

Fact: F(a) subtracts 1 from every entry in a and returns True iff every entry in the modified a is nonnegative

The first call to F modifies a to [0,1,2,3] and returns True.

The second call to F modifies a to [-1,0,1,2] and returns False

So the 10 point solutions are

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True, [0,1,2,3]</td>
</tr>
<tr>
<td>[0,1,2,3]</td>
<td>False, [-1,0,1,2]</td>
</tr>
<tr>
<td>False</td>
<td></td>
</tr>
<tr>
<td>[-1,0,1,2]</td>
<td></td>
</tr>
</tbody>
</table>

8 points

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>[0,1,2,3]</td>
</tr>
<tr>
<td>[0,1,2,3]</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>[-1,0,1,2]</td>
</tr>
<tr>
<td>[-1,0,1,2]</td>
<td>False</td>
</tr>
</tbody>
</table>

5 points

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>[0,2,3,4]</td>
</tr>
<tr>
<td>True</td>
<td>[0,1,3,4]</td>
</tr>
</tbody>
</table>

3 points

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>[1,2,3,4]</td>
</tr>
<tr>
<td>True</td>
<td>[1,2,3,4]</td>
</tr>
</tbody>
</table>

In any of the above, if there are extra lines of output (as if there was a print statement inside the functions) then -2

2 points if some good chit chat and output looks like

```
list
boolean
list
boolean
```
7 Short Answer

(a) Why is inheritance such an important aspect of object oriented programming?

4 points for any of these

With inheritance, it is legal for a method from an existing class to be applied to an object of the new class

2 point answers:

- Enables one to reuse software.
- Enables one to build on old software.
- Makes it easier to maintain software.

(b) What does it mean to say that an operator like "+" is overloaded?

3 point answers:

The operation performed depends on the operands.

Thus, x+y may mean concatenation if
x and y are strings and addition if x and y are floats

(c) The numpy module supports the addition of arrays. What does this mean?

3 point answers

If x and y are numerical arrays of the same size, then x+y creates a new array of the same size obtained by adding entries. (OK not to say "numerical")

An example like [1,2,3]+[4,5,6] = [5,7,9]
8 Inverting a Dictionary

Implement the following function so that it performs as specified.

```python
def Invert(D):
    """ Returns a dictionary that is obtained from D by swapping its keys and values."

    PreC: D is a dictionary with the property that every value is either a string or a number, and no values are repeated throughout the entire dictionary.

    Thus, if D = {1: 'x', 'z': 4, 'x': 'z'}, then the dictionary { 'x': 1, 4: 'z', 'z': 'x' } is returned. You are not allowed to use the dictionary methods keys or values.

    5 points
    E = {}
    for d in D:
        theKey = d
        theValue = D[d]
        E[theValue] = theKey
    return E

    5 points
    E = {}
    for d in D:
        E[D(d)] = d
    return E

    2 points
    E = {}
    for d in D:
        E.append(d)
    return E

    5 points
    Keys = []
    Values = []
    for d in D:
        Keys.append(d)
    Values.append(D[d])
    E = {}
    for k in range(length(Keys)):
        E[Values[k]] = Keys[k]
    return E

    3 points If everything is OK but they overwrite D and that causes a screw up
9 A Modified Energy Class

Consider the following modification of the class `Energy` that was part of A7:

```python
class EnergyMod:
    
    Name: a string that is the name of the building
    Image: a string that specifies the path of the building's jpeg image
    E_rate: a length-24 numpy array where E-Rate[k] is the cost of electricity per
            unit of consumption during the kth hour of the day, k in range(24)
    S_rate: a length-24 numpy array where S-Rate[k] is the cost of steam per
            unit of consumption during the kth hour of the day, k in range(24)
    C_rate: a length-24 numpy array where C-Rate[k] is the cost of chilled water per
            unit of consumption during the kth hour of the day, k in range(24)
    A: a 35040-by-3 numpy array that houses all the energy consumption snapshots.
       In particular, A[k,0], A[k,1], and A[k,2] house the
       electricity, steam, and chilled water consumption during the kth 15-minute
       period of the year.
    TS_dict: a 35040-item time stamp index dictionary. If ts is a valid time stamp and
             k is the value of TS_dict(ts), then A[k,:], houses the consumption data
             associated with ts.
```

Notice that instead of a single consumption rate for each of the three energies we have a list of 24 rates, one for each hour in the day. ASSUME STANDARD TIME. And just to be clear about what we mean by “hour of the day”, if a consumption reading is associated with time stamp `dd-MMM-2014-hh-mm`, then the relevant hour of the day is `int(hh)`.

Implement a method `arbitraryBill(self, T1, T2)` for the `EnergyMod` class that returns the total cost of energy consumed by the building represented by `self` from time stamp `T1` up to time stamp `T2`. As an example,

```python
M = EnergyMod('Gates')
x = M.arbitraryBill('15-May-2014-08-00', '16-May-2014-11-45')
```

would assign to `x` the total energy cost of running Gates Hall from 8AM May 15 up to noon May 16. You are allowed to use the function `Invert` from Problem 8.

```python
def arbitraryBill(self, T1, T2):
    D = Invert(self.TS_dict)  
    total = 0  
    k1 = self.TS_dict[T1]  
    K2 = self.TS_dict[T2]  
    for k in range(T1, T2):
        TS = D[k]  
        Hour = int(TS(12:14))  
        E = self.E_rate[Hour]  
        S = self.S_rate[Hour]  
        C = self.C_rate[Hour]  
```

-1 if they do not use the `self.` notation
10 Methods

Assume the availability of the following class.

class Fraction:
    
    A class that can be used to represent fractions.

    Attributes:
    num: the numerator [int]
    den: the denominator [positive int]

    Invariant: num and den have no common factors larger than 1.
    
    def __init__(self,p,q):
        
        """ Returns a Fraction Object that represents p/q in lowest terms.
        
        PreC p and q are ints and q is nonzero
        """

    def lowestTerms(self):
        
        """ Updates self so that its numerator and denominator are
        reduced to lowest terms.
        """

(a) Write a method AddOne(self) that updates self by adding one to the numerator and denominator of the fraction represented by self.

5 points 3 points 2 points

def AddOne(self):
    self.num += 1
    self.den += 1
    self.lowestTerms()

At most -1 for syntax errors like

p.Fraction(q) instead of Fraction(p,q)
lowestTerms(self) instead of self.lowestTerms()

No points if they leave off the  
def AddOne(self) header
(b) Consider the class

```python
class pointFract:
    """
    A class that can be used to represent points whose
    x and y coordinates are fractions
    """

    Attributes:
    x: x-coordinate [Fraction]
    y: y-coordinate [Fraction]

    """
    def __init__(self,F1,F2):
        """
        Returns a Fraction Object that represents the point (F1,F2)
        """
        PreC: F1 and F2 are Fractions

    """
```

Write a method `distToOrigin(self)` for this class that returns the distance of self to the origin. FYI, the distance of the point \((a, b)\) to the origin is given by \(\sqrt{a^2 + b^2}\). You may assume that `math.sqrt` is available.

8 point solutions

```python
distToOrigin(self):
    F1 = self.x  # 1
    xfloat = float(F1.num)/float(F1.den)  # 1
    F2 = self.y  # 1
    yfloat = float(F2.num)/float(F2.den)  # 1
    d = math.sqrt(xfloat**2 + yfloat**2)  # 1
    return d  # 1

distToOrigin(self):
    xfloat = float(self.x.num)/float(self.x.den)  # 2
    yfloat = float(self.y.num)/float(self.y.den)  # 2
    return math.sqrt(xfloat**2 + yfloat**2)  # 1
```

3 point solution

```python
distToOrigin(self):
    a = self.x  # 1
    b = self.y  # 1
    return math.sqrt(a**2 + b**2)  # 1
```

2 point solution

```python
distToOrigin(self):
    a = self.F1  # 1
    b = self.F2  # 1
    return math.sqrt(a**2 + b**2)  # 1
```

-1 if forget to use float

-1 (max) if syntax mistake like self(x)

(c) Consider the code

```python
F1 = Fraction(1,2)
F2 = Fraction(3,4)
P1 = pointFract(F1,F2)
P2 = P1
F2 = F1
```
P2 references a `pointFraction` object. What are the coordinates of the point represented by that object? For full credit, you must draw a state diagram that fully depicts all the references and objects.

**P2 represents the point (1/2,3/4)**

```
   --------------  ---------
   F1--->| num: 1 |   |   |
         | den: 2 |<------ x    |<------ P1
         /   |
         /   |
         /   |
         /   |<-------- P2
   F2   | num: 3 |<-------- y   |
         | den: 4 |   |   |
   --------------  ---------
```

2 points for correct point.  
2pts: (.50,.75)  1pt: (.50,.50)

3 points for state diagram.  
1 point for showing 3 objects.  
1 point for arrows from P1, P2, F1 and F2  
1 point for arrows from x and y
## Function Information

<table>
<thead>
<tr>
<th>Function</th>
<th>What It Does</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>len(s)</code></td>
<td>returns an <code>int</code> that is the length of string <code>s</code></td>
</tr>
<tr>
<td><code>s.count(t)</code></td>
<td>returns an <code>int</code> that is the number of occurrences of string <code>t</code> in string <code>s</code></td>
</tr>
<tr>
<td><code>s.find(t)</code></td>
<td>returns an <code>int</code> that is the index of the first occurrence of string <code>t</code> in the string <code>s</code>. Returns -1 if no occurrence.</td>
</tr>
<tr>
<td><code>s.replace(t1,t2)</code></td>
<td>returns a string that is obtained from <code>s</code> by replacing all occurrences of <code>t1</code> with <code>t2</code>.</td>
</tr>
<tr>
<td><code>floor(x)</code></td>
<td>returns a float whose value is the largest integer less than or equal to the value of <code>x</code>.</td>
</tr>
<tr>
<td><code>ceil(x)</code></td>
<td>returns a float whose value is the smallest integer greater than or equal to the value of <code>x</code>.</td>
</tr>
<tr>
<td><code>int(x)</code></td>
<td>If <code>x</code> has type <code>float</code>, converts its value into an <code>int</code>. If <code>x</code> is a string like <code>-123</code>, converts it into an <code>int</code> like -123.</td>
</tr>
<tr>
<td><code>float(x)</code></td>
<td>If <code>x</code> has type <code>int</code>, converts its value into a <code>float</code>. If <code>x</code> is a string like <code>'1.23'</code>, converts it into a <code>float</code> like 1.23.</td>
</tr>
<tr>
<td><code>str(x)</code></td>
<td>Converts the value of <code>x</code> into a string.</td>
</tr>
<tr>
<td><code>DrawDisk(x,y,r,c)</code></td>
<td>Draws a circle with center <code>(x, y)</code>, radius <code>r</code> and color <code>c</code>.</td>
</tr>
<tr>
<td><code>x.append(y)</code></td>
<td>adds a new element to the end of the list <code>x</code> and assigns to it the value referenced by <code>y</code>.</td>
</tr>
<tr>
<td><code>deepcopy(x)</code></td>
<td>creates a complete copy of the object that is referenced by <code>x</code>.</td>
</tr>
<tr>
<td><code>sum(x)</code></td>
<td>returns the sum of the values in list <code>x</code> assuming that all its entries are numbers.</td>
</tr>
<tr>
<td><code>(m,n) = A.shape</code></td>
<td>assigns the row and column dimensions of the numpy 2D array <code>A</code> to <code>m</code> and <code>n</code> resp.</td>
</tr>
<tr>
<td><code>x.sort()</code></td>
<td>modifies the list of numbers <code>x</code> so that its entries range from smallest to largest</td>
</tr>
</tbody>
</table>