This 150-minute exam has 8 questions worth a total of 120 points. Scan the whole test before starting. Budget your time wisely. Use the back of the pages if you need more space. You may tear the pages apart; we have a stapler at the front of the room.

It is a violation of the Academic Integrity Code to look at any exam other than your own, to look at any other reference material, or to otherwise give or receive unauthorized help.

You will be expected to write Python code on this exam. We recommend that you draw vertical lines to make your indentation clear, as follows:

```python
def foo():
    if something:
        do something
        do more things
    do something last
```

Unless you are explicitly directed otherwise, you may use anything you have learned in this course.

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Total:</td>
<td>120</td>
<td></td>
</tr>
</tbody>
</table>

The Important First Question:

1. [2 points] Write your last name, first name, and Cornell NetId at the top of each page.
Throughout this exam, there are several questions on sequences (strings, lists, and tuples). All sequences support slicing. In addition, you may find the following expressions below useful (though not all of them are necessary).

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>len(s)</td>
<td>Returns: number of elements in sequence s; it can be 0.</td>
</tr>
<tr>
<td>x in s</td>
<td>Returns: True if x is an element of sequence s; False otherwise.</td>
</tr>
<tr>
<td>s.index(x)</td>
<td>Returns: index of the FIRST occurrence of x in s. Raises a ValueError if x is not found.</td>
</tr>
<tr>
<td>s.rindex(x)</td>
<td>Returns: index of the LAST occurrence of x in s. Raises a ValueError if x is not found.</td>
</tr>
<tr>
<td>s.count(x)</td>
<td>Returns: number of times x appears in s. Could be zero</td>
</tr>
<tr>
<td>s.append(x)</td>
<td>(Lists Only) Adds x to the end of list s, increasing length by 1.</td>
</tr>
</tbody>
</table>

2. [18 points total] **Classes and Subclasses**

In Assignment 7 you got experience with GRectangle and GEllipse which extended GObject. They did not have any new attributes beyond those in GObject, but behaved very differently. This question deals with three very similar classes: Shape, Rectangle, and Circle.

These classes are similar to those in A7 except for two very important details. First of all, there is no drawing code. More importantly they do not use the advanced keyword arguments used by Kivy. The expression **keyword should not appear anywhere in your solution.

(a) [8 points] The skeleton for class Shape is provided on the next page. The attributes are immutable; you are to implement properties that enforce this. In addition, you should complete the constructor and method __str__ according to their specification. Your constructor must enforce all invariants.

(b) [10 points] You are to create the classes Rectangle and Circle, each of which is a subclass of Shape. We have not provided you with any skeleton code for these classes; you are to implement everything yourself.

The classes have no new attributes beyond those inherited from GObject. For each class implement a constructor and one other method, according to the following constraints.

- The constructor for Rectangle should have an header that looks exactly like the constructor for Shape, including default values. The body of this constructor should be a single line call to super.

- The constructor for Circle should have three parameters: x, y, and radius. Using super, it should set the width and height attributes to the diameter (= 2r).

- Both classes should have a method calculateArea(), which returns the area of the shape. For class Circle, you may use the constant PI in module math. Remember that the area of a circle is $\pi r^2$.

You can assume that math is imported; you do not need to write an import statement in your code. Implement your classes on the blank page after the class Shape.
class Shape(object):
    """Instance is 2-dimensional geometric shape""
    # IMMUTABLE ATTRIBUTES
    _x = 0.0 # x-coordinate of bottom-left corner; must be a float.
    _y = 0.0 # y-coordinate of bottom-left corner; must be a float.
    _width = 0.0 # shape width; must be a float >= 0.
    _height = 0.0 # shape height; must be a float >= 0.

    # DEFINE PROPERTY x for FIELD _x (specification not necessary)

    # DEFINE PROPERTY y for FIELD _y (specification not necessary)

    # DEFINE PROPERTY width for FIELD _width (specification not necessary)

    # DEFINE PROPERTY height for FIELD _height (specification not necessary)

    def __init__(
    ): # Fill in
        """Constructor: shape with given values x, y, width, and height (in order).
        Precondition: x, y, width, and height are floats with width, height >= 0.0.
        All parameters have a default of 0.0."""

    def __str__(self):
        """Returns: Description of shape geometry in format '[x,y,width,height]'.
3. [20 points total] **Call Frames and Diagrams**

Suppose you were to modify class `Shape` to include the following method.

```python
def contains(self,q):
    """Returns: True if point q is in this Rectangle; False otherwise.
    Precondition: q is a list [x,y]."
    1 in_x = self.x < q[0] and q[0] < self.x+self.width
    2 in_y = self.y < q[1] and q[1] < self.y+self.height
    3 return in_x and in_y
```

Consider then the following python code.

```python
rect = Rectangle(0.0,0.0,1.0,2.0)
circle = Circle(1.0,1.0,3.0)
shape = rect
p = [1.0, 2.0]
value = shape.contains(p)
```
(a) [10 points] Execute the code on the previous page in global space. In other words you should draw all the variables in global space, as well as any folders created in heap space. Do not worry about call frames (yet). You do not have to draw the object partitions.

(b) [10 points] Draw your execution of the call `shape.contains(p)`. You will draw what the call frame looks like at four points in time: when the function starts, and once after it completes each of the line in the function. You do not need to redraw the folders for `shape` and `p`; simply use the folder names for your answer in part (a).
4. [14 points] **Recursion**

We want to compress strings that have long sequences of equal characters. For example, we want to compress 'bbbbaaaa$\ldots$$\ldots$$d'$ to 'b4a3$16d1'$. In the compression, each sequence of equal characters is given by the character followed by the length of the sequence. Write the function `compress` to do this. Use no loops; use only recursion. You may use the function `eq_chars` specified below as a helper. **Do not implement `eq_chars`.**

**Hint:** The base case is not necessarily a string with one character.

```python
def eq_chars(s, i):
    """Returns: length of sequence of equal characters starting at s[i].
    Examples: eq_chars('aaaxxyx', 0) is 3 and eq_chars('aaaxxyx', 5) is 1
    Precondition: s is a string, 0 <= i < len(s).""

def compress(s):
    """Returns: the compression of s, as explained above.
    Precondition: s a nonempty string with no digits 0..9.""
```

5. [16 points total] **Exceptions and Dispatch-on-Type**

In all of the questions below, you may assume that `ValueError` and `TypeError` are subclasses of `StandardError`, but neither is a subclass of the other.

(a) [9 points] Suppose you are given the following function definitions.

```python
def first(n):
    x = 0
    try:
        x = second(n)
    except StandardError:
        x = x+1
    return x

def second(n):
    y = 2
    try:
        y = third(n)
    except ValueError:
        y = y+5
    return y

def third(n):
    if n == 0:
        raise ValueError()
    elif n == 1:
        raise TypeError()
    return n+10
```
Give the value of each function call below. If there is no value (e.g. program crashes), tell us that. To get full credit, explain how the call recovers from errors, if it recovers at all, with the line numbers provided.

i. first(0)

ii. first(1)

iii. first(2)

(b) [7 points] The method eq is used to define == on objects. When sorting objects, we need the comparison operators (e.g. <, >, <=, and >=). We implement these via the method cmp specified below. Implement this method for the class Cornellian, paying special attention to how it should handle errors.

```python
class Cornellian(object):
    # Instance is a person at Cornell.
    def __eq__(self, other):
        # Returns: -1, 0, 1 indicating whether self is less than, equal to, or greater than other (e.g. -1 means self < other).
        raise TypeError if other is not an instance of Cornellian.
        To compare self and other, first compare names as strings. If names are equal, compare netids as strings. (Recall we can always compare strings with < and >). If both attributes are equal, return 0.
```

Page 7
6. [16 points] **While-Loops and Lists**

A triangular array is a ragged list whose rows start (end) at size one and increase (decrease) by one each row. Examples of triangular arrays are shown to the right. These arrays have applications in scientific computation.

As with images in Assignment A6, we can talk about the *transpose* of a triangular array. The transpose is a triangular array with rows and columns swapped with each other. The two triangular arrays shown to the right are the transpose of one another.

Complete the function `transpose`, which takes an increasing triangular array and returns the (decreasing) transpose of the original, by filling in the blanks below. Note that the loop invariants are given. **Solutions that do not preserve the invariant will lose points.**

```python
def transpose(b):
    """Returns: a new list which is the transpose of b.
    Precondition: b is a list representing an increasing triangular array."""

    result = __________
    i = __________

    # inv: result[0..i-1] contains rows 0..i-1 of the transpose
    while ________________:
        # Initialize whatever you need for inner invariant here

        # inv: row[0..k-1] contains elements 0..k-1 of column being transposed
        while ________________:
```

---

**A triangular array (Increasing):**

```
1
2   0
5   6   -1
4   -2   3   8
```

**A triangular array (Decreasing):**

```
1   2   5   4
0   6   -2
-1   3
8
```
7. [18 points total] **Sorting**

The act of sorting is taking a list with given precondition, and rearranging the elements so that the list satisfies the following postcondition.

$$\begin{array}{cccc}
0 & h & k & \text{len}(b) \\
\text{pre:} & b & & ??? \\
0 & h & k & \text{len}(b) \\
\text{post:} & b & & \text{sorted} \\
\end{array}$$

(a) [6 points] Give the loop invariant for sorting using **insertion sort**. You must write the invariant using our pictorial representation. Your placement of variables should be clear.

(b) [12 points] Implement the function **insertion_sort** below. You do not need to state the invariant above, but you must follow it. Answers that do not follow the invariant can get at most half-credit. If you need any helper functions, you must implement them as well (you do not need specifications or invariants for helpers).

```python
def insertion_sort(b, h, k):
    """Uses insertion sort on the segment b[h..k]"
    """Precondition: b is a list, h <= k are positions in b"""".
8. [16 points total] **Odds and Ends.**

   (a) [3 points] What is an assertion? What is the relationship between an assertion and the `assert` statement?

   (b) [4 points] Describe the four steps that happen when you call a constructor.

   (c) [3 points] Explain the difference between the command `import math` and the command `from math import *`. 
(d) [3 points] Below are three expressions. For each one, write its value. If evaluation leads to an error, just say BAD (do not tell us the exception)

\[
\begin{align*}
\text{True or } (5/0 < 1) & \quad (5/0 < 1) \text{ or True} & \quad 3/2
\end{align*}
\]

(e) [3 points] Consider the function `foo` defined below.

```python
def foo():
    return 5
```

What is the difference between the contents of the variables `x` and `y` after the assignment statements below?

```python
x = foo()
y = foo
```