Do not open this exam until you are told. Read these instructions:

1. This is a closed book exam. No calculators, notes, or other aids are allowed. If you have a question during the exam, please come to the front of the class.
2. You must turn in your exam immediately when time is called at the end.
3. Three problems, which add up to 50 points total. 120 minutes.
4. In order to be eligible for as much partial credit as possible, show all of your work for each problem, write legibly, and clearly indicate your answers. Credit cannot be given for illegible answers.

Fill in the following:

NAME :

G# :

Lab section number: ____________________
scratch
1. Trace through the following code. What does it print?

def func1(x, lis):
    print "lis[x]: " + lis[x]
    if x > 0:
        x = x - 1
    else:
        x = x + 6
    try:
        integer = int(lis[x])
        print "div: " + str(5//integer)
        print "all good"
    except TypeError:
        print "incompatible types!"
    except ZeroDivisionError:
        print "div by zero!"
        raise ValueError()
    finally:
        print "finally"

    print "func1"
    return "done func1"

def main(x):
    try:
        x = func1(x, ["c", 2, 1, "0", "7"])
        print "success"
    except TypeError:
        print "wrong"
    except IndexError:
        print "bad index"
    except Exception:
        print "oops"
    return x

print main(3)
print main(2)
print main(0)
2. Trace through the following code. What does it print?

def func1(x,lis):
    print "lis[\(x\)]: " + lis[x]
    if x > 0:
        x = x - 1
    else:
        x = x + 6
    try:
        integer = int(lis[x])
        print "div: " + str(5//integer)
        print "all good"
    except TypeError:
        print "incompatible types!"
    except ZeroDivisionError:
        print "div by zero!"
        raise ValueError()
    finally:
        print "finally"
    print "func1"
    return "done func1"

def main(x):
    try:
        x = func1(x, ["c",2,1,"0","7"])
        print "success"
    except TypeError:
        print "wrong"
    except IndexError:
        print "bad index"
    except Exception:
        print "oops"
    return x

def other():
    x = 5 // 0
    try:
        print x
    except ZeroDivisionError:
        print "another zero"
    finally:
        print "another finally"
    return x

print main(1)
print main(4)
print other()
3. Trace through the following code. What does it print?

```python
q = [0, 1]
z = 5
y = [3, q, z, 5]
print y
q.append(3)
q[0] = -1
print q
print y
```

4. Trace through the following code. What does it print?

```python
def star(q, y):
    q[0] = y[1]
    q = y
def foo(a, y):
    a = [6, 2]
y[0] = a
q = 3
y.append(q)
a[0] = 0
return q

dict = {6: "bird", "a": 3, 4: 5}
q = [7, 1]
z = 4
y = [3, z, 5, q]
print y
q.append(3)
q[0] = 11
print q
print y

dict[4] = foo(q, y)
print q
print y
print dict.values()
print dict.keys()
q = [1, 2]
y = [q, 8, 9]
print star(q, y)
print q
print y```
5. Trace through the following code. What does it print?

```python
def func1(x, y, lis):
    print lis[y:x]
    lis[x] = lis[y]
    lis = [x]
    return lis

def func2(a, b, nums):
    print a in nums.keys()
    print nums[b]
    nums[1] = 2

def main(x):
    x = x + 1
    lis = [4, 6, 8, 9, 10]
    things = [7, lis, x]
    lis = func1(x, 2, lis)
    print lis
    print things
    if x == 3:
        dict = {x:x, 1:lis, "c": "dog", 0:6}
        print func2('x', 0, dict)
        print dict[1]

main(2)
main(1)
main(3)
```
6. Trace through the following code. What does it print?

def func1(x, lis):
    print "lis[x]: " + lis[x]
    if x > 0:
        x -= 1
    else:
        x += 6
    try:
        integer = int(lis[x])
        print "div: " + str(5//integer)
        print "all good"
    except TypeError:
        print "incompatible types!"
    except ZeroDivisionError:
        print "div by zero!"
    raise ValueError()
    print "func1"
    return "done func1"

def main(x):
    try:
        print func1(x, ["c", 2, "1", "0", "7"])
        print "success"
    except TypeError:
        print "wrong"
    except Exception:
        print "oops"

main(2)
main(0)
main(1)
main(4)
7. Trace through the following code. What does it print?

```python
def yellow(x, a):
a = a + 1
x.append(a)
return x
def green(x, y):
x[1] = 11
y[0] = x[0]
x = [8, 8]
return x
def blue(a, b):
b[0] = b[1]
a = b
print b

x = [6, 7]
y = [7, 8]
z = 2
q = [x, [8, 9], z]
print yellow(y, z)
z = z + 1
print x
print y
print z
print q
y = [7, 8]
green(x, y)
print x
print y
print z
x = blue(x, y)
print x
print y
print z
```
8. Trace through the following code. What does it print?

def func1():
    print "hi"
def foo(x):
    x = x + 1
    print (x + 1)
    return x
def func2(x,y,z):
    x = x + 1
    foo(x)
y[0] = z
    print x
    print z
    z[1] = 9
    print y
    z = [4,6]
y = [2,3]
    return z

x = 3
z = [1,x]
y = [8,7]
y = func2(2,y,z)
print y
z[1] = 3
y = [0,2]
q = [3,3]
print func2(x,y,q)
print x
print y
print z
print q
print func1()
Below are some examples of written questions. For further review, please go over assessments 4 and 5: there WILL be questions like these on both the paper final and the live coding portion.

9. Write a function called `calculate` that takes as arguments a string and a substring, and replaces every instance of the substring with the string `HH`. For example, if the input is `abhelloabcatttbqabe` and the substring was `ab`, the function will return the string `HHhelloHHcatttbqHHe`. Substrings will be of length three or less. Remember that strings are immutable. You may not use any built-in functions/methods besides `len()` and `.append()` – you may NOT use `.replace()`.

10. Write code (or pseudo code) for a method in a class called `check52` that checks if a private attribute called `myList`, has of the two numbers in the list sum to 52. The function will return `True` or `False`, according to this property of the incoming list. A number may NOT be added to itself to get this sum; i.e. simply having 26 in the list once does NOT mean the function should return `True`. However, a number may be added to another number that happens to appear more than once in the list; for example, if the list contains 26 more than once, the function would return `True`. You may not make any assumptions about the list, other than it will be a list of integers of at least size two. You may NOT use any built-in functions besides `len()`.
11. Write a function that removes every odd number from an incoming list and returns the result. For example, the list [2, 3, 4, 5, 6, 7, 8, 9, 1, 10] would yield [2, 4, 6, 8, 10] You may not use any built-in functions/methods besides `len()` and `.append()` – you may NOT use `.remove()`

12. Write a function that takes a string and a substring, and counts how many times the substring occurs in the string followed by the letter A. For example, for the string abcatdecatAerfcatAcatax and the substring cat, your code would return 2. You may not use any built-in functions/methods besides `len()` and `.append()`
Question 1:

lis[x]: 0
div: 5
all good
finally
func1
success
done func1
wrong
2
lis[x]: c
finally
bad index
0

Question 2:

wrong
1
lis[x]: 7
div by zero!
finally
oops
4
EXCEPTION
(note that this is printing an uncaught exception as the last thing; it looks different in the terminal than what we have here, but we just want you to be able to identify that it’s an uncaught exception; the format doesn’t matter)

Question 3:

[3, [0, 1], 5, 5]
[-1, 1, 3]
[3, [-1, 1, 3], 5, 5]

Question 4:

[3, 4, 5, [7, 1]]
[11, 1, 3]
[3, 4, 5, [11, 1, 3]]
[11, 1, 3]
[[0, 2], 4, 5, [11, 1, 3], 3]
[3, 3, 'bird']
['a', 4, 6]
None
[8, 2]
[[8, 2], 8, 9]
**Question 5:**

[8]
[3]
[7, [4, 6, 8, 8, 10], 3]
False
6
None
2
[]
[2]
[7, [4, 6, 8, 9, 10], 2]
[8, 9]
[4]
[7, [4, 6, 8, 9, 8], 4]

**Question 6:**

lis[x]: 1
div: 2
all good
func1
done func1
success
lis[x]: c
oops
wrong
lis[x]: 7
div by zero!
oops

**Question 7:**

[7, 8, 3]
[6, 7]
[7, 8, 3]
3
[[6, 7], [8, 9], 2]
[6, 11]
[6, 8]
3
[8, 8]
None
[8, 8]
3
Question 8:

5
3
[1, 3]
[[1, 9], 7]
[4, 6]
6
4
[3, 3]
[[3, 9], 2]
[4, 6]
3
[[3, 9], 2]
[1, 3]
[3, 9]
hi
None

Question 9:

def calculate(string, substring):
    result = ""
    ctr = 0
    while ctr < len(substring):
        if string[ctr:ctr_len(substring)] == substring:
            result += "HH"
            ctr += len(substring) - 1
        else:
            result += string[ctr]
            ctr += 1
    return result

Question 10:

def check52(self):
    result = False
    ctr1 = 0
    while ctr1 < len(self.__myList):
        ctr2 = 0
        while ctr2 < len(self.__myList):
            if ctr1 != ctr2 and self.__myList[ctr1] +
            self.__myList[ctr2] == 52:
                result = True
            ctr2 += 1
        ctr1 += 1
    return result
Question 11:

def remove(listIn):
    result = []
    ctr = 0
    while ctr < len(listIn):
        if listIn[ctr] % 2 == 0:
            result.append(listIn[ctr])
        ctr += 1
    return result

Question 12:

def check(string, substring):
    result = 0
    ctr = 0
    while ctr < len(string):
        if string[ctr:ctr+len(substring)] == substring + "A":
            result += 1
            ctr += len(substring) - 1
        else:
            result += string[ctr]
        ctr += 1
    return result