BBM 101 – Introduction to Programming I - Fall 2015
Final Exam
January 8, 2016

Name: _____

Student ID number: _____ Section: _____

Problem	Points	Grade
1	6	
2	8	
3	6	
4	9	
5	10	
6	8	
7	8	
8	8	
9	8	
10	6	
11	5	
12	10	
13	8	
Total	100	

INSTRUCTIONS

- Do not open this exam booklet until you are directed to do so. Read all the instructions first.
- When the exam begins, write your name on every page of this exam booklet.
- The exam contains **thirteen** multi-part problems. You have **120 minutes** to earn 100 points.
- The exam booklet contains **7 pages** to the exam, including this one.
- This exam is a **closed book and notes exam**.
- Show all work, as partial credit will be given. You will be graded not only on the correctness and efficiency of your answers, but also on your clarity that you express it. Be neat.
- Good luck!

Q1. (*6 points*) What does the following code print?

```
def f(L):
    result = []
    for e in L:
        if type(e) != list:
            result.append(e)
        else:
            return f(e)
    return result

print f([1, [[2, 'a'], ['a','b']], (3, 4)])
```

Q2. *(8 points)* For each of the following expressions, write the value expression evaluates. Special cases: If an expression evaluates to a function, write Function. If evaluation would never complete, write Forever. If an error would occur, write Error. Assume that the expressions are evaluated in order. Evaluating the first may affect the value of the second, etc. Assume that you have started Python and executed the following statements:

```
jan = [1, 3, 5]
feb = [3, 5, 7]

def mar(apr, may):
   if not apr or not may:
         return []
   if apr[0] == may[0]:
         return mar(apr[1:], may[1:]) + [apr[0]]
   elif apr[0] < may[0]:
         return mar(apr[1:], may)
   else:
         return mar(apr, may[1:])
```

Expression	**Evaluates to**
`5*5`	`25`
`feb[jan[0]]`	
`mar(feb, jan)`	
`feb`	
`len(mar(range(5, 50), range(20, 200)))`	

Q3. *(6 points)* We want to compress strings that have long sequences of equal characters. For example, we want to compress `'bbbbaaa$$$$$$$$$$$$$$$$d'` to `'b4a3$16d1'`. In the compression, each sequence of equal characters is given by the character followed by the length of the sequence. Complete the definition of the function that performs the compression. Use no loops; use only recursion. Do not implement eq chars.

```
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."""

    num = eq chars(s,0)
    if num == len(s):
          return _____________________________________
    else:
          return _____________________________________
```

Q4. *(9 points)* Below are two function definitions using `assert` and `try-except`. Write out the series of print statements displayed for each of the given function calls.

```
def first(n):
    print 'Start first'
    try:
         second(n)
         print 'In first try'
    except:
         print 'In first except'
    print 'Done first'
```

```
def second(n):
    print 'Start second'
    try:
        assert n <= 0
        print 'In second try'
    except:
        print 'In second except'
    assert n >= 0
    print 'Done second'
```

Function calls:

i. `first(-1)`	ii. `first(1)`	iii. `first(0)`

Q5. *(10 points)* Indicate the output produced by the following C program.

```
#include<stdio.h>

int main() {
    int a[5] = { 1, 2, 3, 4, 5};
    int *p, *q, i;
    p = a+4;
    q = &a[1];
    *p += *q;
    p-=3;
    if (p==q) {
        *p = 12;
        q[2] = p[2]/2;
    }
    else {
        *p = 4;
       q[2] = p[2]/4;
    }
    p[-1] += 2;
    q[1] = 8;
    for (i =0; i<5; i++)
         printf("%d ",a[i]);
    printf("\n");
    return 0;
}
```

Q6. (*8 points*) What does the following code print?

```
def f1():

    def f2():
        global x
        x = 9
        print x
        x = 7
        f4()

    def f4():
        x = 0
        print x

    x = 1
    print x
    f2()

def f3():
    f1()
    print x

x = 10

f3()
```

Q7. *(8 points)* Indicate the output produced by the following C program. Note that in the C programming language, there is no Boolean data type. Every number other than zero is considered Boolean value *true*, whereas the number zero is interpreted as the Boolean value *false*.

```
#include <stdio.h>

int p(int x) { return (x-1); }
int q(int x) { return (x*x); }

void f(int *a, int b, int (*g) (int i)) {
   if (b<3) {
      *(a+b) = (*g)(b)-1;
      a[b-1] += 2;
   }
   else
      *(a+b) = (*g)(b+1);
}

int main(void) {
   int x[4] = {0, 1, 2, 0};
   int i;
   for(i=0;i<4;i++)
      if (x[i])
          f(x, x[i], p);
      else
          f(x, i+1, q);
   for(i=0;i<4;i++)
       printf("%d ", x[i]);
   printf("\n");
   return 0;
}
```

Q8. (*8 points*) What does the following code print?

```
# file.seek(offset[, whence]): Set the file's current position, like
# stdio's fseek(). The whence argument is optional and defaults to
# os.SEEK_SET or 0 (absolute file positioning); other values are
# os.SEEK_CUR or 1 (seek relative to the current position) and os.SEEK_END
# or 2 (seek relative to the file's end). There is no return value.

myFile = open("myFile.txt", "w+")
myFile.write("Python is great!")

myFile = open("myFile.txt", "r+")
myFile.seek(-6, 2)
myFile.write("really ")

myFile = open("myFile.txt", "r")
for aLine in myFile:
    print(aLine)

myFile.close()
```

Q9. (*8 points*) What does the following code print?

```
list1 = [4, 2, 3, 1]
list2 = list1[2:]
list1.sort()
list3 = list2
list1.remove(2)
list2.append(5)
print list1
print list2
print list3
```

Q10. (*6 points*) Consider the following code running on a 64-bits computer system where integer and double numbers are represented with 4 and 8 bytes, respectively. If the address of `d` has the value of `804512`, what would would be the output of the code?
Hint: Incrementing the value of a pointer yields a greater address position.

```
int *p;
double d = 8;

p = &d;
p = p + 1;
printf("p = %ld\n", p);
p = (int*)((char*)p + 4);
printf("p = %ld\n", p);
```

Q11. (*5 points*) What does the following code print?

Hint: The *break* statement terminates the loop statement and transfers execution to the statement immediately following the loop. The *continue* statement causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating. The *pass* statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.

```
for num in range(1, 20):
    if num % 7 == 0:
        break
    if num % 4 == 0:
        continue
    if num % 2 == 0:
        print "Even"
    if num % 2 == 1:
        print "Odd"
    if num % 3 == 1:
        pass
        # I will figure later what to do
```

Q12. *(10 points)* For each of the expressions in the table below, write the output displayed by the interactive Python interpreter when the expression is evaluated. The output may have multiple lines. Expressions are evaluated in order, and expressions may affect later expressions.

Whenever the interpreter would report an error, write Error. If execution would take forever, write Forever. Assume that you have started Python and executed the following statements:

```
def f1(b):
   print(len(b))
   b[1].append(b)
   return b[0:1]

def f2(s):
   return s.pop()

x = [3, [4,2],5]
y = [3,[3, 4], [6, 8], [5,6]]
z = [y, y]
```

Expression	**Interactive output**
`x.pop()`	`5`
`z[0].pop()`	
`y[y[1][1]-y[2][0]]`	
`len(f1(y))`	
`y[1][2][1][1]`	
`f2(f2(z))`	

Q13. (*8 points*) What does the following code print?

```
def f(s):
    if len(s) <= 1:
        return s
    return f(f(s[1:]))+s[0]
    #Note that there is double recursion here

print f('bab')
print f('baba')
```