Term Exam 1: Wednesday February 13th, 2013
Instructor : Osmar Zaïane

Student Name :_____

Student ID :_____

- Do not open this exam until you are instructed to do so. Read the instructions.
- The duration of the exam is 50 minutes.
- There are 4 sections worth a total of 100 points.
- This midterm exam will count for 10% of the overall course grade.
- Read all questions carefully. Do not read in diagonal. You may miss things.
- Use a pen not a pencil. You can use a pencil but then can't challenge the marking.
- For full grades, answer all parts of all questions.
- Be concise and give clear and legible answers.
- Non-legible answers will not be marked.
- Cheating is a serious offence in the code of student behaviour.
- No books, notes or other aids are permitted during the exam.
- Good luck!

Section 1	Section 2	Section 3	Section 4	**Total**

Section 1: Complexity [25 points]

Given three lists table1, table2 and table3, guess what this function in python is doing, evaluate the exact time complexity in terms of number of accesses and the big-O time complexity assuming the length of table1 is N and the length of table2 is M.

```
def comeTogether(table1, table2):
    i=0
    table3=[]
    while i<len(table1) and i<len(table2):
        table3.append(table1[i]+table2[i])
        i+=1

    j=i
    while i<len(table1):
        table3.append(table1[i])
        i+=1

    while j<len(table2):
        table3.append(table2[j])
        j+=1

    return table3
```

Exact time complexity:

Justify your answer:

Big-O:

What would be the output of `print(comeTogether([1,2,3,4,5],[6,7,8,9]))`

Section 2: Short questions [25 points]

1- What is the big-O time complexity of the following piece of code:

```
for i in range(3*n):
     for j in range(i+10):
          data[i]=j*data[j]
```

Answer:

2- After the following statements, what are the values of both variables?

```
x = 2013
y = x
x= 13
```

- ☐ a- x is 2013 and y is 2013
- ☐ b- x is 13 and y is 13
- ☐ c- x is 2013 and y is 12
- ☐ d- x is 13 and y is 2013

3- After the following statements, what are the values of both variables?

```
 a=[1,2,3,4]
b=a
a.append(5)
```

- ☐ a- a is [1,2,3,4] and y is [1,2,3,4]
- ☐ b- a is [1,2,3,4,5] and y is [1,2,3,4,5]
- ☐ c- a is [1,2,3,4,5] and y is [1,2,3,4]
- ☐ d- a is [1,2,3,4] and y is [1,2,3,4,5]

4- Given a list of items of size **n** and given five algorithms that process this list to produce the same output, but all having a different run-time complexity. If we want to select the fastest one which one should it be?

- ☐ a- log linear
- ☐ b- linear
- ☐ c- exponential
- ☐ d- logarithmic
- ☐ e- polynomial

5- The following expression on the python command line would produce which output?

```
>>> 7//2
```

- ☐ a- Raise an exception because operator is following another without operand
- ☐ b- 3.5 because it will assume it is a division
- ☐ c- 4 because it will round to the highest integer after the division
- ☐ d- 3 because it will produce and integer division

6- Which statement is wrong about the ADT **Bag**?

- ☐ a- It is a container of items
- ☐ b- There is no implied order to the items
- ☐ c- duplicates are not allowed
- ☐ d- items can be added, removed and accessed individually

7- Which statement is correct regarding what this python code would generate?

```
>>> s={1,2,3,4}
>>> s.append(6)
```

- ☐ a- would generate an error because s is a set and sets are immutable in python
- ☐ b- would generate an error because append() is not part of the interface of a set
- ☐ c- would generate an error because the item 5 is missing in the set
- ☐ d- would generate {1,2,3,4,6}

8- What are the numbers written by the following piece of code:

```
for i in range(2):
     for j in range(2):
          print (i,j, end=" ")
```

Answer:

9- Write the python code to invoke the method **foo** on the object **obj** with the argument **bar**:

Answer:

10- Give an example of an accessor method for the data structure **Stack**:

Answer:

Section 3: Python programming [25 points]

1- These two functions prodList1() and prodList2() are supposed to return the product of the values stored in a list of integers. The first one iterates over the elements of the lists while the second calls a recursive process to produce the product. Neither of these functions works. Figure out what is wrong or missing and fix the problem by adding your code.
Hint: There is one line (or at most two consecutive lines) missing for each block of code.

```
def prodList1(aList):



    for elements in aList:


        result *= elements


    return result
```

```
def prodList2(aList):


    result=cprod(aList,0, len(aList)-1)


    return result

def cprod(aList, begin, end):



    return aList[begin]*cprod(aList,begin+1,end)
```

2- Both functions prodList1() and prodList2() assume the list received as parameter contains at least one element. We should make sure the list contains at least one element. Write the one line to add at the beginning of the function, required to raise an exception in case we have no elements in the list with the message "List is empty". The same line is to be used for both functions.

Answer:

Section 4: Stacks and Exceptions [25 points]

1- Consider these functions:

```
def fact(n):
    '''Return n!'''
    if n == 0:
        return 1
    else:
        return n * fact(n - 1)

def fib(n):
    '''Return the nth Fibonacci number.'''
    if n == 0 or n == 1:
        return n
    else:
        return fib(n - 1) + fib(n - 2)
```

1-a- If **fact(2)** is called, how many calls to **fact()** will be on the call stack when n is 0?

- ☐ a- 0
- ☐ b- 1
- ☐ c- 2
- ☐ d- 3
- ☐ e- 4

1-b- If **fib(2)** is called, how many calls to **fib()** will be on the call stack when n is 0?

- ☐ a- 0
- ☐ b- 1
- ☐ c- 2
- ☐ d- 3
- ☐ e- 4

1-c- if **fib(3)** is called, how many calls to **fib()** are there in total? Hint: draw the tree of calls.

- ☐ a- 2
- ☐ b- 3
- ☐ c- 4
- ☐ d- 5
- ☐ e- 6

2- Consider the following code where avgReport() calls avg() which calls sum():

```
def sum(aList):
    sum= 0
    for v in aList:
        sum = sum + v
    return sum

def avg(aList):
    try:
        s= sum(aList)
        return float(s)/len(aList)
    except TypeError as ex:
        return "Non-Numeric Data"

def avgReport(aList):
    try:
        m= avg(aList)
        print ("Average+15%=", m*1.15)
    except TypeError as ex:
        print ("TypeError: ", ex.args)
    except ZeroDivisionError as ex:
        print ("ZeroDivisionError: ", ex.args)
```

2-a- the functions receive a list of numbers. However, if this list contains an element that is not a number, the function sum() would raise a TypeError exception because of the addition. What would happen then?

- ☐ a- The program would crash because sum() does not handle exceptions
- ☐ b- The program would continue and ignore the non-number element
- ☐ c- avg() would catch an exception anyways because float(s) would raise it
- ☐ d- avg() would catch it because it calls sum() in a try block and handles TypeError
- ☐ e- avg() would catch it first but the returned value would raise another TypeError exception that avgReport() would catch
- ☐ f- avgReport() would catch it because it is the main caller and it handles TypeError

2-b- if the provided list is empty, the execution would be acceptable for sum() and no exception would be raised in sum(). However, there would be an exception later. Which function would raise the exception? What kind of Exception is it? Which function would catch it if it is caught?

Function that raises the exception: Type of exception: Function that catches the exception:

3- Assume this pseudo-code where “Statement i”, among other things, prints “i”

```
for i in range(2)
    Statement 1
    try
        Statement 2
        Statement 3
    except Error 1:
        Statement 4
    except Error 2:
        Statement 5
    else:
        Statement 6
    finally:
        Statement 7
```

3-a- What would be the output if Statement 2 raises Error 1 each time it is executed?

- ☐ a- 1 2 3 4 5 6 7
- ☐ b- 1 2 4 7 1 2 4 7
- ☐ c- 1 2 3 4 7 1 2 3 4 7
- ☐ d- 1 2 3 6 7 1 2 3 6 7
- ☐ e- 1 2 7

3-b- What would be the output if Statement 2 raises Error 3 each time it is executed?

- ☐ a- 1 2 3 4 5 6 7 error bubbles up
- ☐ b- 1 2 4 7 1 2 4 7
- ☐ c- 1 2 3 4 7 1 2 3 4 7
- ☐ d- 1 2 3 6 7 1 2 3 6 7
- ☐ e- 1 2 7 error bubbles up

3-c- What would be the output if Statement 2 doesn’t raise any errors?

- ☐ a- 1 2 3 4 5 6 7 error bubbles up
- ☐ b- 1 2 4 7 1 2 4 7
- ☐ c- 1 2 3 4 7 1 2 3 4 7
- ☐ d- 1 2 3 6 7 1 2 3 6 7
- ☐ e- 1 2 7 error bubbles up