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
```

For each index up to the size of the smallest among table1 and table 2, elements from table 1 and table 2 are added-up and stored in table 3 in the same index. The remaining elements of the longest array between table 1 and table 2 are simply copied to table 3 at the same index.

Exact time complexity: 3*Min(N,M) + 2*(Max(N,M)-Min(N,M))

Justify your answer:
Each element in table1 and table2 are copied to fill table3. Initially elements in table 1 and table 2 are added up to fill table 3 up the smallest of table 1 and table 2, thus 3 accesses min(N,M) times. After that the remaining of the longest table is simply copied into table 3, thus 2 accesses max(N,M)-min(N,M) times. That is 3*min(N,M)+2*(max(N,M)-min(N,M)).

Big-O: acceptable answers: O(N+M) or O(Max(N,M))

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

[7, 9, 11, 13, 5]

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*(3n*(10+11+12+…+3n+9)) ≈ 6* n^2
= O(n^2)

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 (d)

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 b is [1,2,3,4]

√ b- a is [1,2,3,4,5] and b is [1,2,3,4,5] (b)

□ c- a is [1,2,3,4,5] and b is [1,2,3,4]

□ d- a is [1,2,3,4] and b 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 (d)

□ 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 (d)

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 (c, unlike a set, in a Bag duplicates are 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 (b)

□ 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:
0 0 0 1 1 0 1 1

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

Answer:
obj.foo(bar)

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

Answer:
peek()

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):

    result=1

    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):

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

The stop condition can also be expressed:

if begin>end: return 1.0

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:

assert len(aList)>0, "List is empty"

Section 4: Exceptions [25 points]

1- 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)
```

1-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 (e)

□ f- avgReport() would catch it because it is the main caller and it handles TypeError

1-b- if the provided list is empty, the execution would be acceptable for sum() and no exception would be raised. 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: avg()

Type of exception: ZeroDivisionError

Function that catches the exception: avgReport()

2- 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
```

2-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 (b since else and the except of error 2 are skipped)

□ 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

2-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 (e since error 3 is not caught)

2-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 (d since else and finally are executed)

□ e- 1 2 7 error bubbles up

CMPUT 175: Foundations of Computing

Term Exam 1: Wednesday February 10, 2016
Instructor : Dr. Osmar Zaïane or Dr. Anup Basu

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 15% 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, phones, or other aids are permitted during the exam.
- Good luck!

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

Section 1: Tracing and Algorithm Complexity [25 points]

Consider the following function which receives an Integer and prints a number of lists. Trace this python program to answer the following questions.

```
def myFunction(n):
    row = []
    for i in range(n):
       for j in range(n):
           if (i==j):
              row.append(1)
           else:
               row.append(0)
        print(row)
     return
```

What would be the output of **myFunction(3)**
You can use the back of the sheet as draft.

8

```
[1, 0, 0]
[1, 0, 0, 0, 1, 0]
[1, 0, 0, 0, 1, 0, 0, 0, 1]
```

Evaluate the big-O time complexity of this function myFunction assuming the input is N. Justify your answer.

9

Big-O time complexity:
$O(N^2)$
Justify your answer:
There is one loop in the range of N in which we iterate again in the range of N. As N increases, the time complexity increases in the order of NxN

The purpose of the function above was to write the IDENTITY MATRIX, which has 1s on the diagonal and 0 elsewhere. In case it does not produce the desired result, can you move 1 line of code to a different location to achieve this?
(Just circle the line of code and draw an arrow to indicate where it should be moved.)
The result should be in this format

```
. . .
. . .
. . .
```

8

```
def myFunction(n):
    Row = []
    for i in range(n):
       for j in range(n):
           if (i==j):
              Row.append(1)
           else:
               Row.append(0)
       print(Row)
    return
```

```
for i in range(3):
    row=[]
    for j in range(3):
        if (i==j):
            row.append(1)
        else:
            row.append(0)
    print(row)
```

```
[1, 0, 0]
[0, 1, 0]
[0, 0, 1]
```

Section 2: Multiple choice [30 points]

1- What is the output of the following python statement:

```
print([i*i for i in range(4)])
```

□ a. 0, 1, 4, 9 (1 point for a)

□ b. 1 2 3 4

√ c. [0, 1, 4, 9]

□ d. 0*0 1*1 2*2 3*3

2- What would be the values of L1 and L2 after these statements?

```
L1=[1,2,3]
L2=L1
L1.append(4)
```

□ a. L1== [1,2,3] L2== [1,2,3,4]

√ b. L1== [1,2,3,4] L2== [1,2,3,4]

□ c. L1== [1,2,3,4] L2== [1,2,3]

□ d. L1== [1,2,3] L2== [1,2,3]

3- What is the output of the following python code?

```
v1=50
v2=v1
v1=v1/10
print (v1, v2)
```

□ a. 5, 50 (1 point for a)

□ b. 50, 5.0

√ c. 5.0, 50

□ d. 50 50

4- What does the following python statement produce?

```
"a,b,,,,c".split(',') produces
```

□ a. raises an exception because there are several commas next to each other

□ b. ['a', 'b', 'c',]

√ c. ['a', 'b', '', '', '', 'c']

□ d. ['a', 'b', '', 'c']

5- What would be the value of x after the following statements?

```
     T= (20, True, "Hat", [1,2,3], 5)
     x=T.pop()
```

□ a. 20

√ b. no value. pop() would raise an exception

□ c. 5

□ d. True

6- What is the complexity of the function s?

```
def s(n):
     return n*(n+1)*(n+2)/2
```

□ a. linear time

√ b. constant time

□ c. logarithmic time

□ d. quadratic time

7- What would be returned with the following python code?

```
def s(n):
     return n*(n-1)/2

s(10)
```

□ a. would generate an error

□ b. 45 (1 point for b)

□ c. would loop indefinitely because recursion lacks a stop condition

√ d. 45.0

8- The complexity of the algorithm to compute x^n using the divide and conquer strategy is $\log_2(n)$ because:

□ a. we divide and multiply n times

□ b. we only consider the top right corner of the matrix

□ c. we generate a tree of multiplication

√ d. the number of times we need to divide n by 2 to get 1 is $\log_2(n)$

9- An algorithm is said to be polynomial for which of these complexities?

√ a. $O(n^2)$

□ b. $O(3^n)$

□ c. $O(\log_2 \log_2 n)$

□ d. $O(n \log_2(n))$

10- The postfix notation of the following infix expression is:

3 + 6 * 7 - (2 - 4) * 3 + 2

√ a. 3 6 7 * 2 4 - 3 * - + 2 +

□ b. 3 6 7 2 4 3 2 + * - - * +

□ c. - * + 3 6 7 * - 2 4 + 3 2

□ d. - * + * - + 3 6 7 2 4 3 2

Section 3: Exceptions [15 points]

1- a-We would like the output of the following function to handle all cases that can occur when we divide by 0: (i) If the Numerator is greater than 0, we want to print “+Infinity”; (ii) If the Numerator is less than 0, we want to print “-Infinity”; and (iii) If the Numerator is equal to 0, we want to print “UNDEFINED: Zero divided by Zero”. Fill in the missing code below to make this happen.

```
def DivZERO(a,b):
        try:
                c = a/b
                print(c)
        except ZeroDivisionError:

                if (a>0):
                        print ("+Infinity")
                elif (a<0):
                        print ("-Infinity")
                else:
                        print ("UNDEFINED: Zero divided by Zero")


        return
```

2- a- During the execution of the following code no exception is raised. What are the executed statements?

```
try
    Statement 1
    Statement 2
except ValueError:
    Statement 3
except Exception:
    Statement 4
else:
    Statement 5
finally:
    Statement 6
```

Statement 1, Statement 2, Statement 5, Statement 6

Section 4: Python programming [30 points]

1- Write the code in python to read as input the character Y or N for the question “Do you want to play again? (Y/N)”. The input should be validated.
Hint: You need to iterate until a valid input (either Y or N) is entered. The user should be able to enter either uppercase or lowercase. No exception handling required.

8

```
ch=''
while ch not in 'Y N'.split():
     ch=input("Do you want to play again? (Y/N) ").upper()
```

2- Given the following algorithm, write the equivalent code in python. No exception handling required.

10

```
open file records.txt for reading
read each line
  strip the line
  id, name ← split line on ;
  display id and name
close file
```

```
f=open("records.txt","r")
for line in f:
    line=line.strip()
    id, name = line.split(";")
    print (id, name)
f.close()
```

3- In Lab 4, you were given a python file maze.py in which 2 classes were defined: **Maze** and **MazeSquare**. The constructor of Maze read a file containing the definition of a maze. Moreover the class Maze provided the method **get_start_square()** that returned the starting square of class MazeSquare. It also provided a method **is_finish_square(*square*)** that returned True if *square* (of class MazeSquare) is the finish square of the maze. The MazeSquare class represents a single square in the maze and provides a method **get_legal_moves()** that returns a list of MazeSquare squares that are legal moves from the square this method was invoked.

Complete the following python program based on the following algorithm that uses a stack to test whether a maze is solvable. (lines with . at the start need to be completed)

12

```
1-Add the start square to the stack
2-Repeat the following as long as the stack is not empty
  2.1-Pop a square off the stack and make it the current square
  2.2-If the current square is the finish square then solution found
  2.3-Otherwise get the squares which can be moved to from the current square and add them to the stack
```

```
from maze import Maze
from maze import MazeSquare
from stack import Stack

# This function returns True if the maze has a solution.
def test_maze(maze):
    stack = Stack()
.   stack.push(maze.get_start_square())

    while not stack.is_empty():

.       current_square = stack.pop()

        if maze.is_finish_square(current_square):
            return True

.       for square in current_square.get_legal_moves():
            stack.push(square)


    return False

print("Is it solvable?", test_maze(Maze("mazefile.txt")))
```

CMPUT 175: Foundations of Computing

Term Exam 2: Wednesday March 2nd, 2016
Instructor: Dr. Osmar Zaïane or Dr. Anup Basu

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 25% 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, phones, or other aids are permitted during the exam.
- Good luck!

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

Section 1: Tracing programs and Distance Calculation [24 points]

1- (10pts) Consider the following function which receives a list as input and returns a string. The function assumes the existence of a class Stack, as seen in class implemented with a python standard list. Trace this python program to answer the following three questions.

```
def myFunction(myInput):
    stackA= Stack()
    for i in myInput:
        stackA.push(i)
    queueB = Queue()
    largest = stackA.peek()
    removed=False
    while not stackA.isEmpty():
        current=stackA.pop()
        queueB.enqueue(current)
        if current > largest:
            largest = current;

    while not queueB.isEmpty():
        current=queueB.dequeue();
        if  largest == current and not removed:
            removed = True
        else:
            stackA.push(current)
    return "%s" % (stackA)
```

What would be the output of `print(myFunction([1,2,3,4,5]))`
You can use the back of the sheet as draft.

3 [1, 2, 3, 4]

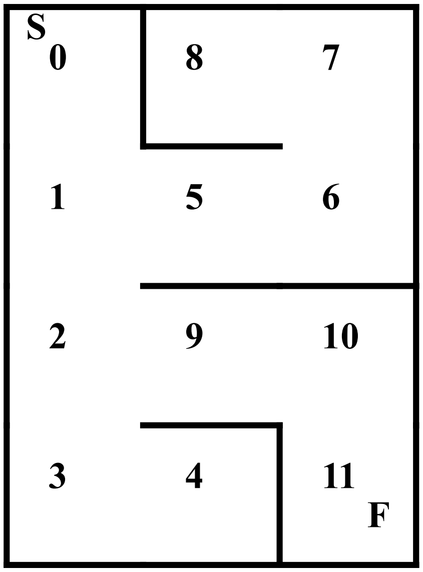
What would be the output of `print(myFunction([4,2,1,5,3,4,5]))`
You can use the back of the sheet as draft.

4 [4, 2, 1, 5, 3, 4]

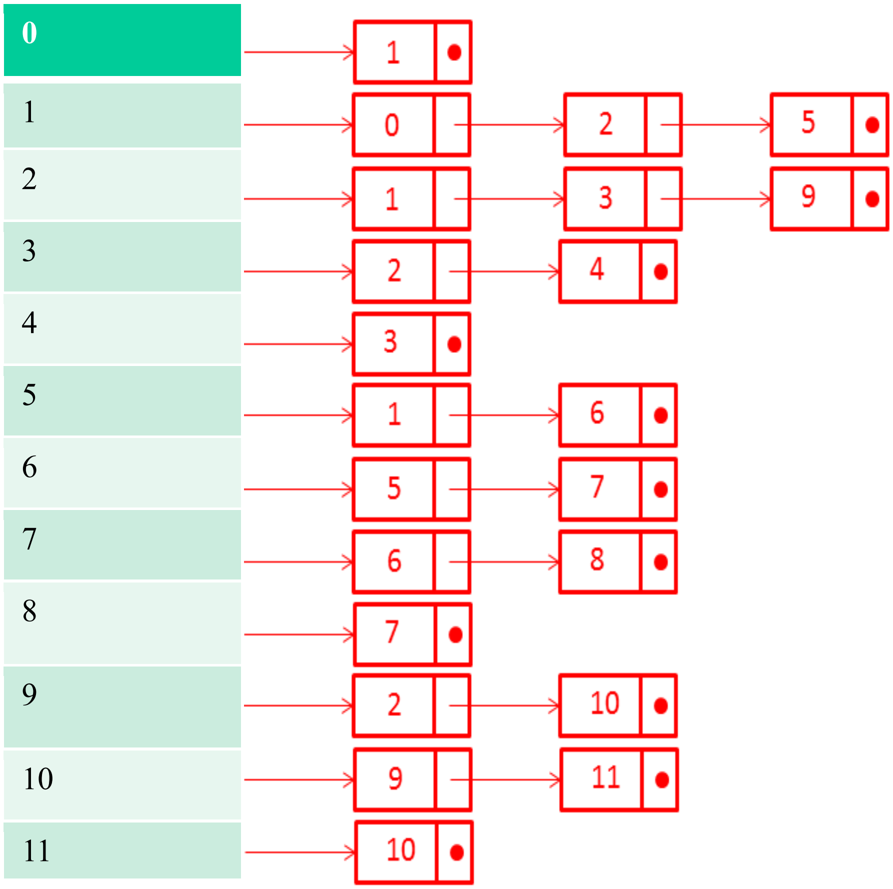
What would be the output of `print(myFunction(["a","g","c","x","s"]))`
You can use the back of the sheet as draft.

3 ['a', 'g', 'c', 's']

2- (14pts) Consider solving the MAZE using an Adjacency list to store positions that are neighbours of other positions. Given the MAZE below, complete by drawing the Adjacency list that follows.



Not linked list -7



Section 2: Multiple choice [36 points] (3 points each)

1- What does the following python statement produce?

```
print('--'.join(['4','2','3','1']))
```

□ a. --4231
□ **b. 4--2--3--1**
□ c. 1--2--3--4
□ d. --1 2 3 4

2- What does the following python statement produce?

```
print('--'.join(['1','2','3',str(min(4, 2, 1))]))
```

□ a. --1 2 3 1
□ **b. 1--2--3--1**
□ c. 1--2--3--4
□ d. --1 2 3 4
□ e. Results is an Exception

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

```
>>> 10//4
```

□ a. Raise an exception because operator is following another without operand
□ b. 2.5 because it will assume it is a division
□ **c. 2 because it will produce an integer division**
□ d. 3 because it will round to the highest integer after the division

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

```
>>> 9%4
```

□ a. Raise an exception because operator is following another without operand
□ **b. 1 because it will print the remainder after integer division**
□ c. 2 because it will produce an integer division
□ d. 3 because it will round to the highest integer after the division

5- What would T contain after the following statements:

```
T= {2, ‘dog’, True, 5.5}
T.add(‘dog’)
```

□ a. would raise an exception because ‘dog’ is already in the set
□ b. {‘dog’, 2, ‘dog’, True, 5.5}
□ c. {2, ‘dog’, True, 5.5, ‘dog’}
□ **d. {2, ‘dog’, True, 5.5}**

6- What would these two python lines of code generate?

```
print("CMPUT","174", end="")
print("*********")
```

□ a. CMPUT174********* 1 point if a (no space before 174)
□ **b. CMPUT 174*********”**
□ c. CMPUT*********174
□ d. CMPUT 174
****

7- After the following statements in python, what would be the value of c?

```
a=175
b="CMPUT"
c="%s-%06d" % (b,a)
```

□ **a. "CMPUT-000175"**

□ b. "CMPUT-175"   1 point if b

□ c. 175CMPUT

□ d. would generate a syntax error

8- One difference between a queue and a stack is:

□ a. Stacks require linked lists, but stacks do not.

□ b. Queues require linked lists, but queues do not.

□ c. Stacks use two ends of the structure; queues use only one.

□ **d. Queues use two ends of the structure, stacks use only one.**

9- If the characters 'X', 'A', 'B', 'C' are placed in a queue (in that order), and then removed one at a time, in what order will they be removed?

□ a. XCAB

□ b. CBAX

□ **c. XABC**

□ d. ABXC

10- Which of the following is true of stacks and queues?

□ a. A stack is a first-in, first-out structure, and a queue is a first-in, last-out structure.

□ b. A stack is a first-in, first-out structure, and both structures are random access structures.

□ c. A stack is a last-in, first-out structure, and a queue is a random access structure.

□ **d. A stack is a last-in, first-out structure, and a queue is a first-in, first-out structure.**

11- Suppose we have a circular queue where we choose to leave one empty cell. Which of the conditions below is a test for an empty queue?

□ a. self._head == (self._tail – 1)

□ **b. self._head == self._tail**

□ c. self._tail == (self._head – 1)

□ d. (self._tail – self._head) == self._capacity

12- In the circular queue implementation, which operations require linear time for their worst-case behaviour?

□ a. dequeue()

□ b. enqueue() when the capacity has not yet been reached

□ c. isEmpty()

□ **d. None of these operations require linear time**

Section 3: Circular Queues [10 points]

Recollect the function in the Circular Queue class that helped us print the queue using "print(queue)" instruction. This is shown below in the top box. This function returns a string representation of a queue where 1, 2, 3, 4 enqueued in sequence as "]1, 2, 3, 4]"; i.e., the front (head) of the queue is on the left and the end of the queue is on the right. We would like to change the function now so the queue is represented as "[4, 3, 2, 1[" instead; i.e., the front (head) of the queue is on the right and the end of the queue is on the left.

```
def __str__(self):
        strExpression="]"
        i=self.__head
        for j in range(self.__count):
            strExpression+=str(self.__items[i])+", "
            i = (i+1) % self.__capacity
        strExpression+="]"
        return strExpression
```

Write the implementation of the __str__() method traversing the queue backwards from the tail.

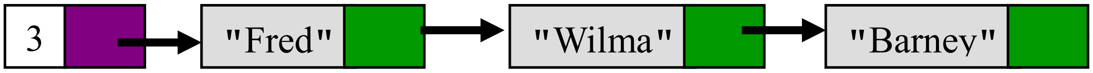
```
def __str__(self):

    strExpression="["

    i=self.__tail - 1 % self.__capacity

    for j in range(self.__count):

        strExpression+=str(self.__items[i])+"  "
        if i != self.__head:
            strExpre3ssion += ","

        i = (i-1) % self.__capacity

    strExpression+="["

    return strExpression
```

Section 4: Programming Linked Lists [30 points]

3- (a) Consider the Singly Linked List below with 3 nodes:

The SLinkedList has 2 fields:
self.size = 3
self.head: Points to First Node

Each node has 2 fields: data & next



Write a method, doubleCut() for the class SLinkedList to delete the first two nodes from the list. In the above case the nodes containing “Fred” and “Wilma” would be removed from the list. Make sure you can delete two nodes. Raise an exception if you can’t.

```
def doubleCut(self):

    if self.size<2:
        raise Exception("List doesn’t have enough elements")


    self.size = self.size-2


    self.head = self.head.getNext().getNext()
```

(b) Write a method *count()* that traverses a singly linked list and counts how many time an item occurs in the list. For instance if the SLinkedList L referred to the list above, L.count("Wilma") should return 1 and L.count("Alfred") should return 0. If L had Fred twice, L.count("Fred") would return 2.

```
def count(self, item):

    current = self.head

    howmany = 0


    while current != None:


        if current.getData() == item:


            howmany +=1


        current = current.getNext()


    return howmany
```

Term Exam 2: Wednesday March 11th, 2020
Version: A
Instructor: Osmar Zaiane

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 20% 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. **Do not use red ink.**
- 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.
- No smartphone or cellphone or PDAs are allowed.
- Good luck!

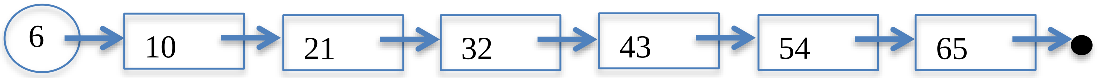
Section 1	Section 2	Section 3	Section 4	**Total**
/20	/30	/20	/30	

Section 1: Tracing [20 points]

Consider the following python code for a method sn() in the class ULinkedList. This class is very similar to the SinglyLinkedList class you know, except the node, in addition to the next reference has always an Integer for the data. The instance of the class caches the size and references head, the first node in the list. In addition, the interface has other methods like sn(). The receiver of sn() is an instance of this ULinkedList class.

```
def sn(self, x, y):
    if x == y:
        return
    px = None
    cx = self.head
    while cx != None and cx.getData() != x:
        px = cx
        cx = cx.getNext()
    py = None
    cy = self.head
    while cy !=None and cy.getData() != y:
        py = cy
        cy = cy.getNext()

    if cx == None or cy == None:
        return

    if px != None:
        px.setNext(cy)
    else:
        self.head = cy
    if py != None:
        py.setNext(cx)
    else:
        self.head = cx

    temp=cx.getNext()
    cx.setNext(cy.getNext())
    cy.setNext(temp)
```

What would be the state of **`my_list`** if we invoke **`sn(21,54)`** when **`my_list`** is the following? Draw the **`my_list`** with consecutive nodes after the method terminates.



(6) → [10]→[54]→[32]→[43]→[21]→65→.   Nodes 21 and 54 are swapped.   8 pts

What would have been the result on the original list if we invoked **`sn(10,21)`**? Draw the **`my_list`** with consecutive nodes after the method terminates.

(6)→[21]→[10]→[32]→[43]→[54]→[65]→. Nodes 10 and 21 are swapped. 8 pts

What would have been the result on the original list if we invoked **`sn(65,73)`**? Draw the **`my_list`** with consecutive nodes after the method terminates.

(6)→[10]→[21]→[32]→[43]→[54]→[65]→. No change in the list (73 not in list) 4 pts

Section 2: Multiple choice [30 points]

For each question given below, circle ONE answer.

1- What block is always executed, independently of a exception being raised?

a. finally
b. else
c. except
d. try

2- The condition to stop for the sequential search

a. is when we find the item we are searching for
b. is when we find the item we are searching for or we reached the end of the list
c. is when we reach the end of the list
d. is when current is None

3- A circular Queue follows

a. A Chronological order
b. LIFO (last In First Out
c. FIFO (First In First Out)
d. A Linear order

4- Each node of a linked list contains two properties, one refers to a data element and the second one is:

a. A reference to an integer
b. A reference to a node
c. A reference to a class
d. A reference to another list

5- Which statement is True
a. The statements after the raise keyword in a program are executed after the handling of the exception
b. An exception can be caught and re-raised to a calling method.
c. The finally clause is executed only if an exception is raised even if no except matches it.
d. Except Exception as e does not catch an AssersionError because e refers to Exceptions

6- Which data structure does not allow insertion of data elements at the rear?

a. stack
b. queue
c. singly linked list
d. doubly linked list

7- When a circular queue is created empty the head and tail are respectively initialized at:

a. 0 and -1
b. 0 and 1
c. -1 and 0
d. 0 and 0

8- If an **unbounded** queue is implemented using a doubly-linked list:

a. its enqueue method is O(1), and its dequeue method is O(1).
b. its enqueue method is O(1), and its dequeue method is O(n).
c. its enqueue method is O(n), and its dequeue method is O(1).
d. none of the above

9- If we do not cache the size of a circular queue, we can check that it is empty by:

a. keeping an empty space, and checking if there is None stored at the tail.
b. keeping an empty space, and checking if the tail modulo the capacity is zero.
c. keeping an empty space, and checking if the head is equivalent to the tail.
d. all of the above.

10- How does the time complexity of the append method compare for built-in Python lists and singly-linked lists?

a. The built-in Python list's append method is more efficient.
b. The singly-linked list's append method is more efficient.
c. Both have the same time efficiency.
d. No way to know.

Section 3: Circular Queue and Exceptions [20 points]

1- Exceptions: In the following code segment, suppose that **Statement2** and **Statement5** raise NameError exceptions, if they are executed:

```
try:
    Statement1
    Statement2
    Statement3
except NameError:
    Statement4
    Statement5
except Exception:
    Statement6
else:
    Statement7
finally:
    Statement8
Statement9
```

Answer <u>Yes</u> OR <u>No</u> to the following questions :

a.	Will `Statement3` be executed?	**No [1 mark]**
b.	Will `Statement4` be executed?	**Yes [1 mark]**
c.	Will `Statement6` be executed?	**No [1 mark]**
d.	Will `Statement8` be executed?	**Yes [1 mark]**
e.	Will `Statement9` be executed?	**No [1 mark]**

2- A **circular queue** is partially implemented below (see Cheat sheet), using a built-in Python list and caching the size of the queue (i.e. the number of elements in the queue). In this implementation, the queue is full if the number of elements in the queue equals the capacity.

Consider the following code segment, which uses the CircQueue class defined on the previous page.

```
served = []
coffeeLine = CircQueue(4)
print(repr(coffeeLine))

coffeeLine.enqueue("Doc")
coffeeLine.enqueue("Sneezy")
served.append(coffeeLine.dequeue())
coffeeLine.enqueue("Sleepy")
served.append(coffeeLine.dequeue())
coffeeLine.enqueue("Happy")
coffeeLine.enqueue("Grumpy")
served.append(coffeeLine.peek())

print(repr(coffeeLine))
print(served)
```

a. What output is displayed when the above code segment is executed? [15 marks]

```
[None, None, None, None], H=0, T=0 (size=0, capacity=4)
['Grumpy', None, 'Sleepy', 'Happy'], H=2, T=1 (size=3, capacity=4)
['Doc', 'Sneezy', 'Sleepy']
```

[0.5 mark list content, 0.5 head, 0.5 tail, 0.5 size, 0.5 capacity]
[4 marks list content, 1 head, 2 tail (wrap), 1 size, 0.5 capacity]
[3 marks for served content]
[1 format]

Cheat sheet for Section 3 Question 2

```
class CircQueue:
    def __init__(self, capacity):
        assert type(capacity) == int, "Incorrect type"
        assert capacity > 0, "Capacity must be positive"

        self.items = []
        for i in range(0, capacity):
            self.items.append(None)

        self.capacity = capacity
        self.count = 0  #queue size (number of elements in queue)
        self.head = 0
        self.tail = 0


    def enqueue(self, item):
        assert self.count < self.capacity, "Queue is full"
        self.items[self.tail] = item
        self.tail = (self.tail + 1) % self.capacity
        self.count += 1

    def dequeue(self):
        assert self.count != 0, "Queue is empty"
        item = self.items[self.head]
        self.items[self.head]= None
        self.head = (self.head + 1) % self.capacity
        self.count -= 1
        return item

    def peek(self):
        assert self.count != 0, "Queue is empty"
        return self.items[self.head]

    def __repr__(self):
        objStr = str(self.items)
        objStr += ', H=' + str(self.head)
        objStr += ', T=' + str(self.tail)
        objStr += ' (size=' + str(self.count)
        objStr += ', capacity=' + str(self.capacity) + ')'
        return objStr
```

Section 4: Programming (Linked List) [30 points]

1- In the Doubly Linked List, there is a reference to the head and a reference to the tail. Write the *pop(pos)* method of DLinkedList that removes and returns the element at position *pos* in the doubly linked list. If the position is out of bound – i.e. *pos* is larger than the length of the list – an exception AssersionError should be raised inside the method.
Use a variable *current* but there is no need to use a variable *previous*.
The method should leave the list in a consistent state.
You are allowed to invoke other methods of DLinkedList and DLinkedListNode.

```
def pop(self, pos):
    # check if the input is correct

    assert type( pos) is IntType, “position not valid: %r” % pos

    assert pos>=0 and pos<self.size,     “position out of bound”

    # traverse the list
    current = self.head

    count = 0

    while (count < pos ):
        current=current.getNext()
        count += 1

    # case current is first node
    if current.getPrevious() == None:
        self.head = current.getNext()
    else:
        current.getPrevious().setNext(current.getNext())


    # case current is the last node
    if current.getNext() == None:
        self.tail = current.getPrevious()
    else:
        current.getNext().setPrevious(current.getPrevious())


    # adjusting the size
    self.size -=1




    return current.getData()
```

1 pt type of pos
4 pts for assertion
2 pts for starting at head of list
5 pts for while loop and advancing in list and counting
3 pts considering the case node at head
3 pts considering the case node at tail
8 pts removing node (for each pointer 2 points)
2 pts decrementing size
2 pts return data of current
-2 pts using pointer

This page was intentionally left empty as scrap paper.

Term Exam 3: Friday, April 8th, 2016
Instructor: Dr. Osmar Zaïane or Dr. Anup Basu

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 25% 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, phones, or other aids are permitted during the exam.
- Good luck!

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

Section 1: Tracing Program and Searching [27 points]

1- (15pts) Consider the following function `my_search()`. Trace this python program to answer the following three questions.

```
def my_search( data, key ):
    found = False
    low = 0
    count = 0
    high=len(data)-1
    while ( not found and low<=high):
        guess = (high+low)//2
        print('guess: ', guess)
        if ( key == data[guess] ):
            found = True
            count = 1
            k = guess - 1
            while k>=0 and data [k] == key:
                  count += 1
                  k -= 1
            k = guess + 1
            while k < len(data) and data [k] == key:
                  count += 1
                  k += 1
        else:
            if (key < data[guess]):
                high=guess-1
            else:
                low = guess+1
    print('count:', count)
    return count
```

What would be the printed output of *my_search*([1,1,2,2,3,3,3,3,3,3,6,7], 1)?
You can use the back of the sheet as draft.

guess: 5
guess: 2
guess: 0
count: 2

What would be the printed output of *my_search*([1,1,2,2,3,3,3,3,3,3,6,7,7,7,7], 7)?

guess: 7
guess: 11
count: 4

What would be the printed output of *my_search*([1,1,2,2,3,3,3,3,3,3,6,7,7,7,7], 6.5)?

```
guess: 7
guess: 11
guess: 9
guess: 10
count: 0
```

2- (5pts) Given a Sorted List with N elements, and that the key being searched is repeated R times in the list, what is the complexity of *my_search()* in terms of O notation?

$O(\log_2(N) + R)$

3- (4pts) Consider the following variation of the code in Question 1. Now we have a new function called *index_search()* :

```
def index_search( data, key ):
    found = False
    low = 0
    high=len(data)-1
    while ( not found and low<=high):
        guess = (high+low)//2
        if ( key == data[guess] ):
            found = True
        else:
            if (key < data[guess]):
                high=guess-1
            else:
                low = guess+1
    return guess
```

How can you use the "index_search" method to find the number of times key appears in a Sorted List, given that all elements in the list are positive integers? Fill in the missing parts in the One Line code below to make this happen. *The key to be search is a number but does not have to be an integer*.

count = index_search(data, key+0.5) - index_search(data, key-0.5)

key +- alpha where alpha is in]0..1[. here we use 0.5

4- (3pts) Given a Sorted List with N elements, and that the key being searched is repeated R times, what is the complexity of the method in Question 3 in terms of O notation?

$2 \times \log_2(N) = O(\log_2 N)$

Section 2: Hashing [10 points]

Consider External Chaining as the approach of choice for resolving collisions during Hashing. Given a string the hashing function is defined as follows:

HASH(string) = (SUM of the Alphabetical Order of All the Letters in the String) MOD 5

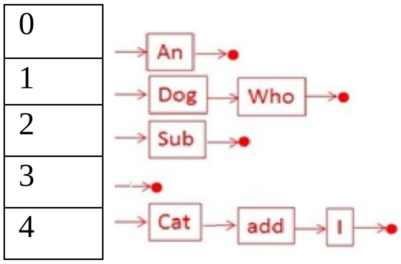
The Alphabetical Orders are: (A or a) = 1; (B or b) = 2; (C or c) = 3; (D or d) = 4; (E or e) = 5; (F or f) = 6; (G or g) = 7; (H or h) = 8; (I or i) = 9; (J or j) = 10; (K or k) = 11; (L or l) = 12; (M or m) = 13; (N or n) = 14; (O or o) = 15; (P or p) = 16; (Q or q) = 17; (R or r) = 18; (S or s) = 19; (T or t) = 20; (U or u) = 21; (V or v) = 22; (W or w) = 23; (X or x) = 24; (Y or y) = 25; (Z or z) = 26

Compute the HASH function for the following strings and show the External Chaining below using Singly Linked Lists:

Cat, Dog, add, Sub, I, Who, An

Cat = 3+1+20=24→ 4
Dog =4+15+7→1
add =1+4+4→4
Sub =19+21+2=42→2
i =9→4
Who =23+8+15→1
An =1+14=15→0

-2 pts If PROD instead of SUM
1 pt for each correct hash
2 pts for correct external chaining
1 pt for link to None at end of list



If PROD instead of SUM, the hash table would be
0 → Cat → Dog → Who → None
1 → None
2 → add → None
3 → Sub → None
4 → I → An → None

Section 3: Multiple Choice [30 points] (3 points each)

1- The worst case complexity of Linear Search is?

□ a. O(log n)

□ b. O(1)

□ c. O(n log n)

□ X d. O(n)

2- The worst case complexity of Binary Search is?

□ X a. O(log n)

□ b. O(1)

□ c. O(n log n)

□ d. O(n)

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

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

□ a- a is [1,2,3] and b is [1,2,3]

□ b- a is [1,2,3] and b is [1,2,3,5]

□ X c- a is [1,2,3,5] and b is [1,2,3,5]

□ d- a is [1,2,3,4] and b is [1,2,3,4,5]

4- Which of the following sorting algorithms is of divide-and-conquer type??

□ a. Bubble Sort

□ b. Insertion Sort

□ X c. Quick Sort

□ d. Selection Sort

5- Which of the following data structure is a linear data structure?

□ a. Trees

□ X b. Stack

□ c. Binary Search Trees

□ d. None of the above

6- Given a list that is already sorted in Increasing Order which of the algorithms discussed in class has O(n^2) complexity?

□ a. Bubble Sort

□ b. Insertion Sort

□ X c. Quick Sort

□ d. Merge Sort

7- Given a list that is already sorted in Decreasing Order which of the algorithms discussed in class has lower than O(n^2) complexity?

□ a. Bubble Sort

□ b. Insertion Sort

□ c. Quick Sort

□ X d. Merge Sort

8- Given two list of size (*n/2*) each that are already sorted, the cost of merging the two lists to create one sorted list of size *n* is at most:

□ X a. O(n)

□ b. n/2

□ c. O(1)

□ d. O(log n)

9- Which of the following is true?

□ a. Best Case complexity of Bubble Sort is O(n^2)

□ b. Best Case complexity of Insertion Sort is O(n log n)

□ c. Worst Case complexity of Merge Sort is O(n^2)

□ X d. Average Case complexity of both Quick Sort and Merge Sort are O(n log n)

10- Given a list of size *n* which we want to Partition in Quick Sort using the First Element as the Pivot, which of the following is true?

□ a. In the Best Case at (n/2) Key Comparisons are needed for Partitioning

□ b. In the Worst Case O(*n log n*) Key Comparisons are needed for Partitioning

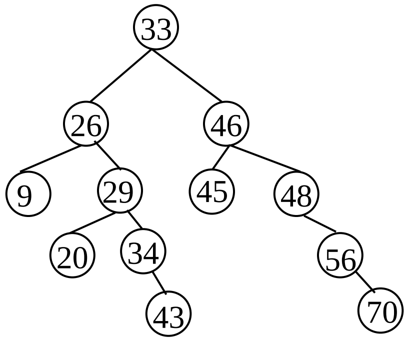
□ X c. In Every Situation Exactly (n – 1) Key Comparisons are needed for Partitioning

□ d. On the Average O(log n) Key Comparisons are needed for Partitioning

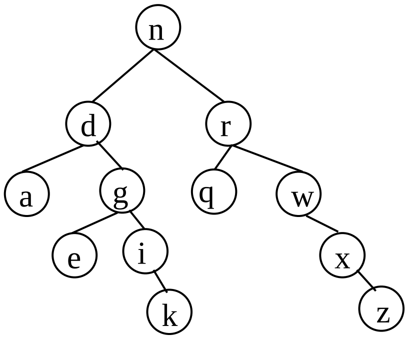
Section 4: Trees [33 points]

1- [14] Consider the following two binary trees:

Tree A



Tree B



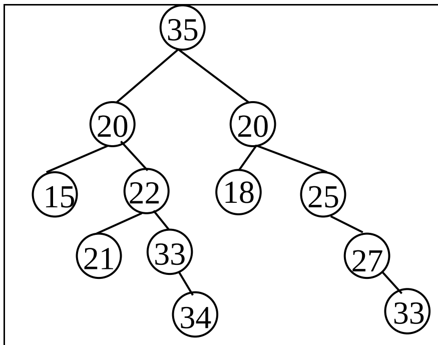
a- (3 pts) Which tree is a binary search tree? B

b- (3 pts) Why is the other tree not a binary search tree? 34 > 33 but on left of 33 also 43 >33 but on left of 33.

c- (3 pts) Give a post-order traversal of Tree A:
9 20 43 34 29 26 45 70 56 48 46 33

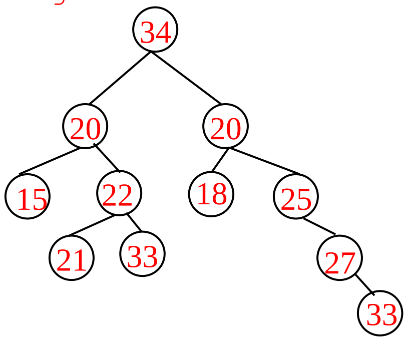
d- (3 pts) Give an in-order traversal of Tree B:
a d e g i k n q r w x z

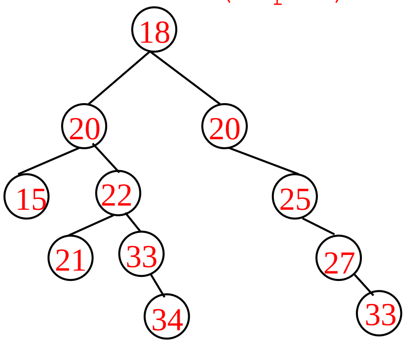
2- (5 points) Redraw the following tree after removing the root (35) and preserve the properties of the BST.



This is not a BST

Tree is not a BST so not possible answer (5pts)
Or give either of these two trees (4 pts)





Or remove node 35 and reorganize tree in BST (5pts)

3- (16 points) Python Programming for Trees.
You are given the following method to Find the Smallest key in a Subtree:

```
def _findSmallest(self,currentNode):
        if currentNode.getLeft():   # there is someone smaller
            return self._findSmallest(currentNode.getLeft())
        else:
            return currentNode.getKey()
```

(a) (5 points) Write the code to Find the Largest key in a Subtree below:

```
def _findLargest(self,currentNode):
        if currentNode.getRight():   # there is someone bigger
            return self._findLargest(currentNode.getRight())
        else:
            return currentNode.getKey()
```

(b) (5 points) Now, write below the code to Find the Range of keys in the Tree (i.e. Largest – Smallest):

```
def _KeyRange(self):
        if self.__root:
          return (self._findLargest(self._root) - self._findSmallest(self._root))
        else:
         return 0
```

(c) (6 points) Write a short Recursive Method to count the Number of Keys that are Even numbers in a subtree:

```
def _CountEven(self,currentNode):
      if currentNode:                          # Initialize any local variables
        Descendents= self._CountEven(currentNode.getLeft())+self._CountEven(currentNode.getRight())
      if    not currentNode:                             # Empty node
        return    0
      elif    currentNode.getKey()  % 2  == 0:             # Key is Even
         return  (1+Descendents)
      else:
         return   (Descendents)
```