Name :____

Code permanent : ____________________________________

Place number : ____________________________________

Directives pédagogiques :

- Write your name, permanent code and place number above.
- **Read all questions and answer directly on the sheets**
- Only a pen/pencil is permitted, **no documentation**, calculator, phone, computer, or any other object
- This exam contains 8 questions for a total of 125 points.
- Points over 100 are considered bonuses. However, pay attention at your time as 1 point = 1 minute. You can thus drop 25 points without penalty.
- This exam contains 15 pages, including 2 pages et the end for your drafts.
- For the developing questions, **write clearly** and **detail your answers**
- You have 100 minutes to complete this exam

GOOD LUCK !

1	/ 20
2	/ 20
3	/ 20
4	/ 15
5	/ 5
6	/ 5
7	/ 20
8	/ 20
Total	/100

Q1. (20) Consider the function “`phantom`” which takes a Python list as an argument:

```
def car( x ):
    return x[0]

def cdr( x ):
    return x[1:]

def phantom( x ):
    if x == []:
        return []
    return phantom( cdr( x ) ) + [car( x )]
```

a) (10) What does the function “`phantom`” does?

b) (5) What are the primitive operations of the function “`phantom`” ?

c) (3) Give a function f(n) indicating how many times each primitive operation is executed for a list of n elements.

d) (2) Give a function g(n) such that f(n) is asymptotically smaller or equal to g(n).

Q2. (20) Suppose you are asked to develop an application for an e-book reader (e-book). Your design must allow at least the users to buy e-books, visualize their list of e-books, and read their e-books.

a) (10) What would be your principal classes and methods?

b) (10) Draw your class hierarchy and methods without writing any code.

Q3. (20) A sequence S contains n - 1 unique integers in the range [0, n-1]. One number of the range is thus missing in S (e.g. for n = 4, the number 2 is missing in the sequence [0,1,3]). Give an algorithm in O(n) in time to find this number. You cannot use more than O(1) additional space, i.e. only one additional variable that can handle a single integer value.

Q4. (15) Consider the Python class `DynamicArray` in Appendix A. Implement (in Python or pseudo-code) the function `pop` which return and remove the last element of a `DynamicArray` and which reduce the capacity of the array by half when the elements in the array occupy less than a quarter of the capacity.

```
#return and remove the last element
def pop( self ):
```

Q5. (5) Suppose an initially empty stack, S, on which the following operations were made: 33 `push`, 9 `top` and 11 `pop` of which 4 generated empty stack exceptions that were caught and ignored. What is the size of stack S?

Q6. (5) Suppose an initially empty queue, Q, on which the following operations were made: 28 `enqueue`, 6 `first` and 7 `dequeue` of which 5 generated empty queue exceptions that were caught and ignored. What is the size of queue Q?

Q7. (20) Given a singly linked list (`SinglyLinkedList`) with sentinels `head` and `last`, as described in Appendix B.

a) (18) Give a non-recursive method to find the middle element of a `SinglyLinkedList`. In the case of an even number of elements, return the element that is slightly on the left of the middle. Your implementation must only use link hopping and must not use a counter.

```
def find_middle( self ):
```

b) (2) What is the running time of this method?

Q8. (20) Fill the table below to show in each row the changes applied to the array when we sort it using heapsort. Use the next page to draw the heaps.

15	9	8	1	4	11	7	12	3

Your heaps here:

Appendix A : Class DynamicArray

```
import ctypes
class DynamicArray:

    #return a pointer to the memory area
    #that contains c Python objects
    def _makeArray( self, c ):
        return( c * ctypes.py_object )()

    #create a 1-element array
    def __init__( self ):
        self._n = 0
        self._capacity = 1
        self._A = self._makeArray( self._capacity )

    #return the number of element in the array
    def __len__( self ):
        return self._n

    #return the capacity of the array
    def capacity( self ):
        return self._capacity

    #return the kth element of the array
    def __getitem__( self, k ):
        if not 0 <= k < self._n:
            raise IndexError( 'invalid index' )
        return self._A[k]

    #dimensionne le tableau à c éléments
    def _resize( self, c ):
        #create an array of c elements
        B = self._makeArray( c )
        #copy the old array's elements in the new one
        for k in range( self._n ):
            B[k] = self._A[k]
        #change the self array reference
        self._A = B
        #adjust the self array capacity
        self._capacity = c
```

Appendix B : Classes `SinglyLinkedNode` and `SinglyLinkedList` with `head` and `last` sentinels.

```
class SinglyLinkedNode:

    def __init__( self, element, next ):
        self.element = element
        self.next = next
```

```
from SinglyLinkedNode import SinglyLinkedNode
class SinglyLinkedList:

    def __init__( self ):
        self._head = None
        self._last = None

    def append( self, element ):
        newNode = SinglyLinkedNode( element, None )
        if self._last == None:
            self._head = self._last = newNode
        else:
            self._last.next = newNode
            self._last = newNode
```