Nom :_____

Code permanent : _____

Numéro de place : _____

Directives pédagogiques :

- Inscrivez votre nom, prénom, code permanent et numéro de votre place.
- Lisez attentivement toutes les questions et **répondez directement sur le questionnaire**.
- Seule l'utilisation d'un crayon est permise, **aucune documentation, calculatrice, téléphone cellulaire, ordinateur, ou autre objet permis**.
- Cet examen contient 8 questions pour 125 points au total.
- Tout point au-dessus de 100 sera considéré comme bonus. Cependant, faites attention au temps car le barème est établi à 1 point par minute. Vous pouvez laisser tomber 25 points sans être pénalisé.
- Cet examen contient 15 pages, incluant 2 pages à la fin pour vos brouillons.
- Pour les questions à développement, **écrivez lisiblement et détaillez vos réponses**.
- Vous avez 100 minutes pour compléter cet examen.

BONNE CHANCE !

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

_____26

février 2014

1. (20) Considérez la fonction "`fantome`" qui prend en argument une liste python :

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

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

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

a) (10) Que fait la fonction "`fantome`" ?

Renverse la liste.

b) (5) Quelle sont les opérations primitives de la fonction "`fantome`" ?

Pour une liste de longeur n :

- **Appels car (n)**
- **Appels cdr (n)**
- **Appels récursif (n)**
- **Concaténation liste (n)**
- **Accès liste (2n)**
- **Test == (n)**

c) (3) Donnez la fonction f(n) représentant combien de fois chaque opération primitive est exécutée pour une liste de n éléments.

f(n) = 7n

d) (2) Donnez une fonction g(n) telle que f(n) est de manière asymptotique plus petite ou égale à g(n).

g(n) = n; f(n) est dans O(n)

2. (20) Supposez qu'on vous demande de développer une application pour un lecteur de livre électronique (livre-e). Votre design doit permettre aux utilisateurs, au minimum, d'acheter des nouveaux livres, voir leur liste de livres achetés et lire leurs livres achetés.

   a) (10) Quelles seraient les classes principales et leurs méthodes ?

Classes : livreE

Méthodes : acheter, visualiser, lire

b) (10) Faites un diagramme hiérarchique de vos classes et méthodes mais n'écrivez aucun code particulier.

livreE		
Acheter	visualiser	lire

3. (20) Une séquence S contient n - 1 entiers uniques dans l'intervalle [0, n-1]. Il y a donc un nombre manquant de cet intervalle dans S (e.g. pour n = 4, il manque 2 dans la séquence [0,1,3]). Donnez un algorithme en temps O(n) pour trouver ce nombre. Vous ne pouvez utiliser qu'un espace additionnel en O(1), i.e. une seule variable additionnelle pouvant contenir un seul entier.

```
def findnum( seq ):
  sum = 0
    for i in range( len( seq ) ):
      sum += seq[i]
  return (( len( seq )+1 ) * len( seq )/2 ) - sum
```

4. (15) Considérez le code Python de la classe `DynamicArray` en Appendice A. Implantez (en Python ou pseudo-code) la fonction `pop` qui retourne et retire le dernier élément d'un `DynamicArray` et qui diminue la capacité du tableau de moitié lorsque les éléments dans le tableau occupent moins que le quart de la capacité.

```
#retourne et retire le dernier élément du tableau
#def pop( self ):
```

```
#retourne et retire le dernier élément du tableau
def pop( self ):
    if self._n == 0:
        return False
    else:
        obj = self._A[self._n - 1]
    self._n -= 1
    if self._n < self._capacity / 4:
        self._resize( self._capacity // 2 )
    return obj
```

5. (5) Supposez une pile initiale vide, S, sur laquelle on a exécuté un total de 33 opérations `push`, 9 opérations `top` et 11 opérations `pop` dont 4 ont levées des erreurs de pile vide attrapées et ignorées. Quelle est la taille de S ?

33 push - (11 - 4) pop = taille de S de 26 éléments

6. (5) Supposez une file (queue) initiale vide, Q, sur laquelle on a exécuté un total de 28 opérations `enqueue`, 6 opérations `first` et 7 opérations `dequeue` dont 5 ont levées des erreurs de file vide attrapées et ignorées. Quelle est la taille de Q ?

28 enqueue - (7 - 5) dequeue = taille de Q de 26 éléments

7. (20) Soit une liste simplement chaînée (`SinglyLinkedList`) avec des sentinelles `head` et `last`, telle que décrite à l'Appendice B.

a) (18) Décrivez une méthode non-récursive pour trouver l'élément milieu d'une `SinglyLinkedList`. Dans le cas d'un nombre paire d'élément, retournez l'élément légèrement à gauche du centre. Votre méthode ne doit utiliser que des parcours de liste et elle ne doit pas utiliser de compteur.

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

```
def find_middle( self ):
  if self._size == 0:
      return False
  else:
      tick = False
      node = self._head
      half = node
      while node.next:
          node = node.next
          if tick:
              half = half.next
          tick = not tick
      return half.element
```

b) (2) Votre méthode est dans quel ordre de temps d'exécution ?

O(n)

8. (20) Remplissez le tableau ci-dessous en montrant à chaque ligne les changements d'états du tableau lorsqu'on le trie par la méthode du monceau (heapsort). Utilisez la page suivante pour dessiner vos monceaux.

15	9	8	1	4	11	7	12	3

26

Vos monceaux ici :

Appendice A : Classe DynamicArray

```
import ctypes
class DynamicArray:

    #retourne un pointeur dans la zone de mémoire
    #qui contient c objets python contigus
    def _makeArray( self, c ):
        return( c * ctypes.py_object )()

    #créé un tableau de 1 élément
    def __init__( self ):
        self._n = 0
        self._capacity = 1
        self._A = self._makeArray( self._capacity )

    #retourne le nombre d'éléments dans le tableau
    def __len__( self ):
        return self._n

    #retourne la capacité du tableau
    def capacity( self ):
        return self._capacity

    #retourne le kème élément du tableau
    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 ):
        #créé un nouveau tableau de c éléments
        B = self._makeArray( c )
        #copier les éléments de l'ancien tableau
        for k in range( self._n ):
            B[k] = self._A[k]
        #changer la référence du tableau self
        self._A = B
        #adjuste la capacité du tableau self
        self._capacity = c
```

Appendice B : Classes `SinglyLinkedNode` et `SinglyLinkedList` avec des sentinelles `head` et `last`.

```
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
```