TD 3 - Les listes et la complexité- Master 1 Bio-informatique

Exercice 1

Ecrivez un programme python qui permet de créer des listes simplement chaînés. Pour cela, vous implémenterez l'api suivante :

```
def create_list():
  """
  Créer et retourne une liste vide.
  """
  return NotImplemented

def create_cell(value, successeur):
  """
  Créer et retourne une cellule contenant la valeur
  ‘valeur‘ et le successeur ‘successeur‘.
  """
  return NotImplemented

def change_cell(cell, valeur, successeur):
  """
  Change les attributs ‘valeur‘ et ‘successeur‘ de la cellule
  ‘cell‘ passée en paramètre.
  """
  raise NotImplementedError()


def push_front(l, e):
  """
  Ajoute l'élément ‘e‘ au début de la liste ‘l‘ : créé une
  nouvelle cellule et l'insère en début de liste.
  """
  return NotImplemented

def remove_front(l):
  """
  Retire le premier élément de la liste si il existe.
  """
  return NotImplemented

def first_cell( l ):
  """
```

```
  Retourne la première cellule de la liste ‘l‘ et None si la liste
  est vide.
  """
  return NotImplemented

def next_cell( cell ):
  """
  Retourne le successeur de la cellule ‘cell‘.
  """
  return NotImplemented

def value_cell( cell ):
  """
  retourne la valeur de la cellule ‘cell‘.
  """
  return NotImplemented
```

Exercice 2

A laide de l'implémentation de la liste chainée précédente, écrivez les fonctions suivantes :

```
def push_back(l, e):
  # Ajoute un élément en fin de liste
def insert_cell( cell, e ):
  # Cree une nouvelle cellule contenant l'élément e et insère cette
  # cellule dans la liste juste après la cellule cell.
def copy( l ):
  # renvoie une copie de la liste l
def milieu(l):
  # renvoie l'élément situé en milieu de liste
def taille(l):
  # renvoie le nombre d'éléments de la liste
def transferer( l1, l2 ):
  # Transfère les éléments de l2 dans l1. La liste l2 devient vide.
def recherche_element(l,e):
  # renvoie la première cellule contenant e ou la cellule de fin de liste.
```

Donnez la compléxité de chacune de ces fonctions (optionnel).

Exercice 3

Reprenez les deux exercices précédent en utilisant des listes doublements chaînées.

Pour aller plus loin, vous pouvez implémenter l'algorithme des pointeurs dansant de Knuth : `https://arxiv.org/abs/cs/0011047`.