Université de Carthage Institut Préparatoire aux Études d'Ingénieur de Nabeul Département : Mathématiques	IPEIN المعهد التحضيري للدراسات الهندسية بنابل Institut Préparatoire aux Etudes d'Ingénieurs de Nabeul	Année universitaire : 2025/2026 Filière : SM/SP/ST Niveau d'étude : 2ème année Semestre : 1 Nombre de pages : 2 pages Date : 22/11/2025 Durée : 1h

TEST Informatique N°1

Nom : ..... **Prénom :** ..... **Classe :** .....

1. Si p est une pile contient [3, 6, 9] (9 sommet) et qu'on fait `p.empiler(4)`, la pile devient :
   (a) ☐ [3, 6, 9, 4]
   (b) ☐ [4, 3, 6, 9]
   (c) ☐ [9, 6, 3, 4]
   (d) ☐ [4, 9, 6, 3]
2. Si f une file contient [3, 6, 9] (9 sommet) et au lieu de faire `File.enfiler(f,x)` , on a fait `Pile.empiler(f, x)`, cela insère x dans f :
   (a) ☐ À la fin
   (b) ☐ Au début
   (c) ☐ Au milieu
   (d) ☐ À une position aléatoire
3. Dans une classe Pile, les elements sont enregistés dans un atribut d'instance de type liste nommé p, par quoi lire le sommet?
   (a) ☐ `p.pop()`
   (b) ☐ `p.insert()`
   (c) ☐ `p[len(pile)-1]`
   (d) ☐ `p[len(p)-1]`
4. Pour une pile p, remplacer `while not p.estVide()` par `while p.estVide()` fait que la boucle :
   (a) ☐ Ne s'exécute jamais
   (b) ☐ Tourne une seule fois
   (c) ☐ Tourne tant que la pile n'est pas vide
   (d) ☐ Tourne tant que la pile est vide
5. Quelle est la sortie de ce code?
   `p = Pile(); p.empiler(1); p.empiler(2); print(p.sommet())`
   (a) ☐ 1
   (b) ☐ 2
   (c) ☐ Erreur
   (d) ☐ None
6. Quel mot-clé spécial permet d'utiliser l'opérateur 'in' sur une instance de type Pile?
   (a) ☐ `__len__`
   (b) ☐ `__contains__`
   (c) ☐ `__eq__`
   (d) ☐ `__add__`
7. Compléter le code pour inverser une file `f` :

```
def inverserfile(f):
  p = Pile()
  while not f.estVide():
    ...
  while not p.estVide():
    f.enfiler(p.depiler())
```

   ☐ f.enfiler(p.depiler())
   ☐ p.empiler(f.defiler())
   ☐ f.depiler(p.enfiler())
   ☐ p.depiler(f.defiler())
8. Une pile vide exécutant `depiler()` provoque :
   (a) ☐ L'insertion d'un None
   (b) ☐ Un arrêt du programme
   (c) ☐ Une exception
   (d) ☐ Une pile inversée
9. Dans une file, remplacer `pop(0)` par `pop()` rend l'ordre :
   (a) ☐ FIFO
   (b) ☐ Non déterminé
   (c) ☐ Trié
   (d) ☐ LIFO
10. On utilise une file f et une pile p. On transfère tous les éléments de f vers p, puis on transfère

tout le contenu de p vers f.
Que devient l'ordre des éléments dans f?

(a) ☐ Il reste inchangé
(b) ☐ Il est inversé
(c) ☐ Les éléments du milieu sont échangés
(d) ☐ Tous les éléments sont supprimés

11. On veut vérifier si une file `f` contient une suite de valeurs qui forme un palindrome. On ne doit pas modifier la file au final. On dispose d'une pile.
Quelle stratégie est correcte?

(a) ☐ Copier `f` dans une pile et comparer les deux séquences
(b) ☐ Dépiler et défiler en même temps pour comparer dans le même ordre
(c) ☐ Trier la file puis comparer
(d) ☐ Aucune des méthodes proposées

12. Si une file contient [5,7,9] et qu'on exécute defiler(), enfiler(4), enfiler(6), elle devient :

(a) ☐ [7, 9, 4, 6]
(b) ☐ [5, 4, 6]
(c) ☐ [9, 4, 6]
(d) ☐ [6, 4, 9]

13. Dans un système de traitement de tâches, les tâches sont ajoutées au fur et à mesure dans une file (FIFO) et traitées dans l'ordre d'arrivée. Que se passe-t-il si on remplace cette file par une pile (LIFO)?

(a) ☐ Les tâches sont toujours traitées dans l'ordre d'arrivée
(b) ☐ L'ordre de traitement est inversé
(c) ☐ Les tâches sont traitées de manière aléatoire
(d) ☐ Impossible de traiter les tâches

14. Compléter le code Python pour inverser une pile `p` :

```
def inverser_pile(p):
  temp = []
  while len(p) > 0:
    ...
  while len(temp) > 0:
    ...
```

Propositions de complétion :

(a) ☐ temp.empiler(p.depiler())
et p.empiler(temp.depiler())
(b) ☐ p.empiler(temp.pop())
et temp.empiler(p.depiler())
(c) ☐ temp.depiler()
et p.depiler()
(d) ☐ p.depiler()
et temp.depiler()

15. On utilise une pile pour effectuer un parcours en profondeur d'un graphe. Que se passerait-il si on remplace cette pile par une file?

(a) ☐ Le parcours reste en profondeur
(b) ☐ Le parcours devient en largeur
(c) ☐ Le parcours devient aléatoire
(d) ☐ Le parcours est impossible

16. Compléter le code Python pour vérifier si une expression contenant des parenthèses est correctement équilibrée.
Exemple :
- L'expression `"((a+b)c)"` est correctement équilibrée → retourne `True`
- L'expression `"((a+b)c"` n'est pas équilibrée → retourne `False`

```
def parentheses_equilibrees(expr):
  p = Pile()
  for c in expr:
    if c == '(':
      ... # compléter
    elif c == ')':
      # compléter la condition
      if p.estVide() or ...:
        return False
  return p.estVide()
```

Propositions de complétion pour les deux parties :

(a) ☐ p.empiler(c) et p.depiler() != '('
(b) ☐ p.depiler(c) et p.depiler() != ')'
(c) ☐ p.empiler(c) et p.empiler() != '('
(d) ☐ p.depiler(c) et p.empiler() != ')'