DST NSI 1ère

Durée 2 heures

Veuillez répondre sur une copie séparée et rendre le sujet avec votre copie.

Question 1 – architecture de Von Neumann (2 points)

- citer les éléments essentiels dans l'architecture de Von Neumann.
- citer également 3 exemples de périphériques.

Question 2 – architecture de Von Neumann (2,25 points)

Une variable x possède une valeur en mémoire à l'adresse ad104103. Un programme souhaite additionner cette valeur avec le nombre entier 7. Recopiez les phrases suivantes sur votre copie en les complétant au niveau des [?] . Elles décrivent ce qui se passe pour que le résultat du calcul soit affiché, en utilisant chacun des termes suivants une seule fois :

- bus d'adresse
- registre d'instruction RI
- compteur de programme CP
- bus de données
- unité arithmétique et logique UAL
- accumulateur ACC
- x
- ad104103
- code opération

L [?] demande au [?] de récupérer la valeur de [?] qui se trouve à l'adresse [?].
L [?] transmet cette valeur à [?] qui utilise l [?] pour effectuer l'addition avec 7.
La valeur obtenue est stockée dans l [?] et l [?] est incrémenté de 1.

Question 3 – tri par insertion (2 points)

On considère la liste suivante :

3 9 6 1 7 5 4

Trier cette liste en utilisant la méthode du tri par insertion, étape par étape en présentant votre travail sous forme de tableau.

Question 4 – tri par sélection (2 points)

On considère la liste suivante :

3 9 6 1 7 5 4

Trier cette liste en utilisant la méthode du tri par sélection, étape par étape en présentant votre travail sous forme de tableau.

Question 5 – Python (3 points)

Quelle est la valeur de n affichée à la fin du programme ?

```
n = 1
a = 20
b = 3
while a >= b :
    n = n + 2
    a = a - b
    b = b + 1
print(n)
```

Question 6 – Python (4 points)

Ecrire en Python une fonction **`est_ordonne(L),`** qui prend en paramètre une liste **`L`** contenant des nombres entiers et renvoie **`True`** si ses éléments sont ordonnés par ordre croissant et **`False`** sinon.

Question 7 – Python (3 points)

Créer un script Python qui affiche le résultat suivant en utilisant une boucle **`for`**. Attention, il n'est pas demandé d'écrire 9 lignes d'instructions **`print`**.

```
X
XX
XXX
XXXX
XXXXX
XXXXXX
XXXXXXX
XXXXXXXX
XXXXXXXXX
```

Question 8 – Python (3 points)

On veut maintenant obtenir l'affichage suivant en utilisant une boucle **`for`** :

```
X
XX
X X
X  X
X   X
X    X
X     X
X      X
XXXXXXXXX
```

Compléter sur votre feuille à la place des **[?]** le code suivant afin d'obtenir le résultat ci-dessus :

```
for i in range([?]):
    if i == 0:
        print('X')
    elif i == 8:
        print(9 * 'X')
    else:
        print([?])
```

Question 9 – Python (5 points)

Ecrire un script Python qui, étant donné un nombre entre 2 et 12, affiche toutes les combinaisons possibles permettant d'obtenir ce nombre avec deux dés à six faces.

Question 10 – Python (8 points)

A partir du code de la question 8, écrire en Python une fonction appelée **`combinaison()`** qui ne prend aucun argument en paramètre et affiche pour chaque nombre entre 2 et 12 une liste de tuples de toutes les combinaisons possibles permettant d'obtenir ce nombre avec deux dés à six faces. Pour cela, on complètera le code ci-dessous au niveau des **`[?]`**

```
def combinaison():
    for n in range(2, 13):
        [?]

# execution du programme
combinaison()
```

L'exécution de ce code permet d'obtenir l'affichage suivant :

```
2: [(1, 1)]
3: [(1, 2), (2, 1)]
4: [(1, 3), (2, 2), (3, 1)]
5: [(1, 4), (2, 3), (3, 2), (4, 1)]
6: [(1, 5), (2, 4), (3, 3), (4, 2), (5, 1)]
7: [(1, 6), (2, 5), (3, 4), (4, 3), (5, 2), (6, 1)]
8: [(2, 6), (3, 5), (4, 4), (5, 3), (6, 2)]
9: [(3, 6), (4, 5), (5, 4), (6, 3)]
10: [(4, 6), (5, 5), (6, 4)]
11: [(5, 6), (6, 5)]
12: [(6, 6)]
```

Question 11 – Python (5 points)

On considère le dictionnaire et le tableau de mots suivants

```
rangement_alphabet = {
    'a': [],
    'b': [],
    'c': []
}

mots = ["avenir", "choix", "bienvenue", "atelier", "aventure"]
```

On souhaite ranger les mots du tableau **mots** dans le dictionnaire **rangement_alphabet**, en respectant la structure logique : un mot commençant par la lettre ‘a’ vient se rajouter au tableau de valeurs correspondant à la clé **‘a’** du dictionnaire. Idem pour ‘b’ et ‘c’.
On souhaite ainsi obtenir le résultat suivant à l’affichage :
{'a': ['avenir', 'atelier', 'aventure'], 'b': ['bienvenue'], 'c': ['choix']}

Compléter le code suivant à la place des **[?]** pour parvenir au résultat souhaité :

```
def placer_mots(tab):
    global rangement_alphabet

    for elt in tab:

        if [?] == 'a':
            [?]
        elif [?] == 'b':
            [?]
        elif [?] == 'c':
            [?]

placer_mots([?])
print(rangement_alphabet)
```

Question 12 – Docstring (4 points)

Ecrire le docstring de la fonction suivante afin d'expliquer ce qu'elle fait :

```
def mystere(tab):
    nb = 0
    for i in range(len(tab) - 1):
        for j in range(i + 1, len(tab)):
            if tab[i] > tab[j]:
                nb += 1
    return nb
```