MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE **(MESRS)**

DIRECTION GENERALE DE
L'ENSEIGNEMENT SUPERIEUR (**DGES**)

Institut National Polytechnique
Félix Houphouët – Boigny

BUREAU CENTRAL des CONCOURS

REPUBLIQUE DE CÔTE D'IVOIRE
Union - Discipline - Travail

Concours A2GP **session 2025**

Composition : Informatique 2

Durée : 2 Heures

Consignes pour les candidats	Le sujet comporte un exercice portant sur les généralités sur l'inforatique et un problème portant sur l'algorithmique et la programmation en langage Python. Merci de ne rien marquer sur le sujet. Déposer uniquement la grille de réponses remplie

Exercice 1. Culture informatique

Indiquer les réponses correctes pour chacune des questions .

Q01. Quel est l'objectif principal de l'Internet des objets (IoT) ?	
A. Améliorer la vitesse des appareils connectés du quotidien B. Connecter des objets du quotidien à Internet et les piloter à distance	C. Protéger les données personnelles grâce à la domotique D. Augmenter la capacité de stockage des ordinateurs connectés
Q02. Quels sont les deux types principaux de logiciels ?	
A. Logiciels de sécurité et logiciels de jeu B. Logiciels en ligne et logiciels hors ligne	C. Logiciels applicatifs et logiciels système D. Logiciels gratuits et logiciels payants
Q03. Quelle est la taille d'un octet ?	
A. 8 bytes B. 16 bits	C. 8 bits D. 32 bits
Q04. Quel est le terme utilisé pour désigner l'espace de stockage temporaire des données ?	
A. Mémoire ROM B. Mémoire cache	C. Mémoire vive D. Disque dur
Q05. Quel est le rôle du gestionnaire de tâches dans un système d'exploitation ?	
A. Gérer les fichiers B. Afficher les performances de l'ordinateur	C. Permettre à l'utilisateur de quitter des applications D. Toutes les réponses précédentes
Q06. Quelle est la représentation binaire de 15 ?	
A. 1110 B. 1111	C. 1101 D. 1011
Q07. Quel est le protocole utilisé pour envoyer des e-mails ?	
A. HTTP B. FTP	C. SMTP D. IMAP
Q08. Quel est l'outil principal pour visualiser le contenu d'un site web ?	
A. Un éditeur de code B. Un navigateur	C. Un système d'exploitation D. Un client de messagerie

Q09. Quelle est la fonction de la carte mère ?	
A. Exécuter des programmes de démarrage B. Gérer l'alimentation du processeur et des disques durs	C. Connecter tous les composants de l'ordinateur D. Assurer le fonctionnement des composants matériels
Q10. Qu'est-ce qu'un pilote (driver) ?	
A. Un logiciel qui permet à un système d'exploitation de communiquer avec du matériel B. Un type de logiciel malveillant	C. Un logiciel programme de jeu D. Un outil de traitement de texte

Problème : Accès sécurisé pour une MarketPlace de Produits Agricoles.

Dans le cadre de la digitalisation des chaînes de valeurs agricoles, des plateformes numériques sont développées pour mettre en contact les différents acteurs et faciliter les échanges directs. Ces plateformes appelées MarketPlace (Place de Marché) permettent aux producteurs d'annoncer la disponibilité de produits agricoles à destination des acheteurs mais elles permettent aussi aux opérateurs tels que les semenciers ou les entreprises phytosanitaires de proposer des produits et des financements aux agriculteurs.
Afin de « sécuriser » les communications entre les membres, chacun d'entre eux dispose de deux codes personnels permettant de se connecter à la page de la MarketPlace : un code d'accès et un code de vérification. Le code de vérification est envoyé par SMS à chaque demande de connexion.
Le principe des codes est le suivant :

- le premier code est un entier codé en base 16. En fait, l'entier correspondant à la date de naissance de chaque membre (jjmmaaaa) est utilisé. Pour un membre né le 23 septembre 2007, c'est la valeur 23092007 qui est codée en 1605B27.
- le second code est un entier compris entre 100 et 999 dont la somme des chiffres correspondant à la somme des chiffres du premier code. Ainsi par exemple un membre ayant le code 1DA9438 correspondant à l'entier 31102008 (3+1+1+0+2+0+0+8=15 ; 1+5=6) a reçu le code de vérification 303 (3+0+3=6).

On vous demande d'assister les animateurs de la MarketPlace pour générer les codes et garantir l'intégrité des accès aux services proposés. On ne vous demande pas de gérer les envois par SMS. Au mieux on suppose que l'utilisateur dispose des 2 codes et on vous demande de vérifier qu'il peut se connecter avec ces deux valeurs.

```
def main():
        while True:
                print("Menu :")
                print("1: Créer un compte")
                print("2: Accès sécurisé à la plateforme")
                choix = input("Sélectionnez une option (1, 2, 0) : ")
                if choix == '1':
                        date_naissance = int(input("Entrez votre date de naissance : "))
                        code_princ = generer_code_principal(date_naissance)
                        print(f"Votre code principal est : {code_princ}")
                elif choix == '2':
                        code_acces_hex = input("Entrez votre code d'accès principal : ")
                        verif_code =
generer_code_verification(somme_chiffre_rec(convertir_hex_en_entier(code_acces_hex)))
                        print(f"===== Code d'accès temporaire:{verif_code} =====")
                        code_verification = int(input("Entrez votre code de vérification : "))
                        if verifier_connexion(code_acces_hex, code_verification):
                                print("Connexion réussie !")
                        else:
                                print("Échec de la connexion. Les codes ne correspondent pas.")
                elif choix == '0':
                        print("Action en cours, ….")
                        break
                else:
                        print("Option invalide, veuillez réessayer.")
```

Q11. Est-il possible de quitter la fonction main() ?	
A. NON puisque le test du while porte sur True qui est toujours vrai. B. NON. La condition du While aurait du porter sur la variable choix	C. OUI. Si l'utilisateur entre la valeur 0 D. OUI lorsque l'utilisateur entre toute valeur invalide autre que 1 ou 2

1. Générer le code d'accès principal

La génération du code d'accès principal est relativement linéaire.

La conversion en base 16 consiste à effectuer des divisions entières successives par 16 et à représenter le résultat sous la forme de symboles correspondant aux restes successifs obtenus. On s'arrête lorsque le reste est inférieur à 16.

Opion1 : On considère la chaîne de caractères suivante nommée hexasymbole=`'0123456789ABCDEF'.` Vu qu'une chaîne de caractère peut se parcourir comme un vecteur de caractères, il suffit de considérer après chaque division entière le symbole se trouvant en position hexasymbole[reste] et de l'ajouter à la représentation hexadécimale.

Option 2 : On considère le dictionnaire suivant nommé hexadico = {0: '0',1: '1',2: '2',3: '3',4: '4',5: '5',6: '6',7: '7',8: '8',9: '9',10: 'A',11: 'B',12: 'C',13: 'D',14: 'E',15: 'F'} . De façon similaire, il suffit de considérer après chaque division entière le symbole se trouvant en position hexadico[reste] et de l'ajouter à la représentation hexadécimale.

```
def generer_code_principal(k):
    if k == 0:
        return '0'
    else:
        hexasymbole = '0123456789ABCDEF'
        hex_result = ''
        while k > 0:
            reste = k % 16
            hex_result =  hex_result + hexasymbole[reste]
            k //= 16
        return hex_result
```

Q12. Avec le code fourni, generer_code_principal(30042025) retournera :	
A. 9A76AC1 comme attendu B. 1CA67A9 comme attendu	C. 9A76AC1 au lieu de 1CA67A9 D. 1CA67A9 au lieu de 9A76AC1

Q13. Le test `if k == 0:` est	
A. nécessaire pour prendre en compte ce cas B. inutile car on aurait pu mettre `while k >= 0:`	C. nécessaire pour éviter de diviser par 0 D. inutile car on ne divise jamais par 0

Q14. Réécrire la fonction generer_code_principal() pour corriger les éventuels dysfonctionnements et utiliser le dictionnaire hexadico

2. Générer le code de vérification

Considérons qu'il nous faut générer un code de vérification pour un entier N correspondant à la somme des chiffres d'un entier compris entre 100 et 999. il nous faut d'abord générer la liste des entiers dont la somme des chiffres vaut N et tirer au sort lequel sera envoyé à l'utilisateur.

Il nous faut tout d'abord une fonction réalisant la somme demandée, disons somme_chiffre(x:entier):entier. Ensuite en parcourant les valeurs comprises entre 100 et 999, nous allons ajouter à liste_code les valeurs pour lesquels somme_chiffre(x) est égal à N.

```
import random
def somme_chiffre(x: int) -> int:
   while x >= 10:
      somme = 0
      while x > 0:
         somme += x % 10
         x //= 10
      x = somme
   return x
def generer_code_verification(N: int) -> int:
        liste_code = []
        for x in range(100, 1000):
                if somme_chiffre(x) == N:
                        liste_code.append(x)
        if liste_code:
                return random.choice(liste_code)
        else:
                return None
```

Q15. L'instruction `x //= 10` de la fonction somme_chiffre() permet	
A. de s'assurer que x est multiple de 10 B. de remplacer x par le multiple de 10 suivant	C. de remplacer x par la partie entière de x D. de retirer le dernier chiffre de x

Q16. Proposer une version récursive de la fonction somme_chiffre()

Q17. L'instruction `if liste_code:` de la fonction generer_code_verification() :	
A. est incorrecte car liste_code est un tableau qui peu contenir potentiellement plusieurs valeurs B. est incorrecte car il manque un terme au test	C. est correcte car teste si le tableau est vide D. est correcte car teste si le tableau existe

Q18. L'instruction `for x in range(100, 1000):` de la fonction generer_code_verification() :	
A. est incorrecte. Il fallait plutôt dire **`range(100, 999)`** B. est incorrecte. Il fallait plutôt dire **`range(101, 999)`**	C. est incorrecte. Il fallait plutôt dire **`range(100, 1001)`** D. est correcte en l'état

3. Contrôle d'accès à la page

Lorsqu'un membre désire se connecter, il doit fournir son code d'accès (en hexadécimal) et son code de vérification.

Pour éviter d'avoir à utiliser une base de données relationnelle pour stocker les codes de vérification qui changent à chaque connexion, l'astuce consiste à vérifier que la somme des chiffres des deux codes est bien la même. Afin de pouvoir utiliser la même fonction somme_chiffre(x:entier):entier, il faut convertir le code hexadécimal fourni en entier.

On utilisera pour cela le dictionnaire hexareverse= {'0': 0,'1': 1,'2': 2,'3': 3,'4': 4,'5': 5,'6': 6,'7': 7,'8': 8,'9': 9,'A': 10,'B': 11,'C': 12,'D': 13,'E': 14,'F': 15}. On vérifie facilement que pour convertir un hexadécimal en entier, il suffit de :
1. Prendre la valeur du premier symbole de la représentation hexadécimale comme V.
2. Multiplier V par 16 et ajouter la valeur du symbole suivant. Stocker cette somme dans V et répéter l'opération 2 avec le symbole suivant.
3. Lorsque tous les symboles ont été traités, l'entier recherché est dans V

Q19. Proposer une fonction `convertir_hex_en_entier()` permettant de convertir un hexadécimal en entier.

```
def verifier_connexion(code_acces_hex, code_verification):
    code_acces_entier = convertir_hex_en_entier(code_acces_hex)
    somme_acces = somme_chiffre(code_acces_entier)
    somme_verification = somme_chiffre(code_verification)
    return somme_acces == somme_verification
```

Q20. L'instruction `return somme_acces == somme_verification`: de la fonction verifier_connexion() :

A. est incorrecte. Elle retourne une erreur B. est incorrecte. Il fallait plutôt réaliser le test et retourner un entier.	C. est incorrecte. Il manque des parenthèses pour indiquer que le test doit être évalué avant le return. D. est correcte en l'état