Question 1 :

Parmi les affirmations suivantes lesquelles sont vraies :
A) En informatique la mémoire est un dispositif électronique qui sert à stocker des informations.
B) La mémoire du disque dur doit pouvoir fonctionner en mode lecture mais pas en mode écriture pour ne pas être détériorée.
C) La mémoire du disque dur doit pouvoir fonctionner en mode écriture mais pas en mode lecture pour des raisons de sécurité informatique.
D) Dans la mémoire vive d'un ordinateur sont stockées définitivement des données importantes.

Question 2 :

Parmi les affirmations suivantes lesquelles sont vraies :
A) Une machine à calculer peut stocker une infinité de données puisqu'elle peut, par exemple, tracer la courbe représentative d'une fonction.
B) Une machine à calculer, aussi performante soit-elle, ne peut contenir qu'un nombre fini de données.
C) Tout nombre entier peut être représenté par une suite finie de 0 et de 1.
D) Tout nombre réel peut être représenté par une suite finie de 0 et de 1.

Question 3 :

Parmi les affirmations suivantes lesquelles sont vraies :
A) Un nombre entier naturel qui est représenté en binaire par une suite de 0 et de 1 et qui se termine par 1 est pair.
B) Le nombre décimal 0,1 possède une représentation binaire finie.
C) Sur un octet de mémoire, le plus grand entier naturel représentable par une suite de 0 ou de 1 est 255.
D) 110 en binaire représente 10 en base dix.

Question 4 :

Parmi les scripts suivants lesquels échangent les valeurs de `a` et `b` ?

A)
```
c=a
b=c
c=a
```
B)
```
c=a
b=c
a=b
```
C)
```
a=a+b
b=a-b
a=a-b
```
D)
```
a=a+b
b=a-b
a=a+b
```

Question 5 :

On définit, par le script suivant, une fonction `ORDONNE` :

```
def ORDONNE(L):
     n=len(L)
     for i in range(0,n-1):
          if L[i]>L[i+1]:
               c=L[i]
```

```
            L[i]=L[i+1]
            L[i+1]=c
    return L
```

Parmi les assertions suivantes lesquelles sont vraies :
A) `ORDONNE([1,3,2,5,4])` renvoie `[1,2,3,4,5]`.
B) Si `L` est une liste de nombres, `ORDONNE(L)` renvoie une liste de nombres réordonnée dans l'ordre croissant.
C) `ORDONNE([1,3,5,2,4])` renvoie `[1,2,4,3,5]`.
D) `ORDONNE(ORDONNE([3,2,1]))` renvoie `[1,2,3]`.

Question 6 :

`L` est une liste de `n` entiers quelconques. Parmi les assertions suivantes lesquelles sont vraies :
A) Pour ordonner `L`, il suffit d'utiliser `n-1` fois la fonction `ORDONNE`.
B) Pour ordonner `L`, il suffit d'utiliser `n` fois la fonction `ORDONNE`.
C) Une liste `L` ordonnée dans le sens décroissant n'est pas modifiée par la fonction `ORDONNE`.
D) Une liste $\widehat{L}$ ordonnée dans le sens croissant est un invariant pour la fonction `ORDONNE`.
NB : La fonction `ORDONNE` est sauvegardée dans un fichier nommé `ORDONNE.py`

Question 7 :

On définit par le script suivant une fonction `Ordonnum` :

```
def Ordonnum(L):
     from ORDONNE import ORDONNE
     n=len(L)
     for i in range(0,n-1):
          L=ORDONNE(L)
     return L
```

`L` une liste de `n` entiers quelconques. Parmi les assertions suivantes lesquelles sont vraies :
A) `Ordonnum` est une fonction dont l'argument est un réel.
B) `Ordonnum` est une fonction dont la valeur de sortie est un réel.
C) `Ordonnum(L)` renvoie `True` si `L` est ordonnée dans le sens croissant.
D) `Ordonnum(L)` renvoie les éléments de la liste `L` ordonnée dans le sens croissant.

Question 8 :

On définit, par le script suivant, une fonction `Ordonnam` :

```
def Ordonnam(L):
     from ORDONNE import ORDONNE
     n=len(L)
     P=list (L)
     for i in range(0,n-1):
          L=ORDONNE(L)
     test=True
     for i in range(0,n-1):
          if L[i]-P[i]!=0:
               test=False
     return test
```

`L` est une liste de `n` entiers quelconques. Parmi les assertions suivantes lesquelles sont vraies :
A) `Ordonnam` est une fonction dont l'argument est un booléen.
B) `Ordonnam` est une fonction dont la valeur de sortie est un booléen

C) `Ordonnam` est une fonction qui dit si oui ou non une liste ordonnée dans le sens croissant.
D) `Ordonnam(L)` renvoie les éléments de la liste `L` ordonnés dans le sens croissant.

Question 9 :

On définit par le script suivant une fonction `Ordonnim` :

```
def Ordonnim(L):
     from ORDONNE import ORDONNE
     n=len(L)
     P=L
     for i in range(0,n-1):
          L=ORDONNE(L)
     test=True
     for i in range(0,n-1):
          if L[i]-P[i]!=0:
               test=False
     return test
```

`L` est une liste de `n` entiers quelconques. Parmi les assertions suivantes lesquelles sont vraies :
A) `Ordonnim` est une fonction qui a la même action que `Ordonnam`.
B) `Ordonnim` renvoie toujours `True`.
C) Dans `Ordonnim`, la variable locale `P` est constamment égale à `L` lors de l'exécution de la fonction.
D) `Ordonnim(L)` renvoie les éléments de la liste `L` ordonnés dans le sens croissant.

Question 10 :

Parmi les assertions suivantes lesquelles sont vraies :
A) La complexité d'un algorithme est une notion qui permet de quantifier la difficulté à concevoir cet algorithme.
B) La complexité d'une algorithme est une notion qui permet de quantifier le coût de cet algorithme tant en terme de temps d'exécution qu'en terme de place mémoire utilisée pendant l'exécution en fonction du nombre et de la taille des données du problème qu'on veut traiter.
C) Deux algorithmes de complexités différentes aboutissent nécessairement à deux résultats différents.
D) Pour un même résultat, il existe des algorithmes de complexités différentes. Celui qui est le plus efficace est celui dont la complexité est la plus élevée pour un nombre de données fixé.

Question 11 :

On définit, par le script suivant, une fonction `Ordonnom` :
Parmi les assertions suivantes lesquelles sont vraies :

```
def Ordonnom(L);
     for  i,s in enumerate (L):
          j=i
          while 0<j and s<L[j-1]:
               L[j]=L[j-1]
               j=j-1
          L[j]=s
     return L
```

A) `Ordonnom` et `Ordormum` sont deux fonctions qui ont le même résultat de sortie si on les applique à une même liste.
B) `Ordonnom` et `Ordonnum` sont deux algorithmes qui n'ont pas toujours le même résultat si on les applique à une même liste.
C) La complexité dans le cas le pire de `Ordonnom` est en $O(n^3)$ où n désigne la taille de la liste `L`.
D) La complexité dans le cas le pire de `Ordonnom` est en $O(n)$ où n désigne la taille de la liste `L`.

Question 12 :

Richard joue avec un dé. Il lance le dé jusqu'à obtenir un 6. On souhaite conserver tous les résultats des différents lancés dans une variable, y compris le 6 qui clôt l'expérience.

Parmi les assertions suivantes lesquelles sont vraies ?

A) Je vais utiliser une structure de pile car je ne connais pas le nombre de données à mémoriser avant d'avoir fini.

B) Je vais choisir une structure de tableau car je vais pouvoir accéder en une instruction à toutes valeurs stockées et si je choisis, pour ce tableau, une taille suffisamment grande je suis certain que ça suffira.

C) En pseudo code, le script suivant répond à ma demande

```
Créer RESULTATS
     Lancer=0
     Tant que RESULTATS vide ou Lancer<>6 faire
          Lire(Lancer) #La machine attend la saisie du résultat du dé
          Empiler(Lancer, RESULTATS)
     fin Tant que
     Empiler (Lancer ,RESULTATS)
```

D) En pseudo code, le script suivant répond à ma demande

```
Créer RESULTATS
     Lancer=0
     Tant que RESULTATS vide et Lancer<>6 faire
          Lire(Lancer) #La machine attend la saisie du résultat du dé
          Empiler(Lancer, RESULTATS)
     fin Tant que
     Empiler(Lancer, RESULTATS)
```

Question 13 :

Richard parie avec Julie que la somme des lancers qu'il va obtenir avant de faire 6 va être au moins de 100.

Parmi les différents versions de la fonction `gain` suivantes lesquelles pourront-ils utiliser pour savoir qui a gagné ?

A)
```
def gain(resultat):
     S=0
     while resultat!=[]:
          S=S+resultat.pop()
     if S<100:
          return ('Richard gagne')
     else:
          return('Julie gagne')
```

B)
```
def gain(resultat):
     S=0
     while resultat!=[]:
          S=S+resultat.pop()
     if S>=100:
          return ('Richard gagne')
     else:
          return('Julie gagne')
```

C)
```
def gain(resultat):
     S=0
     while resultat !=[]:
          S=S+resultat.append()
          if S>=100:
               return ('Richard gagne')
```

```
            else:
                return ('Julie gagne')
```

D)
```
def gain(resultat):
    S=0
    while resultat!=[]:
        S=S+resultat.append()
    if S<100:
        return ('Richard gagne')
    else:
        return ('Julie gagne')
```

Question 14 :

Antoine propose de parier avec Richard en utilisant la fonction suivante :

```
def gagne(resultat):
    Score1=0
    Score2=0
    while resultat!=[]:
        p=resultat.pop()
        if p==1:
            score1=score1+1
        else:
            score2=score2+1
    if score1>score2:
        return ('Richard gagne')
    else:
        return ('Antoine gagne')
```

En utilisant cette fonction `gagne`, parmi les assertions suivantes lesquelles sont justes :

A) Antoine gagne si le nombre de 1 obtenus par Richard est strictement supérieur au nombre de 2.
B) Antoine gagne si le nombre de 1 obtenus par Richard est strictement inférieur au nombre de 2.
C) Antoine gagne si Richard ne fait pas de 1.
D) Antoine gagne si Richard obtient une suite de 1 au moins de longueur 2.

Question 15 :

Pour les questions 15 à 18, on considère une base de données utilisée dans l'agence de location immobilière CHEZ MOI à Toulouse. Cette base de données contient les deux tables suivantes :

`APPARTEMENTS (APT_ID, APT_PRO, APT_VILLE, APT_TARIF, APT_SURF)`
`PROPRIETAIRES (PRO_ID, PRO_NOM , PRO_PRENOM, PRO_ADRESSE, PRO_TEL)`

La première regroupe les données sur les appartements : leur identifiant, l'identifiant de leur propriétaire, la ville, le montant du loyer et la surface.

La deuxième regroupe les données sur les propriétaires des appartements : leur identifiant, le nom, le prénom, leur adresse et le numéro de téléphone.

Les clés primaires sont soulignées.

À quoi servent ces clés ?

A) À enregistrer toutes les données.
B) À effectuer les mises à jour.
C) À distinguer chaque enregistrement de manière unique.
D) À dupliquer plusieurs lignes d'une table

Question 16 :

Que fait la requête SQL suivante :

```
SELECT * FROM APPARTEMENTS WHERE APP_SURF > 40;
```

A) Elle sélectionne tous les champs de la table APPARTEMENTS.
B) Elle sélectionne les données de la table APPARTEMENTS concernant les appartements dont la surface est strictement supérieure à 40.
C) Elle compte le nombre d'appartements de surface supérieure à 40 dans la table APPARTEMENTS.
D) Elle n'est pas valide.

Question 17 :

Lesquelles des requêtes suivantes permettent d'obtenir l'identifiant des propriétaires des appartements dont le loyer est le plus élevé ?

A)
```
SELECT APT_PRO FROM APPARTEMENTS
WHERE APT_TARIF =(SELECT MAX(APT_TARIF) FROM APPARTEMENTS);
```
B)
```
SELECT APT_PRO FROM APPARTEMENTS
WHERE MAX(APT_TARIF);
```
C)
```
SELECT APT_PRO FROM APPARTEMENTS
WHERE APT_TARIF = (SELECT MAX(APT_TARIF) FROM PROPRIETAIRES);
```
D)
```
SELECT APT_PRO FROM PROPRIETAIRES
WITH MAX(APT_TARIF);
```

Question 18 :

Quel opérateur est le plus adapté pour obtenir le numéro de téléphone des propriétaires des appartements dont le loyer est le plus élevé.

A) Une jointure.
B) Une division cartésienne.
C) Une intersection.
D) Il est impossible d'obtenir ces données car les tables concernant les numéros de téléphone et le montant des loyers sont distinctes.

Question 19 :

On considère le problème de Cauchy : $\begin{cases} \frac{\mathrm{d}y}{\mathrm{d}t} = y^3(t) \\ y(0) = 1 \end{cases}$ pour $t \in [0, 1]$.

On décide de calculer de manière approchée par une méthode d'Euler la solution. Soit n un entier non nul, on pose $y_k = y(\frac{k}{n})$. Parmi les assertions suivantes, lesquelles correspondent bien à un schéma d'Euler explicite pour le problème de Cauchy posé :

A) $y_0 = 1$ et $y_{k+1} = y_k + n.y_k^3$ pour tout $k \in [|0, n-1|]$.
B) $y_0 = 1$ et $y_{k+1} = n.y_k + y_k^3$ pour tout $k \in [|0, n-1|]$.
C) $y_0 = 1$ et $y_{k+1} = y_k + \frac{1}{n}.y_k^3$ pour tout $k \in [|0, n-1|]$.
D) $y_0 = 1$ et $y_{k+1} = y_k + n.y_{k+1}^3$ pour tout $k \in [|0, n-1|]$.

Question 20 :

On reprend le problème posé à la question précédente. On implémente sur une machine une fonction permettant de calculer, pour n donné, les valeurs $(y_k)_{0 \leq k \leq n}$ données par le schéma d'Euler explicite précédent. Parmi les assertions suivantes lesquelles sont vraies.

A) Plus n est grand, plus le temps de calcul sera élevé.
B) Plus n est grand, plus l'erreur commise entre la solution calculée et la solution réelle est forte.
C) Le choix de n n'influe pas sur la qualité de la solution calculée.
D) Choisir un n correctement, c'est faire un compromis entre le temps de calcul et la qualité de la solution obtenue.