Bac Général Maths + NSI (Mathématiques et Numérique et sciences informatiques)

SUJET D'EXAMEN (SESSION OFFICIELLE)

Numérique et sciences informatiques - Session 2021

Sujet de l'examen national tel que publié pour la session indiquée. Source : sujets nationaux publiés par le ministère chargé de l'examen et les services académiques. Document mis à disposition par FormaV pour la révision.

Toutes les annales et leurs corrigés : https://bac-maths-nsi.fr/annales-sujets-et-corriges

BACCALAURÉAT GÉNÉRAL

ÉPREUVE D'ENSEIGNEMENT DE SPÉCIALITÉ

SESSION 2021

NUMÉRIQUE ET SCIENCES INFORMATIQUES

Jour 1

Durée de l'épreuve : **3 heures 30**

L'usage de la calculatrice n'est pas autorisé.

Dès que ce sujet vous est remis, assurez-vous qu'il est complet.
Ce sujet comporte 13 pages numérotées de 1/13 à 13/13.

Le candidat traite au choix 3 exercices parmi les 5 exercices proposés

Chaque exercice est noté sur 4 points.

EXERCICE 1 (4 points)

Principaux thèmes abordés : protocoles de communication, architecture d'un réseau et protocoles de routage.

Partie A :

1. Expliquer le rôle du protocole TCP et du protocole IP dans un réseau informatique.

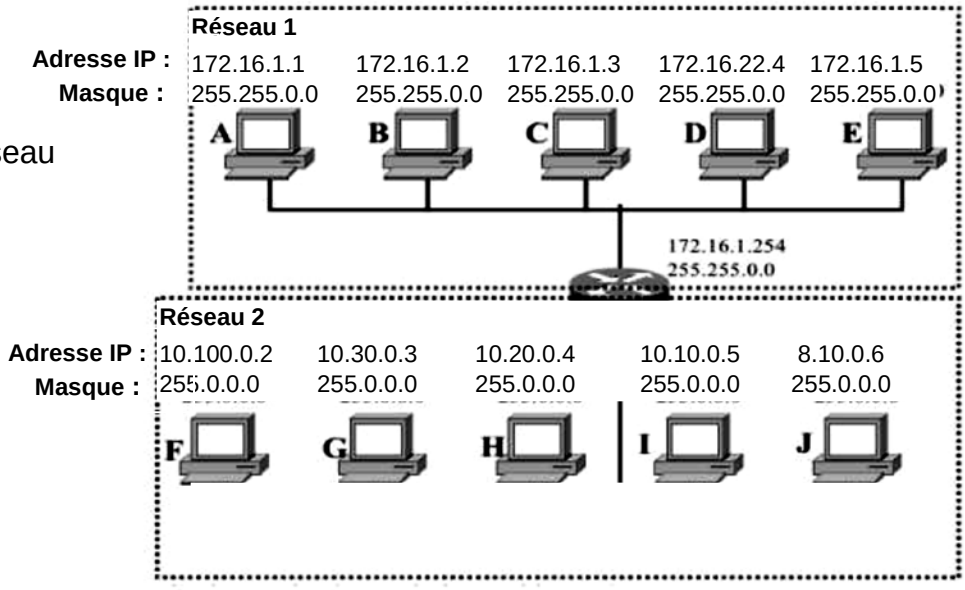
2. On considère un ordinateur dont les paramètres réseau sont les suivants :
Adresse IP : 200.100.10.60
Masque du sous-réseau : 255.255.255.0

a) Donner l'identifiant (adresse) du réseau.

b) Donner la première et la dernière adresse IP qui peuvent être affectées à un hôte. En déduire le nombre de machines hôte identifiables sur un tel réseau.

Partie B :

On considère le réseau donné ci-contre :



1. Donner l'identifiant (adresse) réseau de la machine **A** et celui de la machine **F.**

2. Est-ce que toutes les machines du réseau 1 appartiennent au même réseau IP en considérant le masque proposé? Même question pour le réseau 2. Justifier.

3. Parmi les réponses ci-dessous, donner (en le recopiant) le nombre d'hôtes pouvant être adressés sur le réseau 1.

a) 255^2 b) $255^2 - 1$ c) $256^2 - 1$ d) $256^2 - 2$

4. Sans changer les adresses IP des différentes machines, proposer une architecture réseau permettant d'interconnecter les réseaux en précisant la nature des composants.

EXERCICE 2 (4 points)

Principaux thèmes abordés : algorithmique (recherche dichotomique) et langages et programmation (récursivité)
On veillera à mettre sur la copie toutes les réponses.

Partie A : La recherche dichotomique

1. La recherche d'un élément dans un tableau avec une méthode dichotomique ne peut se faire que si le tableau est trié.
   a) Vrai
   b) Faux
2. Le coût d'un algorithme de recherche dichotomique est :
   a) Constant : Complexité O(1)
   b) Linéaire : Complexité O(n)
   c) Logarithmique : Complexité O(log(n))
3. Justifier pourquoi l'entier **`fin – deb`** est un variant de boucle qui montre la terminaison du programme de recherche dichotomique de l'**annexe 1 de l'exercice 2.**

Partie B : La recherche dichotomique itérative

Le programme de recherche dichotomique de l'**annexe 1 de l'exercice 2** est utilisé pour effectuer des recherches dans une liste.

Dans l'ensemble de cette partie, on considère la liste : **`Lnoms = ["alice", "bob", "etienne", "hector", "lea", "nathan", "paul"].`**

1. Expliquer pourquoi en ligne 2, on a «**`fin = len(liste)-1`**» plutôt que «**`fin = 6`**».
2. En Python, l'opérateur `//` donne le quotient de la division euclidienne de deux nombres entiers.

   Proposer un algorithme pour obtenir ce quotient.
3. Donner la trace complète de l'exécution `rechercheDicho("lea", Lnoms)` en complétant le tableau ci-dessous sur votre copie :

Variables			Condition	Valeur renvoyée
deb	Fin	M	deb <= fin	

4. Sur votre copie, modifier le code du corps de la fonction **`rechercheDicho()`** pour qu'elle renvoie aussi la position (indice) de l'élément cherché ou **`-1`** si l'élément n'est pas trouvé.

   On pourra indiquer sur la copie le numéro des lignes modifiées, à supprimer ou à insérer s'il y a lieu.

Partie C : La recherche dichotomique récursive

1. Donner la définition d'une fonction récursive en programmation.
2. Écrire en langage naturel ou en python, l'algorithme de recherche dichotomique d'un élément dans une liste, triée de façon croissante, en utilisant une méthode récursive. Il renverra **`True`** si l'objet a été trouvé, **`False`** sinon.

EXERCICE 3 (4 points)

Principaux thèmes abordés : bases de données (modèle relationnel, base de données relationnelle et langage SQL).
Répondre aux différentes questions sur votre copie.

Partie A : Modèle et schéma relationnel

1. Associer à chaque mot ci-dessous (issu du vocabulaire du modèle relationnel) un exemple pris dans le schéma relationnel d'AirOne (**Annexe 1 de l'exercice 3**).

a) Table

b) Attribut

2. Parmi les quatre propositions de description données ci-contre, préciser sur votre copie celle associée au concept :

a) Clé primaire

b) Clé étrangère

	Description
1	Attribut(s) permettant d'identifier de façon unique chaque occurrence de la table
2	Attribut dont le domaine est obligatoirement numérique
3	Attribut d'une table qui tire ses valeurs de l'attribut d'une autre table
4	Zone mémoire identifiée par un nom et contenant une valeur unique

Partie B : Base de données

Les concepts définissant le modèle relationnel permettent d'exprimer trois contraintes d'intégrité :

- la contrainte d'intégrité de domaine,
- la contrainte d'intégrité de clé (ou de relation),
- la contrainte d'intégrité référentielle.

Les enregistrements stockés dans la base de données sont présentés dans l'**annexe 2 de l'exercice 3**.

1. L'occurrence suivante est saisie dans la table Vol de la base de données AirOne

numVol	dateVol	hrDep	hrArr	codeIATADep	codeIATAArr	numPilote	numAvion
2549	13/01/2021	12:00	13:45	ORY	BCN	121	F-X25D8F

Décrire l'anomalie créée et indiquer la contrainte d'intégrité correspondante.

2. L'occurrence suivante est saisie dans la table Avion de la base de données AirOne

numA	dateMiseService	type
F-KI452	12/12/2020	A380

Décrire l'anomalie créée et indiquer la contrainte d'intégrité correspondante.

3. L'occurrence suivante est saisie dans la table Type de la base de données AirOne :

nomT	nbPlaces	constructeur
A310	**environ 200**	**Airbus**

Décrire l'anomalie créée et indiquer la contrainte d'intégrité correspondante.

Partie C : SQL

1. Indiquer ce que fait la requête suivante :

```
DELETE
FROM Vol
WHERE dateVol < "11/01/2021" ;
```

2. Corriger la requête ci-dessous afin qu'elle permette d'ajouter un nouveau type d'avion dans la base de données.

```
INSERT VALUES ("A310",250, "Airbus") ;
```

3. Écrire la requête qui donne les types d'avion des vols du 10/01/2021.

EXERCICE 4 (4 points)

Principaux thèmes abordés : Structure de données (programmation objet) et langages et programmation (spécification).

La société LOCAVACANCES doit gérer la réservation de l'ensemble des chambres de ses gîtes. Chaque chambre d'un même complexe sera différenciée par son nom.
Pour cela, d'un point de vue informatique, on a créé deux classes : **`Chambre`** et **`Gite`** dont le code est donné dans l'**annexe 1 de l'exercice 4**.

Partie A - Étude de la classe `Chambre` :

1. Lister les attributs en donnant leur type. Préciser s'ils sont modifiables dans la classe, en explicitant la méthode associée.
2. Écrire un **`assert`** dans la méthode **`reserver`** pour vérifier si le nombre **`date`** passé en paramètre est bien compris entre 1 et 365 (on ne gère pas les années bissextiles).
3. Écrire la méthode **`AnnulerReserver(self, date : int)`** qui annule la réservation pour le jour **`date`**.

Partie B - Étude de la classe `Gite` :

Le gîte « BonneNuit » a 5 chambres dénommées :
`'Ch1', 'Ch2', 'Ch3', 'Ch4', 'Ch5'`
On définit l'objet **`GiteBN`** par l'instruction : **`GiteBN = Gite("BonneNuit").`**

1. Méthode **`ajouter_chambres()`**
   Écrire l'instruction Python pour ajouter **`'Ch1'`** à l'objet **`GiteBN`**.

Dans les questions suivantes 2, 3 et 4, on considère que l'objet **`GiteBN`** contient toutes les chambres du gite « BonneNuit ».

2. La méthode **`ajouter_chambres`** permet d'enregistrer une nouvelle chambre, mais elle ne teste pas si le nom de cette chambre existe déjà.
   Modifier la méthode pour éviter cet éventuel doublon.

3. Étude des méthodes : **`get_chambres() et get_nchambres()`**

a) Parmi les 4 propositions ci-dessous, quel est le type renvoyé par l'instruction Python : **`GiteBN.get_chambres()`**

•String	• Objet Chambre	•Tableau de String	•Tableau d'objet Chambre

b) Qu'affiche la suite d'instructions suivante ?

```
Ch = GiteBN.get_chambres()[1]
print(Ch.get_nom())
```

c) Quelle différence existe-t-il entre les deux méthodes **get_nchambres() et get_chambres()** ?

4. Les chambres **`'Ch1', 'Ch3', 'Ch5'`** sont réservées pour tout le mois de Janvier 2021.

La méthode **`mystère`** étant précisée dans l'**annexe 1 de l'exercice 4,** répondre aux questions suivantes :

a) Que va renvoyer l'instruction **GiteBN.mystere(3)** ?

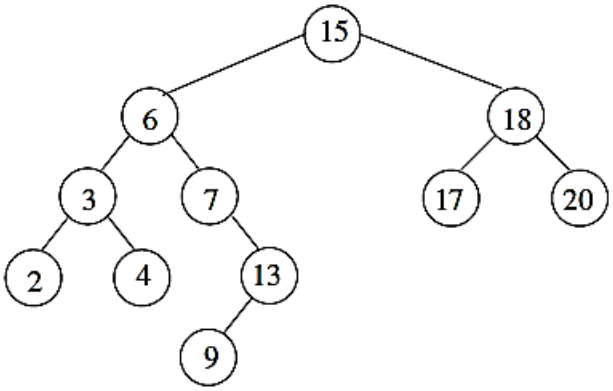
b) Dans la méthode **`mystère`**, quel est type des variables en paramètre et en sortie ?

Quelles sont les méthodes ou attributs dont elle a besoin ?

EXERCICE 5 (4 points)

Principaux thèmes abordés : structures de données (arbre, arbre binaire, pile).

Les valeurs relatives à des temps d'attente en secondes de systèmes électroniques sont stockées dans l'arbre binaire de recherche ci – dessous :



1. a) Rappeler brièvement ce qu'est un arbre binaire de recherche.
b) Quelle est la première valeur qui a été positionnée dans cet arbre ?
c) Préciser la hauteur de cet arbre. (La racine est considérée au niveau 0)

2. Recopier l'arbre donné ci-dessus sur votre copie en y ajoutant successivement les données 16 et 12.

3. Déterminer sur l'arbre ci-dessus, la liste des valeurs obtenue avec un parcours d'arbre en profondeur infixe. Que permet d'obtenir ce parcours ?

4. On considère l'interface suivante relative à la structure de données « arbre binaire » :

- `EstVide : ArbreBinaire[E] → Booléen`
- `Racine : ArbreBinaire[E] → E`
- `Sag : ArbreBinaire[E] → ArbreBinaire[E] (sous arbre gauche)`
- `Sad : ArbreBinaire[E] → ArbreBinaire[E] (sous arbre droite)`

Recopier sur votre copie et compléter l'algorithme de recherche suivant, qui retourne Vrai si la valeur x est dans l'arbre binaire de recherche A, Faux sinon.

```
Recherche(A,x) :
    Si EstVide(A) alors Faux
    Si Racine(A)=x alors ……..
    Si x<Racine(A) alors ……..
    Sinon ……..
```

Exercice 2 - Annexe 1

On considère la fonction de recherche dichotomique suivante :

```
def rechercheDicho (elem, liste):
"""
Cette fonction indique si un élément se trouve dans un
tableau.
Elle utilise la méthode de recherche dichotomique.
Elle prend en arguments :
- elem : élément à rechercher de type string
- liste : liste d'éléments de type string triée
par ordre croissant
Elle renvoie un booléen correspondant à la présence ou
non de l'élément
"""
1       deb = 0
2       fin = len(liste)-1
3       m = (deb+fin)//2
4       while deb <= fin :
5           if liste[m] == elem :
6               return True
7           elif liste[m] > elem :
8               fin = m-1
9           else :
10              deb = m+1
11          m = (deb+fin)//2
12      return False
```

Exercice 3 - Annexe 1

Schéma relationnel de la base de données AirOne

Soit le schéma relationnel suivant permettant la gestion de la compagnie aérienne **AirOne**

Aeroport (codeIATA, nomA, ville, pays)
clé primaire : codeIATA
codeIATA, nomA, ville, pays : String

Type (nomT, nbplaces, constructeur)
clé primaire : nomT
nomT, constructeur : String
nbplaces : entier

Avion(numA, dateMiseService, type)
clé primaire : numA
clé étrangère : type en référence à nomT de la relation Type
numA, type : String
dateMiseService : date

Pilote (numP, nomP, prenom, adresse, dateEmb)
clé primaire : numP
numP, nomP, prenom, adresse : String
dateEmb : date

Vol (numVol, dateVol, hrDep, hrArr, codeIATADep, codeIATAArr, numPilote, numAvion)
clé primaire : numVol
clé étrangère :
- codeIATADep en référence à codeIATA de la table Aeroport
- codeIATAArr en référence à codeIATA de la table Aeroport
- numPilote en référence à numP de la table Pilote
- numAvion en référence à numA de la table Avion

numVol : entier

Exercice 3 - Annexe 2

Extrait de la Base de données AirOne

Table Aeroport

codeIATA	nomA	ville	pays
CDG	Charles de Gaulle	Paris	France
DUB	Dublin International	Dublin	Irlande
DWC	Al Maktoum	Dubai	UAE
JFK	John-F Kennedy	New York	USA
LHR	Heathrow	Londres	Royaume-Uni
ORY	Orly	Paris	France
TLS	Blagnac	Toulouse	France

Table Type

nomT	nbPlaces	constructeur
747	416	Boeing
A320	180	Airbus
A330	375	Airbus
A380	516	Airbus
DHC-8	90	Bombardier

Table Avion

numA	dateMiseService	type
F-KI452	25/01/1990	DHC-8
R-YY45F	10/10/2002	A320
F-JJ254	14/01/2005	A320
US-KKR2	08/05/2005	A330
F-X25D8F	10/01/2006	A380
F-G458F	08/02/2006	747

Table Pilote

numP	nomP	prenom	adresse	dateEmb
120	PALAPATTE	NATACHA	PARIS	05/05/2000
121	LEFRANCOIS	JEAN MICHEL	LONDRES	12/08/2000
122	SMITH	JOHN	LYON	10/10/2000
123	DUPOND	JEAN	PARIS	07/07/2004
124	DUPOND	PATRICK	PARIS	15/01/2005

Table Vol

numVol	dateVol	hrDep	hrArr	codeIATADep	codeIATAArr	numPilote	numAvion
1044	09/01/2020	12:20	13:40	ORY	TLS	120	R-YY45F
1233	10/01/2021	8:00	16:00	LHR	JFK	121	F-G458F
1248	10/01/2021	12:00	23:10	CDG	DWC	123	F-X25D8F
1462	10/01/2021	23:00	0:20	TLS	DUB	120	R-YY45F
1520	11/01/2021	15:00	15:50	DUB	LHR	120	R-YY45F
1562	11/01/2021	20:00	7:25	DWC	CDG	123	F-X25D8F
1589	12/01/2021	8:10	16:20	JFK	LHR	121	F-G458F
2544	12/01/2021	10:10	11:30	LHR	ORY	124	R-YY45F

Exercice 4 - Annexe 1

```
class Chambre:
   def __init__(self, nom: str):
      self._nom  = nom
      self._occupation = [False for i in range(365)]

   def get_nom(self):
      return self._nom

   def get_occupation(self):
      return self._occupation

   def reserver(self, date: int):
      self._occupation[date - 1] = True

class Gite:
   def __init__(self, nom: str):
      self._nom = nom
      self._chambres = []

   def __str__(self):
      n = len(self._chambres)
      if n == 0:
         return "L'hôtel " + self._nom + " n'a aucune chambre."
       else:
         return "L'hôtel " + self._nom + " a " + str(n) + " chambre(s)"

   def get_chambres(self):
      return self._chambres

   def get_nchambres(self):
      return [ch.get_nom() for ch in self._chambres]

   def ajouter_chambres(self, nom_ch : str):
      self._chambres.append(Chambre(nom_ch))

   def mystere(self, date):
      l_ch = []
      for ch in self._chambres  :
      if ch.get_occupation()[date - 1] == False :
         l_ch.append(ch.get_nom())
      return(l_ch)
```