CONCOURS COMMUNS POLYTECHNIQUES – FILIERE TSI

CAHIER DES CHARGES DE L'EPREUVE ECRITE D'INFORMATIQUE

Descriptif de l'épreuve

L'épreuve est d'une durée de 3 heures.

L'épreuve d'informatique doit respecter le programme d'informatique parue au BO spécial N°3 du 30 mai 2013.

Le sujet doit être contextualisé et demande la résolution d'une problématique qui s'appuie sur les thématiques scientifiques et techniques enseignées dans la filière. La problématique s'appuie sur un support concret pouvant prendre différentes formes :

- un support industriel ;
- un support technologique ;
- etc.

À titre d'exemple, des parties du sujet peuvent notamment s'appuyer sur :

- l'analyse de son ou d'image ;
- la résolution d'équations générales issues d'une phase de modélisation ;
- l'analyse et un traitement d'un flux de données ;
- etc.

Langages supports

Le candidat peut utiliser les langages Python ou Scilab. Il doit préciser au début du sujet le langage utilisé. Il peut néanmoins, en le précisant au préalable, changer de langage au cours de l'épreuve. Dans le cas du langage Python, le respect de l'indentation est fondamental (celle-ci sera matérialisée par une barre verticale). Les fonctions spécifiques nécessaires à la résolution d'un problème sont rappelées dans le sujet.

Une attention particulière est apportée par les concepteurs de sujets pour s'assurer que la difficulté de résolution est équivalente dans les deux langages. Par conséquent, des éléments spécifiques à un des langages utilisés ne doivent pas être un élément bloquant dans la progression de l'élève.

Les fautes lexicales qu'un outil de développement intégré détecte immédiatement ne sont pas prises en compte. (Exemple : primt à la place de print).

Si l'usage des calculatrices est laissé au choix des concepteurs de sujets, il convient néanmoins de prendre conscience que certaines machines peuvent embarquer un interpréteur python.

Programme

Situations d'évaluation	Compétences/capacités associées	Propositions d'activités (liste non exhaustive)
Analyser un problème	■ Analyser les influences de la représentation des nombres et du codage des chaînes de caractère. ■ Comprendre un algorithme et expliquer ce qu'il fait. ■ Rechercher une information au sein d'une documentation. ■ Distinguer les rôles respectifs des machines client, serveur et serveur de données.	■ Convertir des nombres. ■ Précision et débordement. ■ Taille d'un fichier. ■ Analyser une documentation.
Spécifier	■ Choisir un type de données en fonction d'un problème à résoudre. ■ Concevoir l'en-tête (signature) et la spécification d'une fonction, puis la fonction elle-même.	■ Choisir un type de donnée. ■ Mode de passage des paramètres (in, out, inout). ■ Réaliser l'en tête d'une fonction.
Traduire	■ Traduire un algorithme dans un langage de programmation. ■ Traduire dans le langage de l'algèbre relationnelle des requêtes écrites en langage courant.	■ Donner un algorithme en pseudo code et le convertir dans un langage python ou scilab. ■ Donner une requête en langage courant et la convertir en requête de l'algèbre relationnelle ou en SQL (uniquement de type SELECT). ■ Utiliser une API fournie dans le sujet.
Modéliser et concevoir	■ Concevoir ou modifier un algorithme existant pour répondre à un problème précisément posé. ■ Réaliser un programme complet structuré permettant de résoudre un problème scientifique donné. ■ Concevoir une base constituée de plusieurs tables et utiliser les jointures symétriques pour effectuer des requêtes croisées.	■ Compléter un algorithme. ■ Modifier un algorithme. ■ Concevoir un algorithme. ■ Concevoir une fonction. ■ Proposer une structure simple de base de données. ■ Concevoir une requête.
Analyser une solution ou des résultats	■ Critiquer la qualité et la précision des résultats de calculs numériques sur ordinateur. ■ Justifier qu'une itération (ou boucle) produit l'effet attendu au moyen d'un invariant. ■ S'interroger sur l'efficacité algorithmique temporelle d'un algorithme et distinguer des algorithmes par leur complexité. ■ Étudier l'effet d'une variation des paramètres sur le temps de calcul, sur la précision des résultats, sur la forme des solutions pour des programmes d'ingénierie numérique choisis, tout en contextualisant l'observation du temps de calcul par rapport à la complexité algorithmique de ces programmes.	■ Comparer plusieurs algorithmes traitant un même problème. ■ Justifier la correction d'un algorithme à l'aide d'un invariant donné ou proposer un invariant de boucle (dans un cas simple). ■ Donner la complexité d'un algorithme (de façon intuitive). ■ Critiquer les résultats produits par un programme.
Communiquer	■ Documenter une fonction, un programme plus complexe. ■ Utiliser des règles claires lors de l'écriture d'un algorithme (indentation, barre verticale...).	■ Documenter une fonction.