ROYAUME DU MAROC

Ministère de l'éducation nationale
de l'enseignement supérieur,
de la formation des cadres
et de la recherche scientifique

Présidence
du Concours National Commun 2011
Institut Nationale des Postes et
Télécomunications (INPT)

CONCOURS NATIONAL COMMUN

d'Admission aux

Grandes Écoles d'Ingénieurs Marocaines et Assimilées

Session 2011

ÉPREUVE D'INFORMATIQUE

Durée **2 heures**

FILIÈRES : **MP/ PSI/ TSI**

Cette épreuve comporte 5 pages au format A4, en plus de cette page de garde.
L'usage de la calculatrice est *interdit*

*Les candidats sont informés que la précision des raisonnements **algorithmiques** ainsi que le soin apporté à la rédaction et à la présentation des copies seront des éléments pris en compte dans la notation. Il convient en particulier de rappeler avec précision les* références *des questions abordées.*
Si, au cours de l'épreuve, un candidat repère ce qui peut lui sembler être une erreur d'énoncé, il le signale sur sa copie et poursuit sa composition en expliquant les raisons des initiatives qu'il est amené à prendre.

Remarques générales :
- L'épreuve se compose de deux problèmes indépendants.
- Toutes les instructions et les fonctions demandées seront écrites en **langage C**.
- Les questions non traitées peuvent être admises pour aborder les questions ultérieures.

PROBLÈME I : OPÉRATIONS SUR DES ENSEMBLES MATHÉMATIQUES

✧✧✧

Préambule
Un **ensemble mathématique** est un groupement d'objets distincts, appelés **éléments** de cet ensemble. La **théorie des ensembles** est l'étude des propriétés et des opérations sur des ensembles (appartenance, inclusion, réunion, ...). Elle représente une branche essentielle des mathématiques.
Ce problème s'intéresse aux algorithmes réalisant quelques traitements sur des **ensembles mathématiques finis de nombres entiers**.

Notation d'un ensemble fini d'éléments
Si **E** est un ensemble fini de **N** éléments ($0 < N$), $e_0, e_1, \ldots, e_i, e_{i+1}, \ldots, e_{N-1}$, alors **E** sera noté ainsi $\mathbf{E} = \{e_0, e_1, \ldots, e_i, e_{i+1}, \ldots, e_{N-1}\}$.

Partie A : Représentation des ensembles finis par des tableaux

Dans cette partie, il s'agit de représenter par **des tableaux**, des ensembles finis dont les éléments sont des **nombres entiers strictement positifs.**

Appellations

On appellera **"Ensemble Tableau de taille N"**, tout tableau de **N**, (**0** < **N**) entiers strictement positifs et tous différents. Ce tableau sera noté $T = \{T[0], T[1], \ldots, T[N-1]\}$

On appellera un **"Ensemble Tableau de taille N trié",** tout **"Ensemble Tableau de taille N"**, **T**, dont les éléments sont triés par ordre croissant : (Pour tout **i** tel que **0** <= **i** < **N** − **1**, on a **T**[**i**] < **T**[**i** + **1**])

Remarque

Dans toutes les questions de la partie A, on suppose que **N**, **N1** et **N2** sont des constantes **entières strictement positives** déjà définies.

Notation :

On notera les **N** éléments du tableau **T** ainsi **T**=$\{T[0], T[1], \ldots, T[N-1]\}$

Question 1 : Vérification d'un Ensemble Tableau

Soi **T** un tableau déclaré et initialisé avec **N** entiers strictement positifs quelconques.

☞ Écrire les instructions qui vérifient si **T** est un **"Ensemble Tableau de taille N"**.

Pour ce faire :

-Déclarer une variable entière de nom **"valide"**.

- Affecter la valeur **1** à la variable **"valide"** si **T** est un **"Ensemble Tableau de taille N"**, ou **0** sinon

Exemples :

-Si N=4 et **T** = {3, 5, 2, 9} alors **valide=1** (**T** est un "**Ensemble Tableau de taille 4"**)

-Si N=5 et **T** = {4, 8, 12, 4, 3} alors **valide=0** (**T** n'est pas un "**Ensemble Tableau de taille 5"**)

Question 2 : Appartenance à un ensemble Tableau

Soi T un "**Ensemble Tableau de taille N"** déclaré et initialisé et soit **x** une variable entière déclarée et initialisée avec un entier quelconque.

☞ Écrire les instructions qui affichent sur l'écran l'une des deux affirmations suivantes (**a**) ou (**b**) (dans les 2 cas, la variable **x** sera remplacée par sa valeur à l'affichage)

(a) "**x appartient à T**" si **x** est un élément du tableau **T**.

(b) "**x n'appartient pas à T**" si **x** n'est pas un élément de **T**.

Exemples :

-Si N = 4, **T** = {3, 5, 2, 9} et **x** = 2 alors on affichera : "2 appartient à **T**"

-Si N = 3, **T** = {12, 6, 9} et **x** = 3 alors on affichera : "3 n'appartient pas à **T**"

Question 3 : Tri d'un Ensemble Tableau

Soit **T** un **"Ensemble Tableau de taille N"** supposé déclaré et initialisé.

☞ Écrire les instructions qui permettent de trier les éléments du tableau **T** par ordre croissant.

Exemple :

-Si N = 4 et **T** = {3, 1, 12, 8} après les instructions de tri, on aura **T** = {1, 3, 8, 12}

Remarque concernant les questions suivantes de la partie A (question 4 et question 5)

On suppose avoir déjà déclaré et initialisé 2 variables globales **T1** et **T2** avec **T1** est un **"Ensemble Tableau de taille N1 trié"** et **T2** un **"Ensemble Tableau de taille N2 trié "**

Question 4 : Inclusion d'un ensemble dans un autre

On dit que **T1** est inclus dans **T2** si tout élément de **T1** est aussi élément de **T2**.
☞ Écrire la fonction d'entête : **int T1inclusdansT2()** qui retourne **1** si **T1** est inclus dans **T2** ou retourne **0** sinon.

Exemple :

- Si **T1** = {4,9,17} et **T2** = {2,4,5,9,17,19} alors l'appel **T1inclusdansT2()** retourne **1**
- Si **T1** = {1,8,10} et **T2** = {1,4,10,6} alors l'appel **T1inclusdansT2()** retourne **0**

Question 5 : Union de deux ensembles tableaux triés

On dit qu'un tableau **T** de taille **N** est l'union de **T1** et **T2** si **T** est un**"Ensemble Tableau de taille N trié"** composé de tous les éléments de **T1** en plus de tous les éléments de **T2**
Soit **T** un tableau de taille **N** déjà déclaré (**N** est la taille du tableau union de **T1** et **T2**).
☞ Écrire les instructions nécessaires pour que **T** soit l'union de **T1** et **T2**

Exemple :

Si **T1** = {2,14,28,75} et **T2** = {1,6,14,28} alors **T** = {1,2,6,14,28,75}

Partie B : Utilisation des listes chaînées

Dans cette partie, on se propose de représenter des ensembles finis **d'entiers strictement positifs triés par ordre croissant** par des listes chaînées définies en **langage C** comme suit

```
typedef struct ens
        { int nombre; //un nombre entier strictement positif élément de l'ensemble
          struct ens *suiv; // l'adresse de l'élément suivant
        } ensembleListe;
```

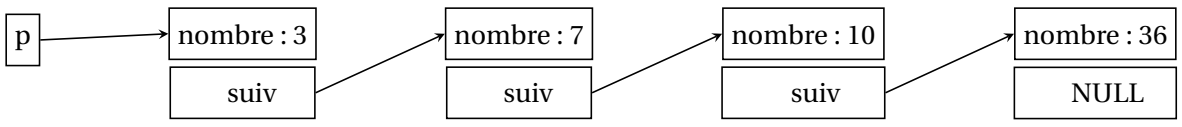
Appellation

On appellera **"Ensemble Liste d'adresse p"** une liste chaînée d'éléments de type **ensembleListe** (définie plus haut) et possédant les propriétés suivantes :

- Le premier élément a l'adresse **p**.
- Le dernier élément a dans son champ **suiv** la valeur **NULL**
- Pour tout élément d'adresse **el** de la liste chaînée, tel que **(el->suiv!=NULL)**, on a **(0< el->nombre < ((el->suiv) ->nombre)** (liste triée par ordre croissant de nombres)

Exemple

L'ensemble {3,7,10,36} sera représenté par **l'Ensemble Liste d'adresse p** comme suit :



Question 6 : Insertion d'un élément dans la liste chaînée triée

Soit la déclaration globale suivante : **ensembleListe *p ;**

On suppose avoir définie et inséré des éléments dans l' **"Ensemble Liste d'adresse p" (p** est déclaré plus haut).

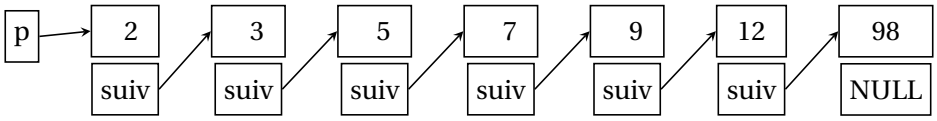
☞ Écrire une fonction d'entête : **void inserer(int val)** qui permet d'insérer **à sa place** l'élément de type **ensembleListe** dans l' **"Ensemble Liste d'adresse p"** pour que la liste reste toujours triée par ordre croissant. Cet élément a dans son champ **nombre**, la valeur **val** (paramètre de la fonction), en plus, on suppose que : **(val>p->nombre)**(voir rappel, remarque et exemple)

Rappel : L'appel de la fonction de la bibliothèque du **langage C malloc(n)** (n étant un entier positif), permet d'allouer n octets dans la mémoire dynamique et retourne l'adresse mémoire du block alloué. La fonction **malloc** est définie dans le fichier de la bibliothèque **stdlib.h**

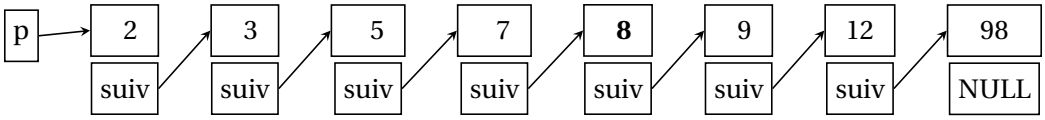
Remarque

- Si le paramètre **val** est la valeur du champ **nombre** d'un élément qui existe déjà dans la liste, aucun élément ne sera inséré.

Exemple : Soit l' **"Ensemble Liste d'adresse p"** suivant :



Après l'appel de la fonction **inserer(8), l' "Ensemble Liste d'adresse p"** devient :



✧✧✧

PROBLÈME II : DISTANCE DE HAMMING

✧✧✧

La **distance de Hamming**, définie par **Richard Hamming** permet de quantifier la différence entre deux séquences de symboles. Elle est utilisée en informatique et en télécommunications pour compter le nombre de bits altérés dans la transmission d'un message d'une longueur donnée.

Dans ce problème, on se propose d'implémenter des fonctions pour calculer la distance de **Hamming**

Question 1 : Distance de Hamming entre deux chaînes de caractères

La distance de **Hamming** entre deux chaînes de caractères de mêmes longueurs est égale au nombre de caractères, à la même position, qui sont différents.

Exemples :

- La distance de **Hamming** entre "sure" et "cure" est **1**,
- la distance de **Hamming** entre "aabbcc" et "xaybzc" est **3**.

☞ Écrire une fonction d'entête : **int distanceH(char S1[ ], char S2[ ], int M)** qui calcule et retourne la distance de **Hamming** entre **S1** et **S2** (Les paramètres **S1** et **S2** sont deux chaînes de caractères de même longueur **M** et on suppose que le paramètre **M** est strictement positif)

Question 2 : Distance de Hamming d'un langage

On appellera **langage**, un tableau de chaînes de caractères toutes de mêmes longueurs. La distance de **Hamming** d'un langage est égale au minimum des distances de **Hamming** entre deux chaînes de caractères de ce langage différentes deux à deux.

Exemple :

- Si **langage={"aabb", "xayy", "tghy", "xgyy"}** , sa distance de Hamming est de **1**

☞ Écrire une fonction d'entête : **int distanceH_langage(char[NB][L] langage)**, qui retourne la distance de Hamming de son paramètre **langage** (Le paramètre **langage** est un tableau de **NB** chaînes de caractères toutes de même longueur **L**, **NB** et **L** sont 2 constantes entières strictement positives déjà définies)

Question 3 : Distance de Hamming entre 2 nombres entiers positifs

La distance de **Hamming** entre 2 nombres entiers positifs est le nombre de bits distincts dans leurs représentations binaires (voir exemple)

Exemple : la distance de **Hamming** entre les nombres 7 et 4 est 2 (7 est représenté en binaire sur un octet (8 bits) par **00000111** et 4 est représenté en binaire par **00000100**)

Question 3-a :

☞ Écrire une fonction d'entête : **void binaire(char *bin ,int N)** qui met dans la chaîne d'adresse **bin**, la représentation binaire de **N** (On suppose que **0< =N <256**)

Question 3-b :

☞ Écrire une fonction d'entête : **int distanceNombre(int A, int B)** qui calcule et retourne la distance de **Hamming** entre les nombre A et B (On suppose que **0** <= **A** < **256** et **0** <= **B** < **256**)

✧✧✧

FIN DE L'ÉPREUVE