Concours NATIONAL COMMUN

D'Admission aux

Grandes Écoles d'Ingénieurs Marocaines et Assimilées

Epreuve zéro (était prévu pour CNC2010)

ÉPREUVE D'INFORMATIQUE

Durée **2 heures**

FILIÈRES : **MP/ PSI/ TSI**

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 appeler 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.

PROBLEME I: Compression et Algorithme RLE

Préliminaires :

- **La compression** de données est le traitement informatique utilisant un algorithme particulier, qui permet de transformer une suite de bits A en une suite de bits B plus courte. Les suites A et B contiennent les mêmes informations, mais codées d'une manière différente

- La décompression est l'opération inverse de la compression

- L'algorithme RLE (Run-Length Encoding), appelé en français **le codage par plages**, est un algorithme de compression informatique qui consiste à repérer et à éliminer la redondance des données. Toute suite de bits ou de caractères identiques est remplacée par un couple (caractère répété suivi de **son nombre d'occurrences** (son nombre de répétitions)).

Exemple: **CCCCBBBRCC** donne: **C4B3R1C2**

Dans ce qui suit, on se propose d'étudier **l'algorithme RLE** pour compresser des chaînes de caractères :

- On appelle **chaîne d'origine**, toute chaîne de caractères avant sa compression.
- On appelle **chaîne compressée de la chaîne d'origine ch,** la chaîne de caractères contenant une occurrence unique (un seul caractère) de toute suite de caractères identiques consécutifs se trouvant dans la chaîne d'origine ch .

Exemple :

- si **ch = AAARTTAAVVTTT ,**sa chaîne compressée **= ARTAVT**
- Soient ch une **chaîne d'origine** et **s sa chaîne compressée**. On appelle **tableau d'occurrences** de **s dans ch**, un tableau d'entiers contenant le nombre des répétions successives de chaque caractère de **s** dans la chaîne d'origine **ch**

Exemple :

- Si la chaîne d'origine **ch = BBBXXXXXXBBBYYYYYB** et **s sa chaîne compressée,** alors le tableau **d'occurrences** de **s** dans **ch** est {**3, 6, 3, 5,1}**

Mise en œuvre de l'algorithme RLE en langage C :

Dans ce problème, il s'agit d'écrire des fonctions en **langage C** utilisant l'algorithme **RLE** pour la compression et la décompression de chaînes de caractères

Rappels :

- Une chaîne de caractères en langage C est un tableau de caractères se terminant par le caractère spécial '**\0**' (**'\0'** ne fait pas partie de la chaîne mais indique sa fin)
- Le premier élément d'un tableau en langage C a pour indice **zéro (0)**

Remarques

- Aucune fonction de la bibliothèque du **langage C** ne peut être utilisée**.** Toute fonction doit être déclarée et définie avant son appel.
- On suppose que toutes les chaînes de caractères manipulées dans ce problème, ont des longueurs strictement inférieurs à **255.**

Partie I- 1 :

Dans cette partie, on se propose d'écrire des fonctions **en langage C** sans paramètres pour la compression et la décompression de chaîne de caractères
On suppose avoir déclaré comme suit les variables globales **ch** pour la chaîne d'origine et **ch_compress** pour sa chaîne compressée.

```
char ch [255];                  /* chaîne d'origine
char ch_compress[255] ;     /* chaîne compressée de ch
```

Question I- 1 -1

+Déclarer en **langage C** une variable globale tableau de nom **occur** destinée à contenir le tableau d'occurrences de **ch_compress** dans **ch**.

Question I- 1 -2

+Ecrire le code d'une fonction d'entête : **void compresser()** qui met dans la chaîne **ch_compress** , la chaîne compressée de **ch**

Exemple : - Si la chaîne **ch = AAAXRRZZZAAAA**
- Après l'appel de la fonction **compresser()** , **ch_compress =AXRZA**

Question I- 1 -3
+Ecrire le code d'une fonction d'entête: **void occurrence()** qui remplit le tableau **occur** (occurrences de **ch_compress** dans **ch)**

Exemple
- Si la chaîne d'origine **ch = DDDRRRRDTTYYYY**,
- Après l'appel de la fonction **occurrence (),** les éléments du tableau **occur** sont : **occur[0]=3, occur[1]=4, occur[2]=1, occur[3]=2, occur[4]=4**

Question I- 1-4
+Ecrire le code d'une fonction d'entête : **void decompresser()** qui permet de décompresser la chaîne **ch_compress** en chaîne d'origine **ch**

Exemple :
- Si la chaîne **ch_compress = BACB** et **occur={3,5,4,1}**
- Après l'appel de la fonction **decompresser(),ch=BBBAAAAACCCCB**

****

Partie I- 2 :
Dans cette partie, on se propose de compresser une chaîne de caractères sans utiliser le tableau d'occurrences et en écrivant des fonctions avec des paramètres

On appelle **chaîne codée** d'une chaîne d'origine, la chaîne de caractères contenant une occurrence unique de toute suite de caractères identiques consécutifs se trouvant dans la chaîne d'origine, suivie du nombre d'occurrence du caractère dans la chaîne d'origine .

Remarques :
- On suppose que le nombre de caractère de toute suite de caractères consécutifs ne dépasse pas 9.
- Pour convertir un chiffre en un caractère on lui ajoute 48 (le code de zéro), exemple : si ch est une chaîne de caractères et j un entier égal à 5 alors: ch[i]=j+48 , donne ch[i]='5'.

Question I- 2-1 :
+Ecrire le code d'une fonction d'entête : **void coder(char *s,char *s_codee)** qui permet de mettre dans la chaîne d'adresse **s_codee** , la chaîne codée de **s.**

Remarque :
Exemple : - Si la chaîne **s = EEEEEAAACCCCAAX**
- Après l'appel de la fonction **coder(s,s_codee) , s_codee=E5A3C4A2X1**

Question I- 2-2
+Ecrire le code d'une fonction en langage C d'entête : **void transformer(char *s)** qui permet de transformer le contenu de la chaîne d'origine s pour qu'elle devienne sa chaîne codée.

Exemple :
- Si la chaîne **s= EEEEEAAACCCCAAX**
- Après l'appel de la fonction **transformer(s) , s=E5A3C4A2X1**

Problème II : Mise à jour des factures d'un magasin

On se propose d'écrire une application en **langage C** qui permet de mettre à jour des factures d'un magasin qui n'ont pas encore été payées. Chaque facture est définie selon le type facture suivant :

```
typedef struct
{ int num ;            // numéro de la facture
char nom[20];          // le nom du client
float prix ;           // le prix à payer en dirhams
 int anlimite ;        // année limite de paiement
} facture ;
```

Pour ce faire :
- **on suppose avoir déjà crée** dans la mémoire dynamique (le tas) une liste chaînée représentant ces factures. Cette liste est définie comme suit :

```
typedef struct tlist
{ facture  info ;          // données de la facture
struct tliste *suiv;       // adresse de l'élément suivant
} liste;
```

- On suppose aussi avoir les déclarations globales suivantes :

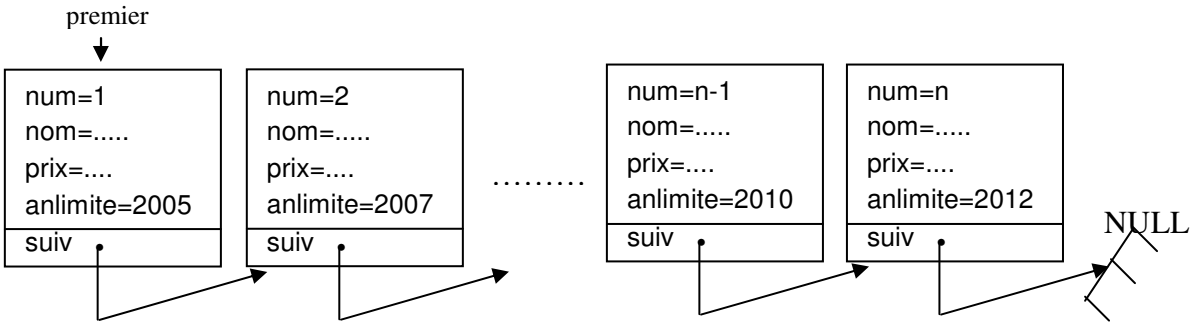
```
const int ancourant=2010 ;     // année en cours
liste *premier;                // pointeur sur le premier élément de la liste
```

- Dans cette liste les éléments sont reliés par **ordre croissant** d'année limite de paiement. Chaque élément représentant une facture de **numéro i** contient dans son champ **suiv**, l'adresse de l'élément suivant représentant une facture de numéro (**i+1).** En plus**,** l'année limite de paiement de la facture numéro **i** est strictement inférieur à l'année limite de paiement de la facture numéro(**i+1).**

Remarques :

- La première facture de la liste a le numéro **1**
- La dernière facture de la liste a dans son champ **suiv** la valeur **NULL**

Exemple de représentation de cette liste de factures



Remarque : Aucune fonction de la bibliothèque du langage C ne sera utilisée

Question II- 1

+Ecrire le code d'une fonction d'entête **int nombre()** qui retourne le nombre de factures représentées dans la liste chaînée identifiée par l'adresse **premier** qui est l'adresse de son premier élément

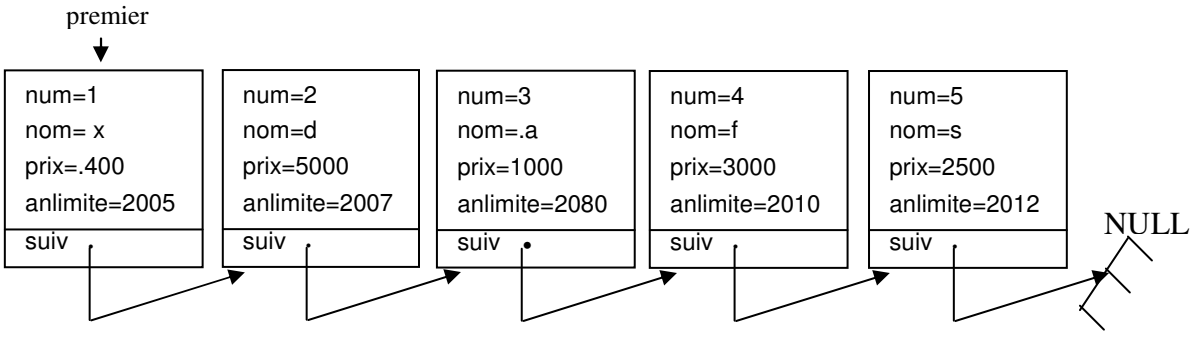
Question II- 2

+Ecrire le code d'une fonction d'entête **void supprimer(int val)** qui supprime de la liste la facture de numéro **val** (cette facture vient d'être payée) et met à jour les numéros des factures de telle sorte à garder la propriété (une facture de **numéro i** est relié à l'élément suivant représentant une facture de numéro (**i+1)).**

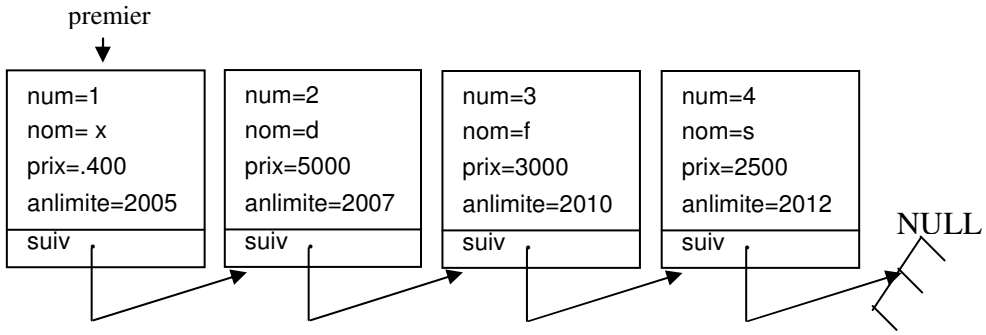
Dans le cas où val ne correspond à aucun numéro d'une facture de la liste, la fonction ne fait rien

Exemple

Etat de la liste avant l'appel de la fonction supprimer



Etat de la liste après l'appel de la fonction supprimer(3)

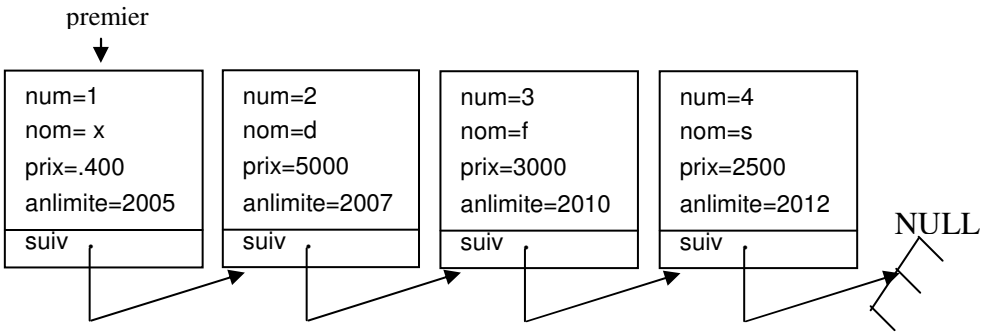


Question II- 3

+Ecrire le code d'une fonction d'entête : **int numero()** qui retourne le plus petit numéro d'une facture dont l'année limite est supérieure ou égale à l'année en cours (le plus petit numéro d'une facture en règle), Cette fonction doit retourner 0 si toutes les factures de la liste ont une année limite de paiement strictement inférieure à l'année en cours

Exemple d'exécution

- Soit la liste suivante :



L'appel de la fonction **numero() retourne : 3**

Question II- 4

+Ecrire le code d'une fonction d'entête **void maj ()** qui permet d'augmenter de 10% par an de retard les prix de toutes les factures de la liste dont l'année limite de paiement est strictement **inférieure à** l'année en cours

Exemple:

Soit la facture ayant l'année limite de paiement 2008 et le prix=100dh

- Si l'année en cours est 2010 , le nouveau prix deviendra 121 dh (De 2008 à 2009 on augmente le prix de 10% soit 110, de 2009 à 2010(l'année en cours), on augmente 110dh de 10% soit 121 dh)

FIN DE L'épreuve