A2023 – INFO MPI 0

ÉCOLE DES PONTS PARISTECH,
ISAE-SUPAERO, ENSTA PARIS,
TÉLÉCOM PARIS, MINES PARIS,
MINES SAINT-ÉTIENNE, MINES NANCY,
IMT ATLANTIQUE, ENSAE PARIS,
CHIMIE PARISTECH - PSL.

Concours Mines-Télécom

CONCOURS 2023

Epreuve de Test

PREMIÈRE ÉPREUVE D'INFORMATIQUE

MPI

Durée de l'épreuve : 3 heures

L'usage de la calculatrice et de tout dispositif électronique est interdit.

Les candidats sont priés de mentionner de façon apparente
sur la première page de la copie :

INFORMATIQUE I - MPI

Cette épreuve concerne uniquement les candidats de la filière MPI.

Ce sujet est proposé à titre d'exemple. Il montre comment le programme de MP2I/MPI, en vigueur à partir de septembre 2021, pourrait être exploité dans le cadre d'une épreuve d'informatique écrite de la banque Mines-Ponts.

Compte tenu des nombreuses annales disponibles et écrites en OCaml (au programme de l'option informatique de MP), ce sujet a été écrit exclusivement en langage C. Il présente comment certaines des nouveautés du programme de MPI pourraient être examinées. Au cours des années et des épreuves, la totalité du programme sera évaluée par le concours.

Préliminaires

Présentation du sujet

L'épreuve est composée d'un problème unique comportant 30 questions. Dans ce problème nous étudions différentes variantes d'implémentation du type de données abstrait EnsembleEntiers, qui permet de stocker une collection d'entiers. Ce type abstrait est muni de trois primitives : l'insertion, la suppression et le test d'appartenance.

Le problème est divisé en trois sections. Dans la première section (page 2), nous implémentons EnsembleEntiers à l'aide de listes chaînées. Dans la deuxième section (page 5), indépendante de la première, nous améliorons le fonctionnement de la structure en passant à des listes à enjambements. Dans la troisième section (page 8), qui prolonge la première, nous étudions le comportement de listes chaînées dans des situations de concurrence.

Dans tout l'énoncé, un même identificateur écrit dans deux polices de caractère différentes désignera la même entité, mais du point de vue mathématique avec la police en italique (par exemple n ou n') et du point de vue informatique avec celle en romain avec espacement fixe (par exemple `n` ou `nprime`).

Travail attendu

Pour répondre à une question, il est permis de réutiliser le résultat d'une question antérieure, même sans avoir réussi à établir ce résultat. Des rappels de mathématique et de programmation sont faits en annexe et peuvent être utilisés directement.

Selon les questions, il faudra coder des fonctions à l'aide du langage de programmation C exclusivement, en reprenant le prototype de fonction fourni par le sujet, ou en pseudo-code (c-à-d. dans une syntaxe souple mais conforme aux possibilités offertes par le langage C).

Quand l'énoncé demande de coder une fonction, sauf demande explicite de l'énoncé, il n'est pas nécessaire de justifier que celle-ci est correcte ou de tester que des préconditions sont satisfaites.

Le barème tient compte de la clarté des programmes : nous recommandons de choisir des noms de variables intelligibles ou encore de structurer de longs codes par des blocs ou par des fonctions auxiliaires dont on décrit le rôle. Lorsqu'une réponse en pseudo-code est permise, seule la logique de programmation est évaluée, même dans le cas où un code en C a été fourni en guise de réponse.

1. Listes triées simplement chaînées

1.1. Autour des opérations de base

Nous supposons définies deux constantes entières `INT_MIN` et `INT_MAX` qui désignent respectivement le plus petit entier et le plus grand entier représentables par le type **`int`** en machine. Elles sont représentées par $-\infty$ et ∞ dans les schémas.

Indication C : Nous introduisons une structure de maillon constituée de deux champs par la déclaration suivante.

```
1.  struct maillon {
2.      int donnee;
3.      struct maillon *suivant;
4.  };
5.  typedef struct maillon maillon_t;
```

Définition : Dans l'ensemble de cette partie, nous réalisons le type abstrait EnsembleEntiers à l'aide d'une liste de maillons simplement chainés. Nous supposons que tous les entiers insérés, supprimés ou recherchés sont strictement compris entre `INT_MIN` et `INT_MAX`. Nous maintenons les trois invariants suivants :

✠ 1. Pour tous maillons m et m' consécutifs dans la liste chaînée, de champs `donnee` respectifs u et u', on a l'inégalité $u < u'$ (autrement dit, la liste est triée et ne contient pas de doublons).

✠ 2. La liste est encadrée par deux maillons sentinelles ayant `INT_MIN` comme champ `donnee` en tête de liste et `INT_MAX` en fin de liste.

✠ 3. Pour toute valeur entière u contenue dans l'ensemble, il existe un maillon accessible depuis le maillon sentinelle de tête ayant u comme champ `donnee`.

Par exemple, l'ensemble $\{2, -7, 9\}$ est représenté par la liste chaînée dessinée en figure 1 (où la croix représente le pointeur nul).

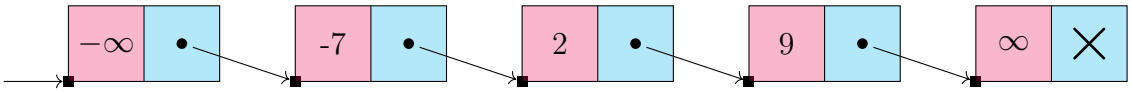

Figure 1 – Exemple d'ensemble d'entier représenté par une liste chaînée

☐ 1 – Écrire en C une fonction `maillon_t *init(void)` dont la spécification suit :
Effet : crée une copie de l'ensemble vide par l'instanciation de deux nouveaux maillons sentinelles chaînés entre eux.
Valeur de retour : pointeur vers le maillon de tête.

☐ 2 – Écrire en C une fonction `maillon_t *localise(maillon_t *t, int v)` dont la spécification suit :
Précondition : Le pointeur t désigne le maillon sentinelle de tête d'une liste chaînée.
Postcondition : En notant u le champ `donnee` du maillon désigné par la valeur de retour et u' celui du maillon successeur, on a les inégalités $u < v \leqslant u'$.

Nous souhaitons écrire une fonction `bool insere(maillon_t *t, int v)` ainsi spécifiée :
Précondition : Le pointeur t désigne le maillon sentinelle de tête d'une liste chaînée.
Postcondition : La liste désignée par le pointeur t contient la valeur entière v ainsi que les autres valeurs précédemment contenues.
Valeur de retour : Booléen `true` si la liste contient un élément de plus et `false` sinon.

☐ 3 – Présenter sous forme de croquis un jeu de données de test de la fonction `insere`, qui couvre notamment l'ensemble des valeurs de retour possibles. Dans chaque cas, on dessinera les états initial et final de la liste à la manière de la figure 1 et on donnera la valeur de retour.

Nous proposons le code erroné suivant.

```
6.  bool insere_errone(maillon_t *t, int v) {
7.      maillon_t *p = localise(&t, v);
8.      maillon_t *n = malloc(sizeof(maillon_t));
9.      n->suivant = p->suivant;
10.     n->donnee = v;
11.     p->suivant = n;
12.     return true;
13. }
```

☐ 4 – Le compilateur produit le message d'erreur : `incompatible pointer types passing 'maillon_t **' to parameter of type 'maillon_t *'`. Expliquer ce message et proposer une première correction.

☐ 5 – Discerner le ou les tests de la question 3 manqués par la fonction `insere_errone`. Corriger en conséquence la fonction `insere_errone`.

Dans ce qui suit, nous supposons que la fonction `bool insere(maillon_t *t, int v)` a été codée correctement.

☐ 6 – Écrire en C une fonction `bool supprime(maillon_t *t, int v)` dont la spécification suit :
Précondition : Le pointeur t désigne le maillon sentinelle de tête d'une liste chaînée.
Postcondition : La liste désignée par le pointeur t ne contient pas la valeur entière v mais contient les autres valeurs précédemment contenues.
Valeur de retour : Booléen `true` si la liste contient un élément de moins et `false` sinon.

☐ 7 – Calculer la complexité en temps des fonctions `insere` et `supprime`.

☐ 8 – Un programme C peut stocker ses données dans différentes régions de la mémoire. Citer ces régions. Dire dans laquelle ou dans lesquelles de ces régions la valeur entière 717 est inscrite lorsque nous exécutons le programme suivant.

```
14. int v = 717;
15. int main(void) {
16.     maillon_t *t = init();
17.     insere(t, v);
18.     return 0;
19. }
```