ÉCOLE POLYTECHNIQUE

CONCOURS D'ADMISSION 2003 **FILIÈRES PSI ET PT**

ÉPREUVE FACULTATIVE D'INFORMATIQUE

(Durée : 2 heures)

L'utilisation des calculatrices **n'est pas autorisée** pour cette épreuve.

⋆⋆⋆

Avertissements :
Le langage de programmation choisi par le candidat doit être spécifié en tête de la copie.
On attachera une grande importance à la concision, à la clarté, et à la précision de la rédaction.

Codage cyclique

On se propose d'implanter plusieurs techniques de détection d'erreurs dans la transmission de données sur des moyens de communication non fiables. Les données transitant sur le réseau sont des séquences de *bits* que l'on notera **0** et **1**. Ces séquences de bits sont découpées en mots $\langle b_0, b_1 \ldots, b_{n-1}\rangle$ de longueur n ($n > 0$) où n est une constante globale, fixée une fois pour toutes. Ces mots sont représentés par des tableaux d'entiers b de longueur n dont l'élément b_i à l'indice i vaut 0 ou 1 ($0 \leq i < n$). Le nombre d'erreurs de transmission du mot b est le nombre de bits ayant changé de valeur après la transmission.

Le temps d'exécution $T(f)$ d'une fonction f de la variable b est le nombre d'opérations élémentaires (addition, soustraction, multiplication, division, affectation) nécessaire au calcul de $f(b)$. Lorsque ce temps d'exécution dépend d'un paramètre p, il sera noté $T_p(f)$. De même l'espace mémoire $E(f)$ est la somme des tailles des données intermédiaires nécessaires pour le calcul de $f(b)$. On dira que $T_p(f)$ est d'ordre $O(g(p))$ pour signifier qu'il existe $K > 0$ tel qu'on a $T_p(f) \leq Kg(p)$ pour tout p.

Partie I – Bit de parité

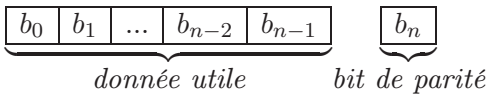
Le ou-exclusif $x \oplus y$ de deux bits x et y est défini par la table des valeurs suivante :

x \ y	0	1
0	0	1
1	1	0

On remarque qu'on a $0 \oplus x = x \oplus 0 = x$ et $x \oplus x = 0$ pour toute valeur de x. Donc x est son propre opposé pour $\oplus$.

Question 1 Ecrire la fonction `ou_exclusif` qui calcule le ou-exclusif $x \oplus y$ des deux bits x et y pris comme arguments.

La technique du *bit de parité* consiste à rajouter un bit b_n (le bit de parité) aux données utiles $\langle b_0, b_1 \ldots, b_{n-1} \rangle$ de façon à ce que $\langle b_0, b_1, \ldots, b_{n-1}, b_n \rangle$ ait un nombre pair de bits à **1**. Ainsi, pour $n = 7$, le bit de parité du tableau $\langle 1, 1, 0, 1, 1, 1, 0 \rangle$ est 1. Le mot transmis sur le réseau est $\langle 1, 1, 0, 1, 1, 1, 0, 1 \rangle$. Schématiquement, on aura :



Question 2 Ecrire la fonction `bit_parite` qui calcule le bit de parité du tableau b, contenant la suite de bits $\langle b_0, b_1, \ldots, b_{n-1} \rangle$.

Question 3 Combien d'erreurs de transmission, la technique du bit de parité permet-elle de détecter dans un mot b de longueur n ? Justifier le.

Partie II – Codage CRC

On peut considérer le tableau b de n bits $\langle b_0, b_1, \ldots b_{n-1} \rangle$ comme les coefficients du polynôme

$$P(X) = b_0 X^{n-1} + b_1 X^{n-2} + \cdots + b_{n-2} X + b_{n-1}$$

Soit $\mathbf{B}[X]$ l'ensemble des polynômes dont les coefficients sont des bits. Le degré du polynôme $P(X) = b_0 X^{n-1} + b_1 X^{n-2} + \cdots b_{n-1}$ de $\mathbf{B}[X]$ est la valeur maximale de k tel que b_{n-1-k} ne soit pas nul.

Question 4 Ecrire la fonction `degre` prenant en argument un tableau b de coefficients représentant le polynôme $P(X)$ de $\mathbf{B}[X]$, et retournant le degré de P (Par convention, le degré du polynôme nul vaudra -1).

La somme exclusive des deux polynômes $P(X)$ et $Q(X)$ de $\mathbf{B}[X]$ dont les coefficients sont les tableaux $\langle b_0, b_1, \ldots b_{n-1} \rangle$ et $\langle c_0, c_1, \ldots c_{n-1} \rangle$ est définie par :

$$P(X) \oplus Q(X) = (b_0 \oplus c_0) X^{n-1} + (b_1 \oplus c_1) X^{n-2} + \cdots + (b_{n-1} \oplus c_{n-1})$$

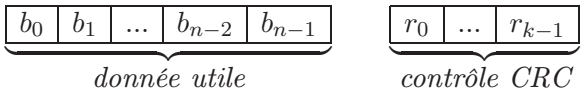
Remarque :
$P(X)$ est son propre opposé, puisque $P(X) \oplus P(X) = 0$ pour tout polynôme $P(X)$ de $\mathbf{B}[X]$.

Question 5 Ecrire la fonction `plus` prenant comme arguments deux tableaux b et c représentant deux polynômes $P(X)$ et $Q(X)$ de $\mathbf{B}[X]$, deux indices i et j dans b et c, et une longueur ℓ, et qui modifie le tableau b pour remplacer ses coefficients b_i, b_{i+1}, ... $b_{i+\ell-1}$ par $b_i \oplus c_j$, $b_{i+1} \oplus c_{j+1}$, ... $b_{i+\ell-1} \oplus c_{j+\ell-1}$.

La technique, dite **CRC** (*Cyclic Redundancy Check*), ajoute plus de bits de contrôle à chaque mot transmis que la méthode du bit de parité. Elle considère un polynôme générateur $G(X)$, bien choisi dans $\mathbf{B}[X]$, de degré k ($0 < k \leq n$), donné une fois pour toutes ; et elle construit pour tout mot b, correspondant au polynôme $P(X)$, le reste $R(X)$ de la division euclidienne de $P(X) \cdot X^k$ par $G(X)$.

$$R(X) = P(X) \cdot X^k \mod G(X)$$

où mod est l'opération modulo. Si $\langle r_0, r_1, \ldots r_{k-1} \rangle$ sont les coefficients de $R(X)$, la donnée transmise $\langle b_0, b_1, \ldots b_{n-1}, r_0, r_1, \ldots r_{k-1} \rangle$ correspondra à un polynôme multiple de $G(X)$. Graphiquement, on a :



Pour calculer le reste $R(X)$ de la division de $P(X) \cdot X^k$ par $G(X)$, on utilise l'algorithme classique de la division. On aligne les bits valant 1 les plus à gauche du dividende et du diviseur, puis on retranche le diviseur au dividende (grâce à l'opération $\oplus$). Le degré du résultat est strictement inférieur à celui du dividende. Et on recommence la division en prenant le résultat comme nouveau dividende, jusqu'à ce que son degré soit strictement inférieur à k. Ainsi, par exemple, pour les tableaux $b = \langle 0, 1, 1, 1, 0, 1 \rangle$ et $g = \langle 0, 1, 0, 1 \rangle$ (et donc $k = 2$), les étapes successives de la division donnent :

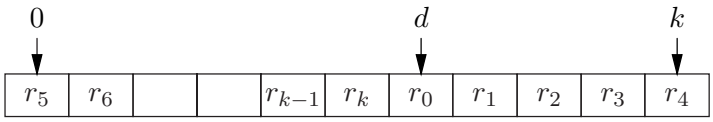
$$\begin{array}{cccccccc}
\langle 0, & 1, & 1, & 1, & 0, & 1, & 0, & 0 \rangle \\
\langle 0, & 0, & 1, & 0, & 0, & 1, & 0, & 0 \rangle \\
\langle 0, & 0, & 0, & 0, & 1, & 1, & 0, & 0 \rangle \\
\langle 0, & 0, & 0, & 0, & 0, & 1, & 1, & 0 \rangle \\
\langle 0, & 0, & 0, & 0, & 0, & 0, & 1, & 1 \rangle
\end{array}$$

D'où la valeur $\langle 1, 1 \rangle$ pour le CRC.

Question 6 Ecrire la fonction `crc` prenant en argument deux tableaux b et g et qui retourne le tableau de bits correspondant aux coefficients du CRC du mot représenté par b par rapport au polynôme générateur représenté par le tableau g. (La valeur de l'argument b ne doit pas être modifiée). Donner un ordre de grandeur du temps et de l'espace mémoire pris par `crc` en fonction de la longueur n de b et du degré k de g.

Pour réduire l'espace mémoire utilisé, on peut ranger les résultats partiels du reste (dans le calcul de la division) dans un registre circulaire de $k + 1$ bits. Ce registre est représenté par

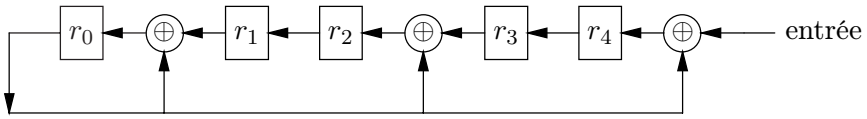
un tableau r de bits de taille $k+1$ et une variable d indique l'emplacement de son bit le plus significatif. Plus exactement le registre est organisé comme suit :



Question 7 Ecrire une nouvelle version `crc1` de la fonction `crc`, qui ne prend qu'un espace mémoire en $O(k)$. (La valeur de l'argument b doit toujours être inchangée)

Les polynômes générateurs habituels utilisés pour le CRC sont $X^{16}+X^{15}+X^2+1$ (CRC-16), $X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^8+X^7+X^5+X^4+X^2+X+1$ (CRC-Ethernet), etc. On réalise des circuits pour calculer rapidement les valeurs du CRC.

Question 8 Expliquer le circuit suivant de calcul du CRC avec $G(X)=X^5+X^4+X^2+1$ comme polynôme générateur (Initialement, tous les r_i sont nuls ; les bits du mot b d'entrée arrivent sur la droite, suivis de 4 bits valant 0 ; le circuit est synchronisé par une horloge globale qui à chaque tranche de temps décale tous les r_i d'un cran vers la gauche ; le résultat est le tableau de bits $\langle r_0, r_1, r_2, r_3, r_4\rangle$ une fois que tous les bits d'entrée ont été lus).



On peut montrer que la technique du bit de parité est un cas particulier de la méthode de détection d'erreurs par CRC.

Question 9 Quel est le polynôme générateur correspondant à la méthode du bit de parité ?

* *
*

Épreuve facultative d'Informatique, filières PSI et PT

Rapport de M. Marc Pouzet, correcteur.

C'était la deuxième édition de cette épreuve spécifique à l'École Polytechnique et commune aux filières PT et PSI. L'épreuve était facultative et seules les copies des candidats admissibles ont été corrigées. Le nombre de copies était faible (12 pour la filière PT, et 36 pour la filière PSI).

Résultats

La moyenne des candidats de la filière PSI est de 13.6 avec un écart-type de 3.2. La moyenne de la filière PT et de 11.4 avec un écart-type de 3.8. La répartition est la suivante :

	$0 \leq N < 4$	$4 \leq N < 8$	$8 \leq N < 12$	$12 \leq N < 16$	$16 \leq N < 20$
PSI	0%	0%	22%	47%	30%
PT	0%	0%	41%	41%	17%

Cette année les copies étaient bonnes, voire très bonnes (44% de copies entre 12 et 16 et 10% de copies entre 18 et 20 sur l'ensemble des deux filières). L'absence de notes entre 0 et 8 montre que tous les candidats savent écrire des programmes. De plus, tous les candidats ont su répondre aux premières questions.

Les coefficients des questions, identiques pour les filières PT et PSI sont les suivants :

Question	1	2	3	4	5	6	7	8	9
Coef.	2	4	2	5	5	7+3	10	4	3

Le langage Maple a été choisi par la majorité des candidats. Les autres langages utilisés étaient Java, Pascal, C et Mathematica.

Évaluation

Cette épreuve était composée de deux parties. La première partie présentait la technique du bit de parité et consistait à écrire des fonctions simples de parcours de tableaux. La seconde présentait la technique des CRC et aboutissait à l'écriture de fonctions de codage basées sur la division euclidienne de polynômes. Les structures de données et de contrôle nécessaires se résumaient aux tableaux d'entiers, boucles `for`, conditionnelles, séquence, définition et utilisation de fonctions. L'utilisation de la récursivité n'était pas indispensable dans cette épreuve.

Dans ce type d'épreuve, il est fortement conseillé d'utiliser des structures de données et de contrôle élémentaires (boucles `for`, tableaux) plutôt que des primitives spécifiques de Maple (opérateur `seq`). L'utilisation de ces primitives nécessite une bonne connaissance de Maple et elles sont utilisées le plus souvent de manière approximative (problèmes de type, de bornes) ce qui fait perdre des points. De plus, les structures algorithmiques élémentaires se prêtent mieux à un calcul de complexité.

Toutes les questions (sauf les **questions 3, 8** et **9**) demandaient d'écrire du code. Le code de chaque question étant court, la correction commence par la lecture du code. Il est donc essentiel que ce code soit bien présenté et les noms des variables choisis de manière judicieuse (un commentaire décrivant succintement le rôle de chaque variable peut être une aide précieuse et il faut veiller à conserver les noms des fonctions données dans l'énoncé). Si le code est juste, je ne tiens pas trop compte des explications (souvent approximatives). Sinon, je les lis attentivement pour comprendre l'idée du candidat. Toutefois, le candidat a déjà perdu une bonne partie de ses points. La situation peut être sauvée partiellement lorsque le candidat à eu l'idée de formuler un invariant de boucle (sous forme de suites récurrentes par exemple).

Lors de la lecture des programmes, les petites erreurs de syntaxe (oubli d'un `end`, d'une marque de fin de boucle) ne sont pas trop prises en compte dès lors qu'il n'y a pas d'ambiguïté possible et qu'il semble qu'elles seraient immédiatement corrigées lors d'une programmation sur machine. En revanche, les erreurs de type et une absence d'initialisation des variables sont systématiquement sanctionnées.

Pour chaque question, j'indique la moyenne obtenue (en ramenant la note de la question sur 5).

Question 1 [4.1] Presque tous les candidats ont répondu correctement à cette question.

S'agissant d'une question facile, la solution devait être parfaite. Un programme compliqué (e.g, non fonctionnel) ou faisant intervenir des variables intermédiaires inutiles a été sanctionné.

Question 2 [3] Cette question a également été traitée par la majorité des candidats. Il fallait clairement privilégier ici une solution modulaire effectuant un appel à la fonction `ou_exclusif` définie précédemment. Une solution non modulaire ou trop compliquée (e.g, calcul de la somme puis du reste modulo 2) ou ne correspondant pas exactement à l'énoncé (e.g, retournant un booléen) n'a pas reçu tous les points.

Question 3 [4.5] Cette question ne demandait pas d'écrire du code. Elle a été traitée par tous les candidats. Une réponse partielle (e.g, indiquant que le

bit de parité permet de détecter une seule erreur de transmission plutôt qu'un nombre impair d'erreurs) ou l'absence de justification ont été sanctionnées.

Question 4 [3.7] La solution attendue consistait à effectuer une boucle (d'indice i, pour $0 \leq i < n$) parcourant le tableau b dans l'ordre des indices croissants jusqu'à ce que b_i soit non nul.

Les raisons principales pour ne pas obtenir tous les points ont été une mauvaise initialisation des variables (ou plus souvent un oubli) ou une erreur dans les bornes de la boucle. Certains candidats ont suivi fidèlement la définition mathématique du degré donnée dans l'énoncé et qui conduisait à un algorithme un peu moins naturel car parcourant tout le tableau même dans le cas d'un polynôme de fort degré.

Question 5 [4.6] Cette question a été bien traitée par l'ensemble des candidats et il suffisait ici de suivre la définition donnée dans l'énoncé.

Cette question étant facile, une absence d'initialisation ou une solution ne suivant pas fidèlement l'énoncé (e.g, retournant un nouveau tableau contenant la somme) ont été sanctionnées.

Question 6 [2.1+1.4] Cette question s'est révélée difficile. Elle demandait d'écrire le code du `crc` basé sur le calcul du reste de la division euclidienne puis de donner un ordre de grandeur de sa complexité.

L'énoncé suggérait d'écrire une seule boucle calculant itérativement le reste de la division euclidienne. Le reste partiel rp (à l'itération j) peut être calculé en fonction du reste partiel (à l'itération $j-1$) et du diviseur g. En initialisant le reste partiel à un tableau de longeur $n+k$ contenant $\langle b_0, \ldots, b_{n-1}, 0, \ldots, 0\rangle$, le `crc` se retrouve dans les k derniers éléments du reste partiel après n itérations. Le reste partiel rp à l'itération j (noté $\langle rp_0, \ldots, rp_{n+k-1}\rangle^j$ est défini par la récurrence suivante :

$$
\begin{array}{lcl}
dg & = & \texttt{degre}(g) \\
\langle rp_0, ..., rp_{n+k-1}\rangle^j & = & \texttt{plus}(\langle rp_0, ..., rp_{n+k-1}\rangle^{j-1}, g, n-dg-1, dg+1) \\
 & & \text{si } rp_{j-1} = 1 \\
\langle rp_0, ..., rp_{n+k-1}\rangle^j & = & \langle rp_0, ..., rp_{n+k}\rangle^{j-1} \text{ sinon} \\
\langle rp_0, ..., rp_{n+k-1}\rangle^0 & = & \langle b_0, ..., b_{n-1}, 0, ..., 0\rangle
\end{array}
$$

Cette solution ne nécessitait pas de recalcul du degré du reste partiel à chaque itération. Celui-ci peut être calculé itérativement à partir de la valeur courante de rp_j.

Il fallait privilégier ici une solution modulaire faisant intervenir la fonction `plus`. Une solution suivant fidèlement l'énoncé mais allouant un nouveau tableau pour stocker le résultat du reste partiel était sanctionnée. Une solution partielle définissant seulement la division euclidienne était fortement

sanctionnée. Enfin, une solution correcte mais recalculant le degré du reste partiel ne recueillait pas tous les points.

La solution attendue était de complexité $O(n.k)$ en temps et $O(n+k)$ en mémoire. Cependant, une réponse donnant une complexité différente, bien justifiée et en accord avec l'algorithme proposé a été peu sanctionnée.

Question 7 [0.4] Cette question faisait suite à la question précédente. Elle s'est révélée relativement difficile et très peu de candidats y ont répondu.

La solution attendue consistait à remarquer que le reste partiel peut être stocké dans un tableau de longeur $k+1$ seulement et à réécrire la fonction `plus` en conséquence.

Question 8 [0.19] Cette question ne demandait pas d'écrire du code. Peu de candidats l'ont abordée. Certains candidats se sont lancés dans des explications compliquées sans penser à donner un exemple montrant le déroulement du calcul et le contenu des registres $r_0, \ldots, r_4$ au cours du temps.

Question 9 [2] Il s'agissait dans cette question de remarquer que le calcul du bit de parité est un cas particulier de CRC en prenant $(X+1)$ comme polynôme générateur.

Certains candidats ont répondu à cette question en donnant le circuit correspondant au calcul du bit de parité (plutôt que le polynôme générateur). Cette réponse recueillait tous les points.