CC JS : Session 1 - 1h30 - Sur machine

Téléchargez et décompressez l'archive déposée sur Moodle pour cet examen. Le dossier résultant contient différents fichiers à réutiliser ou à compléter. A l'issue du temps imparti, archivez votre dossier et déposez l'archive sur Moodle.

La page à développer traite de l'arithmétique modulaire dans les anneaux quotients $\mathbb{Z}/n\mathbb{Z}$. On rappelle que, pour $n \in \mathbb{N}^*$, l'anneau $\mathbb{Z}/n\mathbb{Z}$ contient n éléments notés $\overline{0}, \ldots, \overline{n-1}$ représentant les n classes de congruence des entiers relatifs modulo n. La classe $\overline{k}$ contient les entiers dont le reste de la division euclidienne par n vaut k. Par exemple, $\overline{0}$ contient les multiples de n.

La page est organisée en une grille de 5 conteneurs (`fieldset`) :

- "ORDRE" permet de choisir l'ordre n de l'anneau $\mathbb{Z}/n\mathbb{Z}$ et de visualiser ses éléments sous forme de tableau.
- "CALCUL" affiche une addition et une multiplication sur $\mathbb{Z}/n\mathbb{Z}$ en réactualisant arguments et résultats toutes les 3 secondes.
- "INVERSIBLES" affiche une réplique du premier tableau qui permet de tester interactivement si un élément de l'anneau est inversible pour la multiplication en cliquant sur sa case.
- "THEOREME ..." illustre le théorème des restes chinois : étant donnés 2 entiers co-premiers n_1 et n_2, il existe un unique entier x dans $\{0, \ldots, n_1 * n_2 - 1\}$ ayant a_1 et a_2 pour restes modulo n_1 et n_2, respectivement. L'utilisateur choisit deux restes en cliquant sur la case (a_1, a_2) d'une matrice $n_1 \times n_2$ et l'entier x résultat s'affiche alors dans la case. n_1 et n_2 sont modifiables et la matrice est réactualisée le cas échéant.
- "CHIFFREMENT RSA" illustre les 2 étapes du chiffrement RSA par le biais d'un formulaire permettant de créer clés publique et privée puis de chiffrer et déchiffrer des messages sous forme d'entiers.

Les 3 premiers conteneurs sont liés et réactualisés dès lors que l'utilisateur change l'ordre n. Les 2 autres sont indépendants. La Figure 1 illustre la page en cours d'utilisation qui peut être testée avec ce démonstrateur.

Le dossier décompressé contient les fichiers suivants :

- **arithmod.html** : le fichier `HTML` de la page web.
- **arithmod.css** : la feuille de styles.
- **arithmod.js** : le module `JS` à compléter.
- **solution.js** : le module chiffré des solutions aux questions.
- **ZnZ.js** : un module implémentant une classe pour calculer sur les anneaux $\mathbb{Z}/n\mathbb{Z}$.
- **utils.js** : un module de fonctions utilitaires.
- **pgcd.php** : un script `PHP` de calcul de PGCD.
- **primes.json** : un listing de nombres premiers.

Le site est fonctionnel au chargement grâce à l'import de **solution.js** dans le fichier **arithmod.js**. Vous devrez écraser les appels du type `solution.f()` pour répondre à chaque question. Vous testerez votre page sous Firefox [1] en utilisant le conteneur Docker et son serveur web. Exercices et questions peuvent être traités dans n'importe quel ordre.

1. Pensez à désactiver le cache.

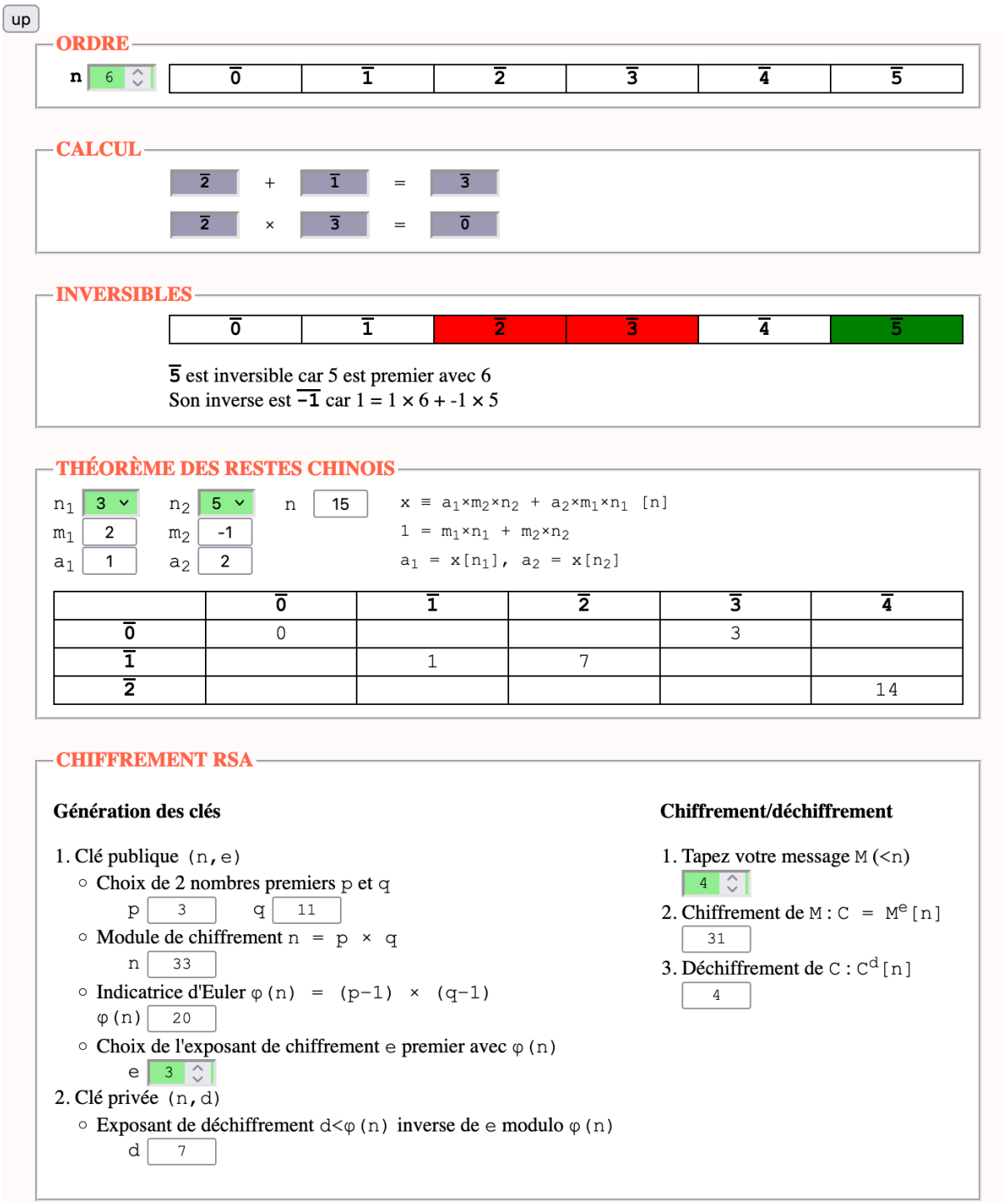

FIGURE 1 – Arithmétique sur $\mathbb{Z}/n\mathbb{Z}$ et chiffrement

---

Exercice 1. Conteneurs ORDRE et CALCUL

Re-implémentez les fonctions décrites ci-dessous.

1. `générerTableau1(k)` crée ou remet à jour l'unique ligne `TR` du premier tableau `HTML` pour afficher les `k` classes de congruence $\overline{0}, \ldots, \overline{k-1}$:

- elle s'appuie sur l'utilitaire `générerCasesCongruence()` pour générer et classer les cases
- elle ne supprime jamais le tableau.

2. `actualiserChampsCalcul()` actualise les champs de calcul toutes les 3 secondes. Chaque pas consiste à :

- tirer aléatoirement 4 valeurs dans $0..n-1$ et les afficher dans les 4 champs "arguments",
- afficher le résultat des opérations $+$ et $*$ sur ces arguments dans les champs "résultats" en déléguant le calcul aux méthodes `plus` et `times` de la classe `ZnZ`,
- appliquer la couleur de fond renvoyée par appel à l'utilitaire `rgb` aux 6 champs.

Exercice 2. Conteneur INVERSIBLES

Re-implémentez les fonctions décrites ci-dessous.

1. `générerTableau2()` crée ou remet à jour le second tableau `HTML` qui est identique au premier mais a pour classes `CSS` `c-inversibles-1` et `math`.

2. `écouterTableau2()` traite tout clic sur le second tableau `HTML` comme suit :

- requête le script **pgcd.php** en `HTTP POST` en lui communiquant les clés `x` et `y` fixées respectivement à l'ordre n choisi par l'utilisateur et à l'entier v correspondant à la classe de congruence affichée dans la case cliquée. Le script renvoie un objet `JSON` à propriétés entières de clés `x`, `y`, `gcd`, `cx` et `cy` où $gcd = PGCD(x, y) = cx * x + cy * y$.
- en cas de succès, écrase le contenu du paragraphe qui suit le tableau avec une phrase indiquant si v est inversible modulo n (quand le PGCD de v et n vaut 1) ou non et colorie le fond de la case en vert ou en rouge selon l'inversibilité obtenue.

Exercice 3. Conteneur THEOREME DES RESTES CHINOIS

Re-implémentez les fonctions décrites ci-dessous.

1. `générerN1N2()` requête en `HTTP GET` le fichier **primes.json** qui contient les 100 premiers entiers premiers. En cas de succès, génère les options des 2 menus déroulants "n_1" et "n_2" comme suit :

- ne conserve que les 30 premiers entiers de la liste
- pré-sélectionne le premier entier (resp. second) pour le premier menu (resp. second).

2. `écouterTableau3()` traite tout clic sur le troisième tableau `HTML` (matrice $n_1 \times n_2$ avec en-têtes) comme suit :

- ignore le clic s'il porte sur une case d'en-tête, sinon
- extrait les valeurs n_1 et n_2 des champs correspondants
- extrait les numéros de ligne a_1 et de colonne a_2 de la case ciblée : ces numéros sont enregistrés dans son attribut `title` au format `a1:a2`.
- appelle la méthode `chineseTheorem()` de la classe `ZnZ` sans fournir l'argument optionnel
- renseigne les champs "n", "m_1", "m_2", "a_1" et "a_2" en utilisant l'objet renvoyé
- écrase le contenu texte de la case en appelant la méthode `modulo` de `ZnZ` avec la valeur de la propriété `"x"` de l'objet réponse.

Exercice 4. Conteneur CHIFFREMENT RSA

Re-implémentez la fonction décrite ci-dessous.

1. `écouterE()` traite toute modification du champ "e" comme suit :

- vérifie que l'entier choisi e est premier avec l'entier $\varphi(n)$ affecté au champ "φ(n)" en utilisant la méthode `pgcd` de la classe `ZnZ` (e et $\varphi(n)$ sont co-premiers ssi $PGCD(e, \varphi(n)) = 1$) sinon l'incrémente automatiquement et force le champ "e" au premier exposant correct trouvé.
- extrait l'inverse d de e modulo $\varphi(n)$ fourni par la méthode `pgcd` ($PGCDE(e, \varphi(n)) = 1 = d * e + k * \varphi(n)$) et remplit le champ correspondant avec d si d est positif ou $\varphi(n) + d$ sinon.

Exercice 5. Customisation

1. Les 5 conteneurs sont organisés sous la forme d'une grille `CSS` à 5 lignes (voir **arithmod.css**). Chacun est placé sur une unique ligne dont le numéro est donné par la propriété `CSS grid-row-start`, la propriété `grid-row-end` du conteneur étant donc fixée à `1+grid-row-start`.

La fonction `commuter()` traite tout clic sur le bouton "up" en déplaçant le conteneur du haut de page en bas de page, cad. sur la dernière ligne de la grille, et en faisant remonter les autres d'une ligne. La Figure 2 illustre la page obtenue après 2 clics.

Re-implémentez cette fonction en utilisant la méthode globale `getComputedStyle` pour accéder aux propriétés `CSS` en lecture.

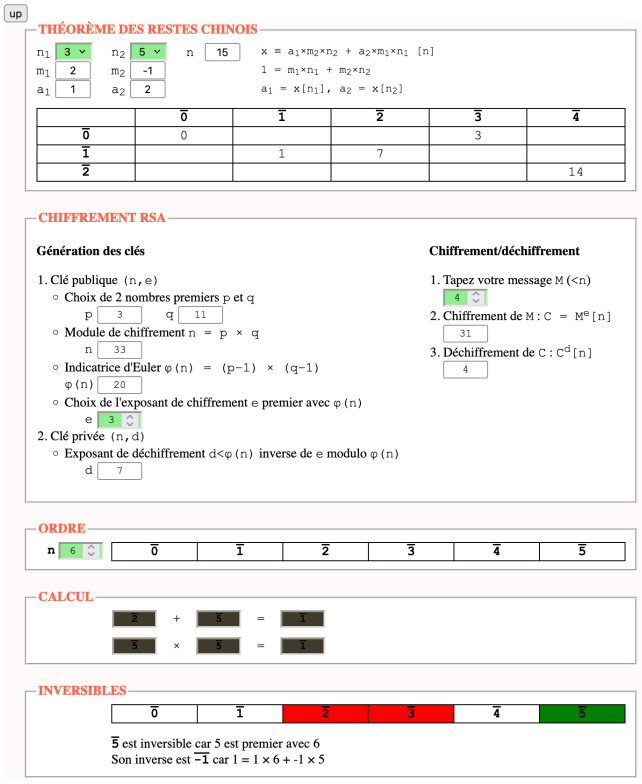

FIGURE 2 – Echange des conteneurs après 2 clics sur le bouton "up"