CORRIGÉ (LES RÉPONSES APPARAISSENT AVEC CE FOND DE CARACTÈRE)

U.B.S. - I.U.T. de Vannes

Département Informatique

Contrôle de 2e année
Date : 20/10/2014

M3101 - Principe des systèmes d'exploitation

Devoir surveillé

Responsable(s) et auteur(s) : F. Merciol – M. Le Lain

Documents personnels autorisés
Calculatrice et téléphone interdits — Durée : 2h
Les barèmes sont donnés à titre indicatif — 12 page(s) de texte

---

Conseils :

- Indiquez **votre nom** sur chaque feuille à rendre **dès qu'elles vous sont données**.
- Affin d'éviter la copie, toute réponse non justifiée sera considérée comme nulle.
- **Les parties sont indépendants** (commencez par la plus simple pour vous).
- Il est demandé des réponses à la fois claires et concises.
- **Lisez en entier** le contrôle avant de commencer à répondre.
- **Ne restez pas bloqué**, vous pourrez revenir sur une question difficile par la suite.
- **Ne brûlez pas toute votre énergie**, des calculs longs peuvent vous rapporter moins de points que la réponse à des questions de réflexion.
- **Conservez du temps pour chaque partie**
  (120 min / 20 pts => pas plus de 6 minutes par point)

Le devoir se compose de 3 parties :

1) [4 pts] questions de cours (QCM : Questions à Choix Multiple)
2) [8 pts] résolution de problème sur les tâches (en java)
3) [8 pts] exercices sur la mémoire virtuelle (calcul)

Ce corrigé est donné à titre indicatif. D'autres réponses peuvent être considérés comme justes.

Corrigé (Les réponses apparaissent avec ce fond de caractère)

CORRIGÉ (LES RÉPONSES APPARAISSENT AVEC CE FOND DE CARACTÈRE)

NOM : _____ PRENOM : _____ GROUPE TD : ____ Note :

1.) [4 pts] question de cours (QCM : Questions à Choix Multiple)

✗ Cochez la case réponse

Question	A	B	C	D
La zone de données contient les variables C dites A) automatiques B) registres C) statiques D) volatiles			○	
En Java, un sémaphore s'écrit A) avec des méthodes synchronisées B) sans « notify » C) sans « wait » D) sans méthode synchronisée	○			
D'où provient la communication par paquet de l'internet ? A) de la NSA B) du MIT C) du ministère des armées étasuniennes D) la fondation des sciences		○		
Sous Unix un « pipe » est un(e) A et logique B) ou logique c) séparateur de colonne C) file				○
Avec la commande « ls -l » le mode s (en début de ligne) désigne A) un lien symbolique B) un pipe nommé C) un périphérique spécial D) un socket				○
Un pilote sert à A) augmenter la mémoire B) communiquer avec un périphérique C) garer le bus de donnée D) imprimer		○		
La virtualisation permet A) l'archivage d'images B) l'envoi de messages C) la cohabitation de système D) la création de pseudo			○	
Le code est une zone mémoire contenant A) les règles de piraterie B) les données des fonctions C) les instructions du programme D) les règles de routage			○	
Pour quel usage était le premier réseau de l'Internet A) Astronomie B) Mathématiques C) Militaire D) Physique			○	
L'Internet A) augmente l'effet de serre B) diminue la température en Californie C) lutte contre la déforestation D) régule la fonte des glaces	○			
Le boutisme est A) la terminaison des fichiers B) un mode de démarrage C) une religion D) une représentation des entiers				○
Sur un disque, la tête désigne un A) arc de cercle B) cercle C) face D) plateau			○	
LRU : A) prend en file B) prive de ressources C) tient compte du dernier utilisé D) vide le cache			○	
Les objets Java ou C++ créés dans un processus sont placés dans quelle mémoire ? A) code B) données C) pile D) tas				○
Little-endian désigne un(e) A) apprenti du sorcier B) lecture de droite à gauche C) lecture de gauche à droite D) petit entier		○		
Lorsqu'un processus « fork », les données dans les tampons d'écriture sont A) placées dans le père B) dupliquées C) effacées D) envoyées avant		○		
Le dispositif qui attribue le temps de chaque processus sur le processeur est un(e) A) horloge B) ordonnanceur C) pile D) sémaphore		○		
Quelle partition est la plus stable ? A) / B) /boot C) /home D) /tmp		○		
Le swap sert à A) augmenter la vitesse des processus B) changer de programme C) garder des pages inutilisées D) nettoyer le disque			○	
Quelles sont les méthodes que peut bloquer le moniteur A) les récursives B) seules les non synchronisées C) seules les synchronisées D) toutes				○

Corrigé (Les réponses apparaissent avec ce fond de caractère)

CORRIGÉ (LES RÉPONSES APPARAISSENT AVEC CE FOND DE CARACTÈRE)

NOM : _____ PRENOM : _____ GROUPE TD : ____ Note :

2.) [8 pts] Exercices de réflexion

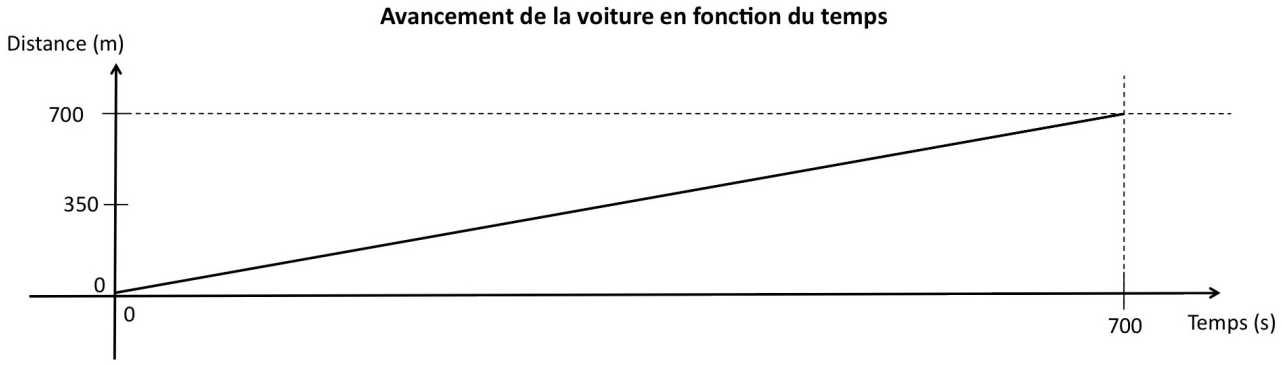
!!! Cette partie est à rendre sur copie séparée !!!

« Allons au Gois »

Pour se rendre sur l'île de Noirmoutier, il existe un passage, « Le Gois », qui se dévoile au gré des marées. Chaque jour, la mer ouvre donc ce passage une heure avant la basse mer, et une heure après. Le reste du temps il est nécessaire de passer par une autre route ayant un pont pour se rendre sur l'île.

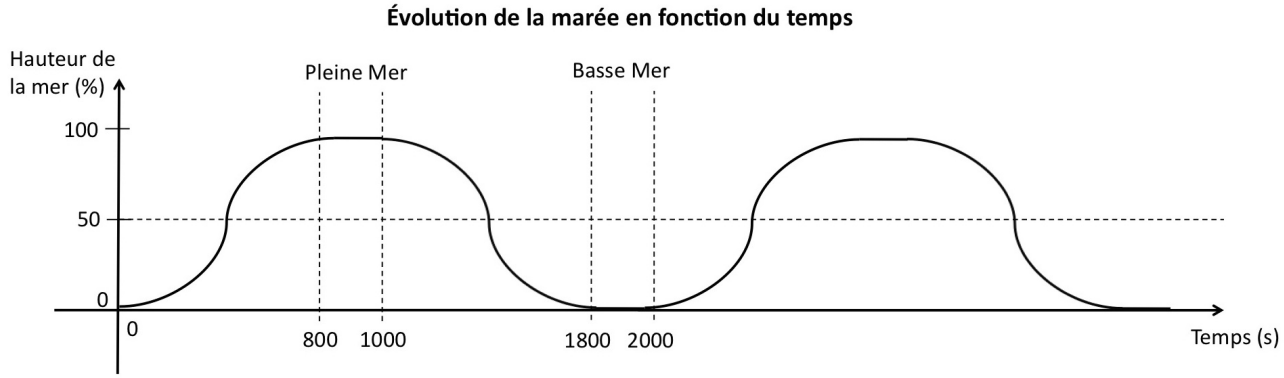
Question N° 1 – Passer

On souhaite écrire un algorithme comportant un Thread qui permet de modéliser un visiteur en voiture qui parcoure ce passage. On considère que le passage fait 700 mètres. L'avancement de la voiture, mètre par mètre, peut ainsi être simuler par une boucle comportant une pause de 1s à chaque itération. Une fois le passage parcouru par la voiture, le Thread est terminé. Écrivez cet algorithme, ainsi que le « Main » pour le tester.



Question N° 2 – Simuler la marée

On souhaite également simuler le mouvement des marées dans un algorithme comportant un Thread. Comme le montre le schéma ci-dessous, les marées montent, et descendent indéfiniment, du moins tant que notre Lune sera présente. Ce mouvement pourra donc être modélisé par l'itération d'un compteur variant de 0 à 100, puis inversement de 100 à 0, par incrément de 1, où la valeur 0 indiquera la basse mer, et la valeur 100 indiquant la pleine mer. Chaque itération comportera une attente de 8s et les phases de pleine mer et basse mer comporteront elles une attente de 200s. Écrivez cet algorithme, ainsi que le « Main » pour le tester.



Corrigé (Les réponses apparaissent avec ce fond de caractère)

Question N° 3 – Passer tant qu'il est encore temps

Avec les algorithmes ci-dessus, de nombreux visiteurs verraient leurs voitures sombrer sous l'eau, ce qui est peu souhaitable. Il faut donc modifier vos algorithmes pour que les voitures ne puissent passer sur le passage que lorsque la mer est en phase descendante à partir du niveau de moitié (indice du niveau de la mer = 50) et ce jusqu'au quart de la phase montante (soit l'indice du niveau de la mer = 25) étant donné qu'il faut compter le temps de chemin. Écrivez la modification des algorithmes permettant de modéliser cela ainsi que le « Main » pour le tester.

Passage.java

```
/**
 * Classe Passage
 * Le Main, initialise une mer et des voitures, et lance les Threads
 * @version v1.0 - 16/10/2014
 *
 * Barème - Question 3
 */
public class Passage {

    public static void main (String args[]) {
        //On initialise une mer et des voitures
        MerPassage mer = new MerPassage ();
        VoiturePassage voit = new VoiturePassage  ("Kangoo", mer);
        VoiturePassage voit2 = new VoiturePassage ("Mercedes", mer);
        VoiturePassage voit3 = new VoiturePassage ("Ford", mer);
        VoiturePassage voit4 = new VoiturePassage ("Citroen", mer);
        VoiturePassage voit5 = new VoiturePassage ("Opel", mer);

        //On lance les threads -> Autrement dit on précise au système
        //qu'il peut appeler la méthode run quand il le souhaite !
        mer.start ();
        voit.start ();
        voit2.start ();
        voit3.start ();
        voit4.start ();
        voit5.start ();
    }
}
```

Mer.java

```
/**
 * Classe mer
 * Simule le mouvement des marées au cours du temps.
 * @version 1.0 - 16/10/2014
 * Partiel M3101 - OS - Exercice 2
 *
 * Barème : Question 2 sur 2 points
 *
 */
public class Mer extends Thread {
    private int niveauMer;
    private Boolean merMontante;
    private Boolean merDescendante;

    public Mer () {
        //On début en basse mer montante.
        niveauMer = 0;
        merMontante = true;
        merDescendante = false;
    }

    public void run () {
        try {
            while (true) {
    // boucle à l'infini (à remplacer par
    // un incrément fini pour vos tests...)
                if (merMontante) {
        // S'en suit une incrémentation pour
        // la mer montante
                    niveauMer++;
                    System.out.println ("Niveau de la mer : "+
      niveauMer);
                    if (niveauMer == 100) {
                        System.out.println ("La mer est Haute / niveau
-> "+
                niveauMer);
                        sleep (200000);
                        merMontante = false;
                        merDescendante = true;
                        continue;
                    }
                } else if (merDescendante) {
        // S'en suit une décrémentation pour
        // la mer descendante
                    niveauMer--;
                    System.out.println ("Niveau mer : "+ niveauMer);
                    if (niveauMer == 0) {
                        System.out.println ("La mer est basse / niveau
-> "+
                niveauMer);
                        sleep (200000);
                        merMontante = true;
                        merDescendante = false;
                        continue;
                    }
                }
                sleep (8000);
    //Attente de 8 secondes à chaque itération
            }
        } catch (InterruptedException e) {
      //imposé par le sleep !
    }
    }

    public static void main (String[] args) {
        Mer m = new Mer(); //On créer une mer..
        m.start (); // On lance le thread
    }
}
```

Corrigé (Les réponses apparaissent avec ce fond de caractère)

MerPassage.java

```
/**
 * Classe MerPassage
 * Simule le mouvement de la marée et gère la passage du Gois.
 * @version v1.0 - 16/10/2014
 */
public class MerPassage extends Thread {
    public int niveauMer;
    private Boolean merMontante;
    private Boolean merDescendante;
    private Boolean etatPassage;

    public MerPassage () {
        //On débute en basse mer, montante.
        niveauMer = 0;
        merMontante = true;
        merDescendante = false;
    }

    /**
     * Méthode gerePassage
     * Bloque le passage si la mer est trop haute
     * Libère le passage et notify les voitures qui attendent
     * quand le niveau de la mer le permet.
     */
   public synchronized void gerePassage ()
  throws InterruptedException {
        // Si la mer monte ou qu'elle n'est pas encore assez descendu,
  // on bloque le passage
        if ((merDescendante && niveauMer > 50) ||
     (merMontante && niveauMer>25)){
            System.out.println ("Le passage est fermé,"+
       " attendez la marée basse -"+
       " Niveau de la mer : " + niveauMer);
       } else if ((merDescendante && niveauMer < 50) ||
      ( merMontante && niveauMer<25)){
     // On ouvre le passage qui est maintenant accessible
     // on prévient les voitures qui attendent
            System.out.println ("Ouverture du passage -"+
       " Bienvenue au Gois -"+
       " Niveau de la mer : "+ niveauMer);
            notifyAll ();
        }
    }

    /**
     * Méthode utiliserPassage
     * Permet à une voiture de passer lorsque la marée le permet
     * Met la voiture en attente si la marée ne le permet pas.
     * Dans notre cas la gestion du passage (l'autre méthode)
     * est très rapide, mais si elle prenait plus de temps
     * l'emploi des mots-clé synchronized sur ces deux méthodes
     * ne permettrait pas à une voiture d'utiliser le passage
     * tant qu'il y aurait une gestion du passage.
     */
   public synchronized void utiliserPassage ()
  throws InterruptedException {
        if ((merMontante && niveauMer > 25) ||
     (merDescendante && niveauMer > 50)) {
     // Si la mer ne permet pas de passer sur le passage,
     // on précise au voiture d'attendre et de se détendre
            // Les threads seront en attente passive !
            wait();
        }
        //Une fois débloqué par le notify -> On passe !
        System.out.println ("On passe sur le passage ! -"+
          " Niveau mer = "+ niveauMer +
          " Mer montante ? = "+ merMontante +
          " Mer descendante ?"+ merDescendante);
    }

    public void run () {
        try {
            while (true) {
   // boucle à l'infini (remplacer par
   // une itération fini lors de vos essais)
                if (merMontante) {
        //Si la mer monte
                    niveauMer++;
                    gerePassage ();
        // Dès que la mer bouge,
        // on gère le passage (Méthode synchronisée !)
                    System.out.println ("Mer montante -"+
          " Niveau de la mer : "+
          niveauMer);
                    if (niveauMer == 100) {
                        System.out.println ("La mer est haute -"+
              " Niveau de la mer -> "+
              niveauMer);

                        sleep (200000); //En millisecondes !
                        // On inverse la phase de la marée,
      // on passe en mer descendante
                        merMontante = false;
                        merDescendante = true;
                        continue;
                    }
                } else if (merDescendante) {
        //Si la mer descend
                    niveauMer--;
                    gerePassage ();
        // Dès que la mer bouge, on gère le passage
        // (Méthode synchronisée !)
                    System.out.println ("Mer descendante -"+
          " Niveau mer : "+niveauMer);

                    if (niveauMer == 0) {
                        System.out.println ("La mer est basse -"+
              " Niveau de la mer -> "+
              niveauMer);
                        sleep (200000); // En millisecondes !
                        // On inverse la phase de la marée,
      // on passe en mer montante
                        merMontante = true;
                        merDescendante = false;
                        continue;
                    }
                }
                sleep (8000);
    // On dors un peu, la mer ça fatigue !
    // Mais surtout parce que c'est demandé
    // et pour afficher les traces correctement
            }
        } catch (InterruptedException e) {
      //imposé par le sleep !
  }
    }
}
```

Voiture.java

```
/**
 * Classe Voiture
 * Simule l'avancement d'une voiture sur un chemin de 700 mètres
 * @version v1.0 - 16/10/2014
 *
 * Barème - Question 2
 */

class Voiture extends Thread {
    private String nomVoiture = "";
    private int distanceParcourue;

    public Voiture (String nomVoiture) {
        this.nomVoiture = nomVoiture;
        distanceParcourue = 0;
    }

    public void run () {
        try {
            for (int i = 0; i < 700; i++) {
                distanceParcourue++;
                System.out.println ("La voiture ("+ nomVoiture +
            ") a parcourue "+
            distanceParcourue + " mètres");
                sleep (1000); //(En millisecondes !)
            }
        } catch (InterruptedException e) {
      //imposé par le sleep !
  }
    }

    public static void main (String[] args) {
  // On initialise les objes voitures
        Voiture v1 = new Voiture ("Kangoo");
        Voiture v2 = new Voiture ("Picasso");
        Voiture v3 = new Voiture ("Logan");

  //On lance les threads
        v1.start();
        v2.start();
        v3.start();
    }
}
```

Corrigé (Les réponses apparaissent avec ce fond de caractère)

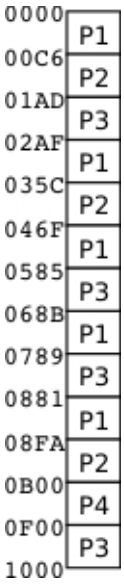
VoiturePassage.java

```
/**
 * Classe VoiturePassage
 * Simule l'avancement d'une voiture sur le passage du Gois
 * en fonction du niveau de la mer
 * @version v1.0 - 16/10/2014
 */
public class VoiturePassage extends Thread {
    private String nomVoiture = "";
    private int distanceParcourue;

    MerPassage mer;

    public VoiturePassage (String nomVoiture, MerPassage m) {
        this.nomVoiture = nomVoiture;
        distanceParcourue = 0;
        mer = m;
    }

    public void run () {
        try {
            for (int i = 0; i < 700; i++) {
    // On dors un peu avant de prendre la route pour
    // que les traces ne défilent pas trop vite
                sleep (1000);
    // On utilise le passage (Méthode synchronisée !)
                mer.utiliserPassage ();
                distanceParcourue++;
                System.out.println ("La voiture ("+ nomVoiture +
            ") a parcourue "+
            distanceParcourue + " mètres.");
    // Pour simuler l'avancement de la voiture comme
    // demandé dans l'énoncé
                sleep (1000);

            }
        } catch (InterruptedException e) {
      //imposé par le sleep !
  }
    }
}
```

CORRIGÉ (LES RÉPONSES APPARAISSENT AVEC CE FOND DE CARACTÈRE)

NOM : _____ PRENOM : _____ GROUPE TD : ____ Note :

[8 pts] exercices sur la mémoire virtuelle (calcul)



Dans cette partie nous partons de la mémoire virtuelle segmenté ci-contre.

Dans cette mémoire ont été implanté les processus P1, P2, P3 et P4. Les processus ont été découpé en segment au fils de leur pris en charge par l'ordonnanceur. Pour chaque segment, il est indiqué le nom du processus encadré de l'adresse de début et l'adresse du segment suivant. On admettra que les segments par ordre croissant d'adresse en mémoire correspondent au découpage dans le même sens de chaque processus. Le schéma ci-contre n'est bien évidement pas à l'échelle.

Ce schéma d'implantation sera utilisé pour les deux question suivantes.

2.A) [1 pt] Donnez le tableau d'occupation mémoire :

processus	taille	@ processus	(@ suivante)	segment	@ mémoire	(@ suivante)
P1	00C6	0000	00C6	1	0000	00C6
P2	00E7	0000	00E7	2	00C6	01AD
P3	0102	0000	0102	3	01AD	02AF
P1	00AD	00C6	0173	4	02AF	035C
P2	0113	00E7	01FA	5	035C	046F
P1	0116	0173	0289	6	046F	0585
P3	0106	0102	0208	7	0585	068B
P1	00FE	0289	0387	8	068B	0789
P3	00F8	0208	0300	9	0789	0881
P1	0079	0387	0400	10	0881	08FA
P2	0206	01FA	0400	11	08FA	0B00
P4	0400	0000	0400	12	0B00	0F00
P3	0100	0300	0400	13	0F00	1000

2.B) [1 pt] Donnez l'adresse processus correspondant à la mémoire virtuelle

@ mémoire	0100	0200	0500	0A00	0B00	0F00
N° segment	2	3	6	11	12	13
@ mémoire (début seg.)	00C6	01AD	046F	08FA	0B00	0F00
décalage	003A	0053	0091	0106	0000	0000
processus	P2	P3	P1	P2	P4	P3
@ processus (début seg.)	0000	0000	0173	01FA	0000	0300
@ processus	003A	0053	0204	0300	0000	

Corrigé (Les réponses apparaissent avec ce fond de caractère)

2.C) [1 pt] Quelle est la principale différence entre mémoire segmentée et mémoire paginée ?

Dans le cas de la mémoire paginée, tous fragment de processus ont la même taille est début sur une adresse multiple de d'une page. A l'exception du dernier fragment qui s'ajuste à la taille du processus est qui est de taille inférieure à une page.

2.D) [½ pt] Quelle sont les avantages de la mémoire segmentée ?

La mémoire virtuelle est exploité à son maximum. Il n'y a pas de "trou".

2.E) [½ pt] Quelle sont les avantages de la mémoire paginée ?

La simplicité des caluls d'adresse.

Par la suite nous cherchons à comparer des algorithmes de gestion de cache. Nous prendrons comme exemple des pages qui appartiennent à 2 processus en mémoire : 1, 2 et 3 sont des pages du processus A et 6 et 8 des pages du processus B. Cette séquence imaginaire montre l'entrelacement des deux processus.

1	2	3	1	2	1	2	6	6	2	1	2	3	6	8	2	3	2	2	8	3	8	3	6	1

2.F) [pt] En quoi consiste la méthode optimum de choix de défaut de page ?

On enregistre toutes les demandes de page et on réfléchit en fonction de l'avenir que l'on connait. Lorsque l'on doit faire sortir une page, on choisi celle qui ne sera plus utilisée pendant la plus grande période de temps.

Corrigé (Les réponses apparaissent avec ce fond de caractère)

2.G) [pt] Pourquoi n'est-elle pas mis en œuvre ?

Elle oblige à connaiter l'avenir. Ce n'est pas facile à mettre en oeuvre.

2.H) [pt] Appliquez l'algorithme OPT sur l'exemple.

page	1	2	3	1	2	1	2	6	6	2	1	2	3	6	8	2	3	2	2	8	3	8	3	6	1
nouv.	1	2	3					6					3		8									6	1
idx C								2					0		2									0	0
Sup.								3					1		6									3	6
C0	1	1	1	1	1	1	1	1	1	1	1	1	3	3	3	3	3	3	3	3	3	3	3	6	1
C1		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
C2			3	3	3	3	3	6	6	6	6	6	6	6	8	8	8	8	8	8	8	8	8	8	8

2.I) [pt] En quoi consiste la méthode LRU de choix de défaut de page ?

Si la page est déjà dans le cache, on la remet en tête de file. Sinon on supprime la dernière page de la file et on place en tête la nouvelle page qui manquait.

2.J) [pt] Appliquez l'algorithme LRU sur l'exemple.

page	1	2	3	1	2	1	2	6	6	2	1	2	3	6	8	2	3	2	2	8	3	8	3	6	1
nouv.	1	2	3					6					3	6	8	2	3							6	1
idx C								2					2	0	1	2	0							2	1
Sup.								3					6	1	2	3	6							2	8
C0	1	1	1	1	1	1	1	1	1	1	1	1	1	6	6	6	3	3	3	3	3	3	3	3	3
C1		2	2	2	2	2	2	2	2	2	2	2	2	2	8	8	8	8	8	8	8	8	8	8	1
C2			3	3	3	3	3	6	6	6	6	6	3	3	3	2	2	2	2	2	2	2	2	6	6

Corrigé (Les réponses apparaissent avec ce fond de caractère)

2.K) [pt] En quoi consiste la méthode LFU de choix de défaut de page ?

Il faut un compteur à chaque page du cache. Si la page est dans le cache, on incrémente son compteur. Sinon, on choisit de supprimer le cadre correspondant au compteur le plus faible (il peut y en avoir plusieurs) et on met la page à la place en remettant le compteur à zéro.

2.L) [pt] Appliquez l'algorithme LFU sur l'exemple.

page	1	2	3	1	2	1	2	6	6	2	1	2	3	6	8	2	3	2	2	8	3	8	3	6	1
nouv.	1	2	3					6					3	6	8		3			8	3	8	3	6	
idx C								2					2	2	2		2			2	2	2	2	2	
Sup.								3					6	3	6		8			3	8	3	8	3	
F0	0	0	0	1	1	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	4
F1		0	0	0	1	1	2	2	2	3	3	4	4	4	4	5	5	6	7	7	7	7	7	7	7
F2			0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
C0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C1		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
C2			3	3	3	3	3	6	6	6	6	6	3	6	8	8	3	3	3	8	3	8	3	6	6

K) [pt] Dans les 3 algorithmes quel est le nombre de défaut de page ?

OPT : 8
LRU : 11
LFU : 13