Final Exam
Theoretical Portion
EEE243 Applied Computer Programming
14 December 2015, 1300 –1600 hrs

Examiner: Dr W. Greg Phillips, CD, PhD, PEng

Instructions:

- **Do not turn this page until instructed to do so.**
- This test has two portions. This is the *theoretical* portion, which is worth 45 marks out of 90. The *practical* portion is out of 45 marks.
- Fill out the front of the examination booklet, which will be used for the theoretical portion only.
- The theoretical portion of the test is closed-book. A copy of the ASCII table is attached at the end of the exam.
- You have 3 hours to complete both portions of the exam.
- Start with the *theoretical* portion and move to the *practical* portion when ready. You must hand in your *theoretical* exam booklet before beginning the *practical* portion of the exam.
- Questions have the values indicated in the centre column. Plan your time appropriately.
- If a question seems unclear, make a *reasonable* assumption, document it, and answer the question as though the assumption were correct. *The examiners will not clarify the meaning of questions during the exam.*
- Good luck!
- **Do not turn this page until instructed to do so.**

Examen Final
Partie Théorique
GEF243 Programmation informatique appliquée
14 décembre 2015, 13h00 –16h00

Examinateur: Capt Adrien Lapointe, CD, MSc

Instructions:

- **Ne tournez pas cette page avant l'instruction de l'examinateur.**
- Ce test a deux parties. Ceci est la partie *théorique* qui a une valeur de 45 points sur 90. La partie *pratique* a une valeur de 45 points.
- Remplissez la page couverture du livret d'examen qui sera utilisé uniquement pour la partie théorique de l'examen.
- La partie théorique de l'examen est à livre fermé. Une copie du tableau ASCII est jointe à la fin de l'examen.
- Vous avez 3 heures pour terminer les deux parties de l'examen.
- Commencez avec la partie *théorique* et continuez avec la partie *pratique* lorsque vous êtes prêts. Vous devez remettre votre livret d'examen *théorique* avant d'entamer la partie *pratique*.
- Les questions ont les valeurs indiquées dans la colonne centrale. Planifiez votre temps en conséquence.
- Si une question ne vous semble pas claire, faite une supposition raisonnable, documentez-la et répondez à la question en prenant la supposition en compte. *Les examinateurs ne clarifieront pas le sens des questions pendant l'examen.*
- Bonne chance!
- **Ne tournez pas cette page avant l'instruction de l'examinateur.**

1. List the three types of constants in C and explain the difference between them from the point of view of memory use.

3

1. Indiquez les trois types de constantes en C et expliquez la différence entre chaque type du point de vue de la mémoire.

2. Answer the following:

4

2. Répondez aux questions suivantes:

a. Draw the memory model for C programs as seen in class.

(2)

a. Dessinez le modèle de mémoire en C que nous avons vu en classe.

b. Give one example of what would be contained in each memory segment.

(2)

b. Donnez un exemple de ce qui serait dans chacun de ces segments de mémoire.

3. We want to create an application that will permit us to track marks for students in EEE243. The students in the course come from three different programs: Electrical Engineering (EE), Computer Engineering (CE) and Computer Science (CS). A student can be represented by a type-defined structure called `Student` that includes the student's first and last name, college number, program, and mark in the course. Student records will be added to the list once, at the beginning of the course. We will need to search for students frequently during the course, to update marks.

14

3. Nous voulons créer un logiciel qui permet de faire le suivi des notes des étudiants de GEF243. Il y a trois types d'étudiants dans le cours: ceux en génie électrique (`EE`), en génie informatique (`CE`) et en science informatique (`CS`). Un étudiant sera représenté par une structure de type définie appelée `Student` qui contient son nom, prénom, numéro de collège, programme et moyenne pour le cours. Les étudiants ne seront ajoutés à la liste qu'une fois au début de la session, mais nous devrons chercher les étudiants dans la liste régulièrement afin de mettre à jour leur note.

a. What data structure should we use to store the list? Why?

(3)

a. Quelle structure de données devriez-vous utiliser pour stocker la liste? Pourquoi?

b. Give the code defining an enumerated type called `Programmes` that we can use to represent the three programs.

(3)

b. Donnez le code pour une énumération appelée `Programmes` représentant les trois programmes.

c. Give the code defining the type-defined structure representing a student, using the enumeration defined in b.

(4)

c. Donnez le code pour la structure de type définie représentant un étudiant, en utilisant l'énumération définie en b.

d. Write a function `add_student` which takes as parameters the list of students, using your data structure from part a, and a student record, using your structure from part c. This function must add the student record to the list, must increment the total count of students, and must increment the count in the student's programme. The counts are named `num_students`, `num_EE`, `num_CE` and `num_CS`, and are declared as global variables. Your code must use a switch statement and the enumeration defined in b.

(4)

d. Écrivez la fonction `add_student` qui prend en paramètre la liste d'étudiants, représentée par la structure de données choisie en a, et un étudiant. En plus d'ajouter l'étudiant à la liste, cette fonction doit incrémenter un compteur pour le nombre total d'étudiants et un compteur pour le programme de l'étudiant. Les compteurs sont nommés `num_students`, `num_EE`, `num_CE` et `num_CS` et sont des entiers déclarés comme variables globales. Vous devez utiliser un énoncé `switch` et l'énumération préalablement définie en b.

4. Functions with the following signatures are provided to you.

2

4. Les fonctions avec les signatures suivantes vous sont fournies.

```
1  int addition(int a, int b);
2  int difference(int a, int b);
3  int multiplication(int a, int b);
4  int division(int a, int b);
```

Write the code to declare and define an array containing pointers to these four functions.

Donnez le code pour déclarer et définir un tableau contenant des pointeurs à ces fonctions.

5. What will the following program print? Explain your answer using a pointer diagram.

5

5. Qu'affichera le programme suivant? Expliquez votre réponse à l'aide de diagrammes de pointeurs.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void do_something_cool(char *p_str, char **p_p_char);
5
6  int main(void) {
7     char *str = "Hello";
8     char *p_str = str;
9     char *p_char = &str[2];
10
11    do_something_cool(p_str, &p_char);
12    printf("p_str: \t%s\n", p_str);
13    printf("p_char: %c\n", *p_char);
14    return 0;
15 }
16
17 void do_something_cool(char *p_str, char **p_p_char) {
18    char *str = "World!";
19    p_str = str;
20    *p_p_char += 2;
21 }
```

6. What are the four tools necessary to create and execute a C program?

2

6. Quels sont les quatre outils nécessaires pour créer et exécutez un programme C?

7. Draw the structure diagram for the following code snippet: **4**

7. Dessinez le graphe de structure pour l'extrait de code suivant:

```
 1 int main(void) {
 2    while (true) {
 3       wait_for_user();
 4       drive(MEDIUM_SLOW, TILE_WIDTH);
 5       turn(RIGHT, 90, SLOW);
 6       drive(MEDIUM_SLOW, 2 * TILE_WIDTH);
 7       turn(LEFT, 180, SLOW);
 8       drive(MEDIUM_SLOW, TILE_WIDTH);
 9    }
10 }
11
12 void wait_for_user() {
13    clear_display();
14    display("Press OK");
15    wait_for_button_down(OK_BUTTON);
16    wait_for_button_up(OK_BUTTON);
17    pause_ms(ALLOW_FINGER_REMOVAL);
18    clear_display();
19 }
```

8. In class we saw the concepts of scope (spatial) and extent (temporal). **4**

a. Explain these concepts. **(2)**

b. What are the scope and the extent of the variable `count` in the following code? **(1)**

c. What would the value of `count` be after three calls to the function? **(1)**

8. Nous avons vu en classe les concepts de portée (spatiale) et de durée (temporelle).

a. Expliquez ces concepts.

b. Quelles sont la portée et la durée de la variable `count` dans le code suivant?

c. Quelle serait la valeur de `count` après 3 appels à la fonction?

```
1 int do_something(void) {
2    static int count = 0;
3
4    // Do something
5    // Fait quelque chose
6    count++;
7    return count;
8 }
```

9. During the course, we saw two operators that can be used to access fields in structures. **3**

a. What are these two operators? **(1)**

b. What is the difference between the two operators? **(2)**

9. Pendant le cours, nous avons vu deux opérateurs pouvant être utilisés pour accéder aux champs des structures.

a. Quels sont ces opérateurs?

b. Quelle est la différence entre ces opérateurs?

10. The name of a function can appear in three different contexts (i.e., for three different purposes) in a C program. Give an example of a function name in each of these possible contexts.

3 10. Le nom d'une fonction peut être utilisé dans trois situations différentes (pour trois buts différents) dans un programme C. Donnez un exemple de nom de fonction dans chacune de ces situations.

11. What is the difference between calls to `malloc` and calls to `calloc`?

1 11. Quelle est la différence entre les appels `malloc` et `calloc`?

End of the **theoretical** part of the examination.

Fin de la partie **théorique** de l'examen.

Final Exam
Setup Instructions for Practical Portion
EEE243 Applied Computer Programming
14 December 2015

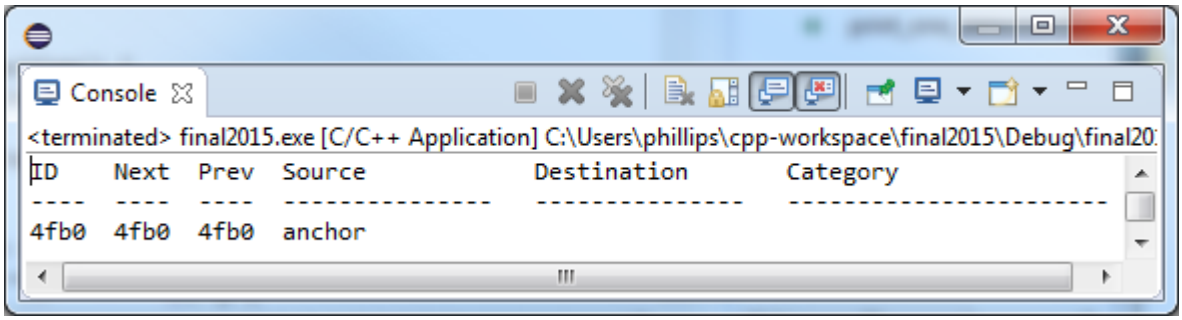
Examiner: W. Greg Phillips, CD, PhD, PEng

- **Begin immediately.**
  - **Stand up** if you run into problems.
- Login to the following temporary account:
  - Username: `ECE\ece-examuser`
  - Password: `EcE&2345`
- Launch Eclipse
  - Accept the default location for your workspace then click the *Workbench* arrow at the upper right of the Eclipse window.
- Open a browser and download the file located at `http://last3.in/udkwo.zip`
  Do not unzip the file.
- Import the project by choosing…

File → Import → General → Existing Projects into Workspace → Next → Select archive file → Browse

- … then navigating to the `udkwo.zip` file and click ***Open*** and ***Finish***.
- Right-click on the project name, choose ***Rename***, and change the project name to your family name.
- Open the `main.c` file and add your name to the first line.

- Run the program:
  - compile the ***Debug*** configuration; and
  - execute the ***Debug*** configuration.
- You should see an output similar to what is shown below in the Console



- Configure your workspace by selecting

Windows → Preferences → General → Editors → Text Editors → Show line numbers → Apply
Windows → Preferences → General → Workspace → Save automatically before build → Apply

- Do not log off the computer, but turn off both monitors.

- Put everything other than pens, pencils, etc away.
- Ask the invigilator for a copy of the theory portion of the exam. **Do not begin the theory portion until instructed.**

Examen final
Instructions de mise en place pour la partie pratique
GEF243 Programmation informatique appliquée
14 décembre 2015

Examinateur: Capt Adrien Lapointe, CD, MSc

- **Commencez tout de suite.**
  - **Levez-vous** si vous avez des problèmes.
- Ouvrez une session avec le compte temporaire suivant :
  - Nom d'utilisateur : `ECE\ece-examuser`
  - Mot de passe : `EcE&2345`
- Lancez Eclipse
  - Acceptez l'emplacement du workspace par défaut puis cliquez sur la flèche *Workbench* en haut à droite de la fenêtre Eclipse.
- Ouvrez un navigateur et téléchargez le fichier situé au `http://last3.in/udkwo.zip`
  Ne décompressez pas le fichier.
- Importez le projet avec…

- … et naviguez jusqu'au fichier `udkwo.zip` et cliquez ***Open*** et ***Finish***.
- Cliquez-droit sur le nom de projet, choisissez ***Rename***, et modifiez le nom de projet à votre nom de famille.
- Ouvrez le fichier `main.c` et ajoutez votre nom à la première ligne.

- Exécutez le programme:
  - compilez la version ***Debug***; et
  - exécutez la version ***Debug***.
- Vous devriez obtenir un résultat semblable à l'image ci-dessous dans la Console

- Configurez votre *workspace* en choisissant

- Ne vous déconnectez pas de l'ordinateur, mais éteignez les deux écrans.
- Rangez vos effets personnels sauf les stylos, crayons, etc.
- Demandez au surveillant une copie de la partie théorique de l'examen. **Ne commencez pas la partie théorique avant d'en recevoir l'instruction.**

Final Examination
Practical Portion
EEE243 Applied Computer Programming
14 December 2015

Examiner: W. Greg Phillips, CD, PhD, PEng

Instructions

- You may begin the practical portion of the exam immediately.
- The practical portion is out of 45 marks.
- The practical portion of this examination is open book. You may refer to course notes and you may access the Internet.
- Communication with any person other than the invigilators during the practical portion of the test **is forbidden**.
- You will not receive an exam booklet for the practical portion of the test; you have received some notepaper instead.
- If you wish your notepaper to be marked by the examiner, write your name and student number on it and turn it in at the end of the exam.
- Ask the invigilator for more notepaper if you require it.

Hints: Compile and execute your program **often**. You will lose marks if your program does not compile and execute as submitted, or if it generates compiler warnings. Test your code as you go. Start small and build from there.

Draw memory diagrams of all pointer manipulations before writing the code!

Examen Final
Partie Pratique
GEF243 Programmation informatique appliquée
14 décembre 2015

Examinateur: Capt Adrien Lapointe, CD, MSc

Instructions

- Vous pouvez commencer la partie pratique de l'examen tout de suite.
- La valeur de la partie pratique est de 45 points.
- La partie pratique de l'examen est à livre ouvert. Vous pouvez faire référence aux notes du cours et vous pouvez accéder à l'Internet.
- La communication avec toutes personnes autre que les surveillants pendant la partie pratique de l'examen **est interdite**.
- Vous ne recevrez pas de livret d'examen pour la partie pratique du test; vous avez plutôt reçu des pages de papier de notes.
- Si vous voulez que les pages de notes soient corrigées par l'examinateur, écrivez-y votre nom et numéro de collège et remettez-les à la fin de l'examen.
- Nous avons amplement de papier de notes, demandez-en plus si vous en avez besoins.

Indices : Compilez et exécutez votre code **souvent**. Vous serez pénalisé si votre programme ne compile et n'exécute pas ou s'il produit des avertissements. Testez votre code souvent. Commencez petit et ajoutez des fonctionnalités au fur et à mesure.

Dessinez des diagrammes de mémoire pour toutes les manipulations de pointeurs avant de coder.

Introduction

Open the project that you created at the beginning of the examination.

If you haven't already done so, **add your name** to the comment block at the beginning of the `main.c` file.

You must provide evidence of your design for each part of the tasks. Clear, well-structured code that works and includes appropriate comments is sufficient. However, if your code does not work or is not clear or well structured, you may receive part marks for design documentation in the form of flowcharts, pseudo-code, pointer or memory diagrams, or textual descriptions of your design. Write these on the provided notepaper.

You have been provided with three files: `main.c`, `ids.c` and `ids.h`. You are **not permitted** to make any changes to `ids.c` and `ids.h`.

Overview

The code provided is a partial simulation of a logging system for a network intrusion detection system. As network intrusions are detected, they are reported via the `get_next_detection` function. Your job is to write code that maintains the log of detections.

Introduction

Ouvrez le projet que vous avez créé au début de l'examen.

Si vous ne l'avez pas encore fait, **ajoutez votre nom** au bloc commentaire au début du fichier `main.c`.

Vous devez fournir des traces de votre conception pour chaque partie des tâches. Du code clair et bien structuré qui fonctionne et qui inclus des commentaires appropriés est suffisant. Toutefois, si votre code ne fonctionne pas ou s'il n'est pas clair ou bien structuré, vous pourrez recevoir des points partiels pour la documentation de conception sous la forme d'organigrammes, de pseudo-code, de schémas de la mémoire ou des pointeurs, ou de descriptions textuelles de votre conception. Écrivez-les sur le papier de notes fournit.

Nous vous fournissons trois fichiers : `main.c`, `ids.c` et `ids.h`. Vous **ne devez pas** modifier les fichiers `ids.c` et `ids.h`.

Vue d'ensemble

Le code fournit est une simulation partielle d'un système de journal pour un détecteur d'intrusions réseau. À mesure que les intrusions de réseau sont détectées, elles sont rapportées par la fonction `get_next_detection`. Votre tâche est d'écrire le code qui maintient le journal des détections.

Log representation

The log must be stored in a circular, doubly linked list with an anchor node. The anchor node is not part of the log; it serves only to mark the start and end of the circular list structure.

Each log entry is an instance of the `Detection` structure that is defined in `ids.h`. The anchor node is also an instance of `Detection`.

The required structure of the log is illustrated below.

On the left is an empty log, which includes only the anchor node (shown in white). This is the log's structure after the call to `create_anchor` in the provided `main`.

In the middle is the log with one entry (shown in grey).

On the right is the log with three entries.

Note the circular, bi-directional list structure, which is maintained using the `next` and `prev` pointers in the `Detection` structure.

New entries are added at the beginning of the log, which is indicated by the anchor node's `next` pointer. The anchor node's `prev` pointer indicates the end of the log (the oldest entry).

Représentation du journal

Le journal doit être stocké dans une liste doublement chaînée circulaire avec un nœud d'ancrage. Le nœud d'ancrage ne fait pas parti du journal. Il sert à indiquer le début et la fin de la structure circulaire de la liste.

Chaque entrée du journal est une instance de la structure `Detection` qui est définie dans `ids.h`. Le nœud d'ancrage est aussi une instance de `Detection`.

La structure demandée pour le journal est illustrée ci-dessous.

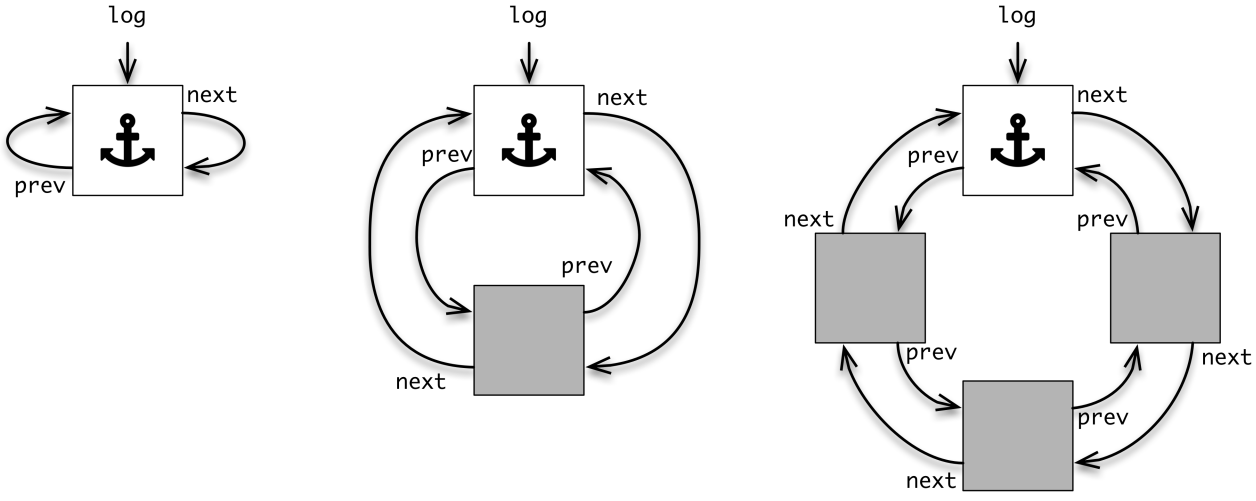
Un journal vide est montré à gauche, il n'inclus que le nœud d'ancrage (en blanc). C'est la structure du journal après un appel à `create_anchor` donné dans la `main`.

Au milieu le journal est montré avec une entrée (en gris).

À droite le journal est montré avec trois entrées.

Notez la structure circulaire et bi-directionnelle de la liste qui est maintenue grâce aux pointeurs `next` et `prev` dans la structure `Detection`.

Les nouvelles entrées sont ajoutées au début du journal qui est indiqué par le pointeur `next` du nœud d'ancrage. Le pointeur `prev` du nœud d'ancrage indique la fin du journal (entrée la plus ancienne).



You must complete the following tasks. We suggest you complete them in the order listed. As you complete each task, write test code in your `main` function to ensure your implementations work as intended. You may comment out this test code and leave it in the `main` function when it is no longer required.

Note that the last task is to provide particular code in your `main`.

Vous devez compléter les tâches suivantes. Nous vous recommandons de les faire dans l'ordre où elles apparaissent. Écrivez du code de test dans votre fonction `main` à mesure que vous compléter les tâches pour vous assurer que l'implémentation fonctionne. Vous pouvez mettre ce code de test en commentaires et le laisser dans la `main` lorsqu'il n'est plus nécessaire.

Notez que la dernière tâche indique le code particulier à mettre dans votre `main`.

Task 1: Add entries to the log **[10]**

Implement the `add_to` function so that it adds a new log entry (the parameter `new`) to the log whose anchor node is the parameter `anchor`. New entries are added at the beginning of the log; i.e., they are attached to the anchor node's `next` pointer. Your code must correctly maintain the doubly linked structure illustrated on the previous page.

You can generate a new log entry by calling the `get_next_detection` function. See also task 2, which provides a hint that will be useful for testing.

Tâche 1 : Ajouter des entrées au journal **[10]**

Implémentez la fonction `add_to` afin qu'elle ajoute une nouvelle entrée (le paramètre `new`) au journal dont le nœud d'ancrage est le paramètre `anchor`. Les nouvelles entrées sont ajoutées au début du journal : elles sont rattachées au pointeur `next` du nœud d'ancrage. Votre code doit maintenir correctement la structure doublement chaînée de la liste tel qu'illustré précédemment.

Vous pouvez générer une nouvelle entrée pour le journal en appelant `get_next_detection`. La tâche 2 fournit un indice utile pour vos tests.

Task 2: Print the log **[5]**

Complete the implementation of the `print_entire` function so that it prints all entries in the log in order from newest to oldest. You can use the provided `print_one_detection` function to print an individual log entry.

Tâche 2 : Afficher le journal **[5]**

Finissez l'implémentation de la fonction `print_entire` afin qu'elle affiche toutes les entrées dans le journal en ordre de la plus récente à la plus ancienne. Vous pouvez utiliser la fonction `print_one_detection` pour afficher une seule entrée du journal.

Task 3: Count log entries **[5]**

Implement the `count_entries` function so that it returns the number of entries currently in the log. The count does not include the anchor node. Your implementation must compute the number of entries by walking the log, counting each entry found.

Tâche 3 : Compter les entrées de journal **[5]**

Implémentez la fonction `count_entries` afin qu'elle retourne le nombre d'entrées dans le journal en ce moment. Le compte ne doit pas inclure le nœud d'ancrage. Votre implémentation doit calculer le nombre d'entrées en parcourant le journal et comptant chaque entrée trouvée.

Task 4: Remove oldest log entry **[10]**

Implement the `remove_oldest` function so that it removes the oldest log entry from the log and returns its memory to the operating system. Your function must correctly maintain the circular structure of the remaining log entries.

Tâche 4: Retirer l'entrée la plus ancienne **[10]**

Implémentez la fonction `remove_oldest` afin qu'elle retire la plus ancienne entrée du journal et retourne sa mémoire au système d'exploitation. Votre fonction doit maintenir correctement la structure circulaire des entrées restantes.

Task 5: Remove all entries

Implement the `erase_all` function so that it removes all entries from the list, returning their memory to the operating system. After removing all entries, the log structure should be as shown in the left-hand diagram on page 3.

Task 6: Enforce maximum log size

Our intrusion detection system is to be implemented as an embedded system with very little available memory. Modify your `add_to` function so that the log always contains the most recent entries and never contains more than `MAX_LOG_LENGTH` entries (defined in `ids.h`). If the log already contains `MAX_LOG_LENGTH` entries, remove the oldest node to make room for the new node to be inserted.

Task 7: main

Comment out any test code remaining in your `main` function and add code to do the following, using the functions you have implemented:

- Initialize the log.
- Generate 20 log entries, adding each one to the log. If you have completed task 6, the log should never grow beyond `MAX_LOG_LENGTH` entries. Print the entire log after each entry is added.
- Remove all entries from the log.
- Print the entire (now empty) log.
- Exit cleanly.

When you are finished
Ensure your name is in the header comment of the main file and on your notes pages. Leave the computer logged in and your project open in Eclipse, and leave your notes pages on the keyboard. Leave quietly by the nearest door.

End of the examination

[5] Tâche 5: Retirer toutes les entrées

Implémentez la fonction `erase_all` afin qu'elle retire toutes les entrées de la liste et retourne leur mémoire au système d'exploitation. Après avoir retiré toutes les entrées, la structure du journal devrait être comme illustrée par le diagramme de gauche à la page 3.

[5] Tâche 6: Faire respecter la taille maximale

Notre système de détection des intrusions sera implémenté en tant que système embarqué avec très peu de mémoire disponible. Modifiez votre fonction `add_to` afin que le journal contienne toujours les entrées les plus récente et jamais plus que `MAX_LOG_LENGTH` entrées (définie dans `ids.h`). Si le journal contient déjà `MAX_LOG_LENGTH` entrées, retirez le nœud le plus ancien pour faire place au nouveau nœud à insérer.

[5] Tâche 7: main

Mettez en commentaire tout code de test restant dans votre fonction `main` et ajoutez le code faisant ce qui suit en utilisant les fonctions que vous avez implémentés :

- Initialisez le journal.
- Générez 20 entrées et ajoutez-les au journal. Si vous avez compléter la tâche 6, le journal ne devrait jamais grossir au-delà de `MAX_LOG_LENGTH` entrées. Affichez le journal après que chaque entrée soit ajoutée.
- Retirez toutes les entrées du journal.
- Affichez le journal (maintenant vide) au complet.
- Sortez proprement.

Lorsque vous avez terminé
Assurez-vous que votre nom est dans le bloc commentaire des fichiers et sur vos pages de notes. Laissez l'ordinateur connecté et votre projet ouvert dans Eclipse. Laissez aussi vos pages de notes sur le clavier. Partez discrètement par la porte la plus proche.

Fin de l'examen