

Final Exam Theoretical Portion

EEE243 Applied Computer Programming
10 December 2014, 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 95. The *practical* portion is out of 50 marks with a 10 mark bonus question.
- 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
10 décembre 2014, 13h00–16h00

Examineur: Capt Adrien Lapointe, CD, MSc

Instructions:

- **Ne tournez pas cette page avant l'instruction de l'examineur.**
- Ce test a deux parties. Ceci est la partie *théorique* qui a une valeur de 45 points sur 95. La partie *pratique* a une valeur de 50 points avec une question boni qui vaut 10 points.
- Remplissez l'en-tête 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 bien votre temps.
- Si une question ne vous semble pas claire, faite une supposition raisonnable, documentez-la, et répondez à la question en prenant compte de la supposition. *Les examinateurs ne clarifieront pas le sens des questions pendant l'examen.*
- Bonne chance!
- **Ne tournez pas cette page avant l'instruction de l'examineur.**

- | | |
|--|--|
| <p>1. Why is it better not to use literal constants in code (e.g. <code>c = 2*3.1416*r;</code>)? Suggest a better solution.</p> <p>2. Explain what functional decomposition is and why it is useful in programming.</p> <p>3. We have seen in class that it is possible to pass parameters <i>by value</i> and <i>by reference</i>. For each of these two methods, give:</p> <ul style="list-style-type: none"> a. A definition. b. An example of its use (in code). c. A situation in which it should be used. <p>4. Give the signature for a <code>register_robot</code> function that takes a pointer to a function as a parameter and returns nothing. The function passed as a parameter returns a pointer to a <code>char</code> and takes a pointer to a <code>char</code> and an <code>int</code> as parameters.</p> <p>5. What are the values of <code>a</code>, <code>b</code>, <code>c</code>, and <code>d</code> after the call to the <code>mystery_fctn()</code> function in <code>main()</code>?
Hint: You may find it useful to use diagrams.</p> | <p>4 1. Pourquoi est-il préférable de ne pas utiliser des constantes littérales dans le code (ex. : <code>c = 2*3.1416*r;</code>)? Suggérez une meilleure solution.</p> <p>4 2. Expliquez ce qu'est la décomposition fonctionnelle et pourquoi elle est utile en programmation.</p> <p>6 3. Nous avons vu en classe qu'il est possible de passer des paramètres <i>par valeur</i> et <i>par référence</i>. Pour chacune de ces méthodes donnez :</p> <ul style="list-style-type: none"> a. Une définition. b. Un exemple de son utilisation (en code). c. Une situation où elle devrait être utilisée. <p>5 4. Donnez la signature d'une fonction <code>register_robot</code> acceptant un pointeur à une fonction en paramètre et ne retournant rien. La fonction en paramètre retourne un pointeur à un <code>char</code> et accepte un pointeur à <code>char</code> et un <code>int</code> en paramètres.</p> <p>4 5. Quelles sont les valeurs de <code>a</code>, <code>b</code>, <code>c</code> et <code>d</code> après l'appel à la fonction <code>mystery_fctn()</code> à partir de <code>main()</code>? Indice : Utilisez des dessins.</p> |
|--|--|

```

1  int mystery_fctn(int* a, int* y, int z) {
2      int b = 0;
3      if (*a == b) {
4          *a += 9;
5      } else {
6          *a -= 9;
7          y = a;
8      }
9      return (a - y + z);
10 }
11
12 int main(void) {
13     int a = 10, b = 10, c = 12, d = 24;
14     b = mystery_fctn(&a, &c, d);
15     return 0;
16 }

```

6. Implement a `capitalize_characters()` function that satisfies the interface below.

```
void capitalize_characters(char* string)
```

The function must take a string such as "Hello World!" and capitalize all the letters so it becomes "HELLO WORLD!". You must implement this function with a *for* loop. An ASCII table is attached to the exam.

7. You are currently working on the following function and want to make it easier to read and to maintain. You know that using explicit integers for status values is a bad idea. You also expect new status values to be added soon. Rewrite the function to address these goals, adding any code necessary.

- 6 6. Implémentez la fonction `capitalize_characters()` qui satisfait l'interface ci-dessous.

La fonction doit prendre une chaîne de caractères tel que "Bonjour le monde!" et mettre toutes les lettres en majuscules, la transformant en "BONJOUR LE MONDE!". Votre implémentation doit utiliser une boucle *for*. Un tableau ASCII est joint à l'examen.

- 4 7. Vous travaillez présentement sur la fonction suivante et vous voulez la rendre plus facile à lire et à maintenir. Vous savez que l'usage d'entiers explicites n'est pas une bonne idée. Vous savez aussi que de nouveaux états seront ajoutés sous peu. Réécrivez la fonction pour tenir compte de ces buts. Ajoutez le code nécessaire.

```
1 void check_battery_status(int battery_status) {
2     switch (battery_status) {
3         case 0:
4             printf("Unknown / inconnu\n");
5             break;
6         case 1:
7             printf("Charging / en chargement\n");
8             break;
9         default:
10            printf("Error / erreur \n");
11        }
12    }
13 }
```

8. Consider the following C program:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 typedef struct {
6     char first_name[15];
7     char last_name[15];
8     int number;
9 } Student;
10
11 void some_fcn(Student* x, char* a, char* b);
12
13 int some_var = 0;
14
15 int main(void) {
16     char prenom[] = "Joe";
17     char nom[] = "Bloggins";
18     Student* head = (Student*) malloc(sizeof(Student));
19     some_fcn(head, prenom, nom);
20     return 0;
21 }
22
23 void some_fcn(Student* x, char* a, char* b) {
24     static int count = 0;
25     strcmp((*x).first_name, a);
26     strcmp((*x).first_name, a);
27     (*x).number = count++;
28 }
```

9 8. Considérez le programme C suivant :

- a. Draw the memory model for C programs that we have seen in class. (2)
- b. Referring to that model, in what regions of memory are some_var, prenom, head, count and some_fcn located? (2)
- c. Where is the memory space allocated when using malloc? When is that space freed? (2)
- d. What are the scope and the temporal extent of the variables some_var, prenom, head, and count? (3)

- a. Dessinez le modèle de mémoire en C que nous avons vu en classe.
- b. En vous référant au modèle, dans quelles régions de mémoire se situe some_var, prenom, head, count et some_fcn?
- c. Où est alloué l'espace mémoire lorsque malloc est utilisé? Quand est cet espace libéré?
- d. Quelle sont la portée et la durée des variables some_var, prenom, head, et count?

9. In the context of software engineering, what is information hiding?

3 9. Dans la contexte du génie logiciel, qu'est-ce que le masquage d'information?

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

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

Decimal - Binary - Octal - Hex – ASCII Conversion Chart

Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII
0	00000000	000	00	NUL	32	00100000	040	20	SP	64	01000000	100	40	@	96	01100000	140	60	`
1	00000001	001	01	SOH	33	00100001	041	21	!	65	01000001	101	41	A	97	01100001	141	61	a
2	00000010	002	02	STX	34	00100010	042	22	"	66	01000010	102	42	B	98	01100010	142	62	b
3	00000011	003	03	ETX	35	00100011	043	23	#	67	01000011	103	43	C	99	01100011	143	63	c
4	00000100	004	04	EOT	36	00100100	044	24	\$	68	01000100	104	44	D	100	01100100	144	64	d
5	00000101	005	05	ENQ	37	00100101	045	25	%	69	01000101	105	45	E	101	01100101	145	65	e
6	00000110	006	06	ACK	38	00100110	046	26	&	70	01000110	106	46	F	102	01100110	146	66	f
7	00000111	007	07	BEL	39	00100111	047	27	'	71	01000111	107	47	G	103	01100111	147	67	g
8	00001000	010	08	BS	40	00101000	050	28	(72	01001000	110	48	H	104	01101000	150	68	h
9	00001001	011	09	HT	41	00101001	051	29)	73	01001001	111	49	I	105	01101001	151	69	i
10	00001010	012	0A	LF	42	00101010	052	2A	*	74	01001010	112	4A	J	106	01101010	152	6A	j
11	00001011	013	0B	VT	43	00101011	053	2B	+	75	01001011	113	4B	K	107	01101011	153	6B	k
12	00001100	014	0C	FF	44	00101100	054	2C	,	76	01001100	114	4C	L	108	01101100	154	6C	l
13	00001101	015	0D	CR	45	00101101	055	2D	-	77	01001101	115	4D	M	109	01101101	155	6D	m
14	00001110	016	0E	SO	46	00101110	056	2E	.	78	01001110	116	4E	N	110	01101110	156	6E	n
15	00001111	017	0F	SI	47	00101111	057	2F	/	79	01001111	117	4F	O	111	01101111	157	6F	o
16	00010000	020	10	DLE	48	00110000	060	30	0	80	01010000	120	50	P	112	01110000	160	70	p
17	00010001	021	11	DC1	49	00110001	061	31	1	81	01010001	121	51	Q	113	01110001	161	71	q
18	00010010	022	12	DC2	50	00110010	062	32	2	82	01010010	122	52	R	114	01110010	162	72	r
19	00010011	023	13	DC3	51	00110011	063	33	3	83	01010011	123	53	S	115	01110011	163	73	s
20	00010100	024	14	DC4	52	00110100	064	34	4	84	01010100	124	54	T	116	01110100	164	74	t
21	00010101	025	15	NAK	53	00110101	065	35	5	85	01010101	125	55	U	117	01110101	165	75	u
22	00010110	026	16	SYN	54	00110110	066	36	6	86	01010110	126	56	V	118	01110110	166	76	v
23	00010111	027	17	ETB	55	00110111	067	37	7	87	01010111	127	57	W	119	01110111	167	77	w
24	00011000	030	18	CAN	56	00111000	070	38	8	88	01011000	130	58	X	120	01111000	170	78	x
25	00011001	031	19	EM	57	00111001	071	39	9	89	01011001	131	59	Y	121	01111001	171	79	y
26	00011010	032	1A	SUB	58	00111010	072	3A	:	90	01011010	132	5A	Z	122	01111010	172	7A	z
27	00011011	033	1B	ESC	59	00111011	073	3B	;	91	01011011	133	5B	[123	01111011	173	7B	{
28	00011100	034	1C	FS	60	00111100	074	3C	<	92	01011100	134	5C	\	124	01111100	174	7C	
29	00011101	035	1D	GS	61	00111101	075	3D	=	93	01011101	135	5D]	125	01111101	175	7D	}
30	00011110	036	1E	RS	62	00111110	076	3E	>	94	01011110	136	5E	^	126	01111110	176	7E	~
31	00011111	037	1F	US	63	00111111	077	3F	?	95	01011111	137	5F	_	127	01111111	177	7F	DEL

Setup Instructions

Final Examination: Practical Portion

EEE243 Applied Computer Programming


10 December 2014

Examiner: W. Greg Phillips, PhD, PEng

1. Fill out your place card. Make sure you put the correct computer number on it.

Eclipse project setup

2. Log in using the user name
ECE\ece-examuser and the password
ECE-4321.
3. Download the file from
<http://last3.in/gnwcg.zip>
4. Launch Eclipse and accept the default workspace location.
5. Create a new C (MinGW GCC) project using your family name for the project name.
6. Import the file you downloaded in step 3.
7. Compile and run the Debug configuration. At the top of the console window, you should see something similar to what's shown below.



```
<terminated> final2014practical [C/C++ Application] /Users/phillips/Documents/courses/243/eclipse/final2014practical/Debug/final2014practical (12/8/14, 3:35 PM)
Routing starting from head, no priority
-----
Queue before routing:

head: b30
tail: null

-----
from  to  pri  this  next  prev
-----
103   243   1    b30   b10   null
188   07    2    b10   b30   null
```

8. Add your name at line 1 of main.c
9. Configure your workspace by selecting:

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

Last steps

10. Do not log off the computer, but turn off both monitors.
11. Put everything other than pens, pencils, etc away.
12. Ask the invigilator for a copy of the theory portion of the exam. **Do not begin the theory portion until instructed.**

Instructions de mise en place

Examen final: partie pratique

GEF243 Programmation informatique appliquée

10 décembre 2014

Examineur: Capt Adrien Lapointe, MSc

1. Remplissez votre carte de place. Assurez-vous que vous avez indiqué le bon numéro d'ordinateur.

Mise en place du projet Eclipse

2. Connectez-vous avec le nom d'utilisateur
ECE\ece-examuser et le mot de passe
ECE-4321.
3. Téléchargez le fichier situé à
<http://last3.in/gnwcg.zip>
4. Lancez Eclipse et acceptez l'emplacement du
« workspace » offert par défaut.
5. Créez un nouveau projet C (MinGW GCC) avec
votre nom de famille comme nom de projet.
6. Importez le fichier téléchargé dans l'étape 3.
7. Compilez et exécutez la version *Debug*. En haut de la fenêtre de « console », vous devriez voir un résultat semblable à celui montré ci-dessous.

8. Ajoutez votre nom à la ligne 1 de main.c
9. Configurez votre workspace en choisissant:

Dernières étapes

10. Ne vous déconnectez pas de l'ordinateur, mais éteignez les deux écrans.
11. Rangez vos effets personnels sauf les stylos, crayons, etc.
12. 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
10 December 2014, 1300–1600 hrs

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

Instructions

- You may begin the practical portion of the exam immediately.
- The practical portion is out of 50 marks with a bonus question worth 10 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 three pages of 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. Start small and build from there.

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

You may find the `show_queue()` function in `router.c` useful for debugging.

Examen Final

Partie Pratique

GEF243 Programmation informatique appliquée
23 octobre 2014, 13h00–16h00

Examineur: Capt Adrien Lapointe, CD, MSc

Instructions

- Vous pouvez commencer la partie pratique tout de suite.
- La valeur de la partie pratique est de 50 points, avec une question boni qui vaut 10 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 trois pages de papier de notes.
- Si vous voulez que les pages de notes soient corrigées par l'examineur, 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. 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.

La fonction `show_queue()` située dans `router.c` vous sera probablement utile pour déboguer.

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.

The code provided is a partial simulation of an Internet router. It is intended to have two modes of operation: normal mode, where it sends each packet out in order received, and priority mode, where it sends all waiting priority 1 packets before sending any priority 2 packets.

An inefficient implementation of the normal mode has been provided. Your job is to write an efficient implementation of the normal mode, plus an implementation of the priority mode.

There is also a bonus task, which requires you to write a more efficient implementation of the priority mode.

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, memory diagrams, or textual descriptions of your design. Write these on the provided notepaper.

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`.

Le code fourni est une simulation partielle d'un routeur pour l'Internet. Il est supposé avoir deux modes d'opération : le mode normal, où les paquets sont envoyés dans l'ordre de réception, et le mode prioritaire, où les paquets de priorité 1 en attente sont envoyés avant ceux de priorité 2.

Une implémentation inefficace du mode normal vous est fournie. Votre tâche est d'écrire une implémentation efficace de ce mode ainsi qu'une implémentation du mode prioritaire.

La tâche boni consiste à écrire une implémentation plus efficace du mode prioritaire.

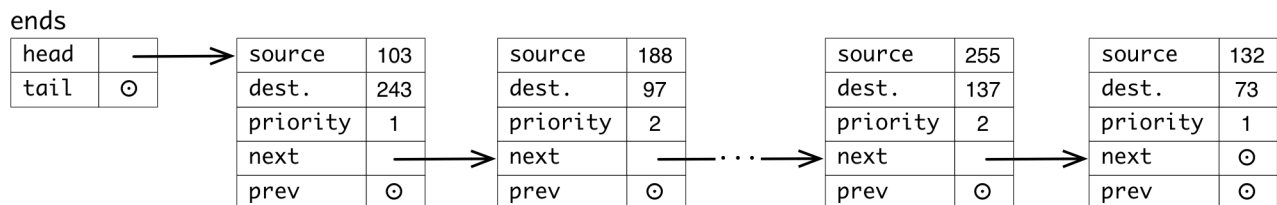
Vous devez fournir des traces de votre conception pour chaque partie des tâches. Du code clair et bien structuré qui fonctionne et qui inclut des commentaires appropriés est suffisant. Toutefois, si votre code ne fonctionne pas ou s'il n'est pas clair ou bien structuré, vous pourrez recevoir des points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

You have been provided with three files: `main.c`, `router.c` and `router.h`. You are **not permitted** to make any changes to `router.c` and `router.h`, except when completing the bonus task.

Initial routing queue representation

Packets in the routing queue are represented by structures of type `Packet`, joined in a linked list. When a packet arrives it is added to the head end of the list. In normal mode, packets are sent in order from oldest (farthest from the head) to newest.

The simulation begins by adding several packets to the queue. The initial structure of the queue is as shown below, where \odot indicates a null pointer and the ... indicates omitted packets.



Initial packet routing algorithm

After creating this structure, simulation then sends all the packets by repeatedly calling the implementation of normal mode in `starting_from_head_send_oldest_packet()`. This function finds the oldest packet, removes it from the list, sends it, and frees its memory.

This implementation is inefficient since it starts from the head and traverses the list to its end in order to find the oldest packet. This is not a big deal when the routing queue is small, but would be a significant problem if the routing queue contained many packets. The implementation would be more efficient if we simply had a reference to the oldest packet.

Nous vous fournissons trois fichiers : `main.c`, `router.c` et `router.h`. Vous **ne devez pas** modifier les fichiers `router.c` et `router.h`, sauf quand vous ferrez la tâche boni.

Implémentation initiale de la file de routage

Les paquets dans la file routage sont représentés par une structure de type `Packet` et sont liés entre eux dans une liste chaînée. Lorsqu'un paquet est reçu, il est ajouté au début de la liste. En mode normal, les paquets sont envoyés en ordre du plus ancien (le plus loin du début) au plus récent.

La simulation commence en ajoutant plusieurs paquets à la file. La structure initiale de la file est illustrée ci-dessous. Le \odot indique un pointeur nul et les ... indiquent des paquets non-illustrés.

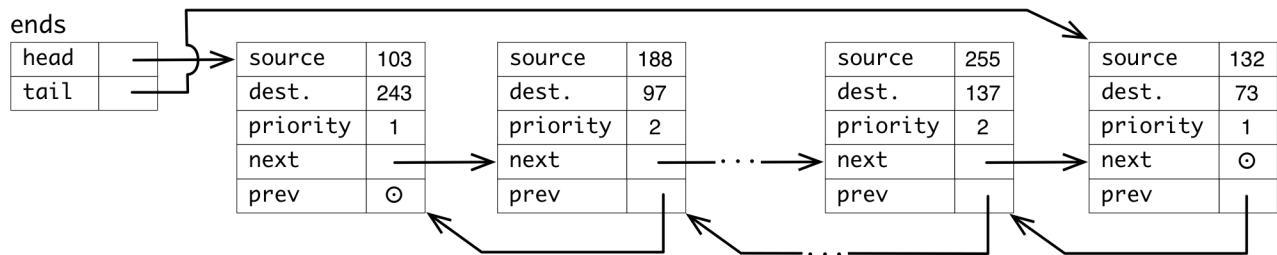
Algorithme initial de routage des paquets

Après avoir créé cette structure, la simulation transmet tous les paquets en appelant plusieurs fois la fonction du mode normal `starting_from_head_send_oldest_packet()`. Cette fonction trouve le paquet le plus ancien, l'enlève de la file, le transmet et libère sa mémoire.

Cette implémentation est inefficace puisqu'elle commence toujours au début de la file et la traverse au complet afin de trouver le paquet le plus ancien qui est situé à la fin. Cela ne pose pas problème lorsque la file est courte, mais serait un problème majeur si la file était plus longue. L'implémentation serait plus efficace si nous avions simplement une référence au paquet le plus ancien.

Task 1: Doubly-linked list

Modify `enqueue_packet()` so that the `tail` field of the `ends` structure always points to the end of the list, and the `prev` field of each `Packet` points to the previous element in the list, as illustrated below. This is a doubly linked list. Note that the `prev` field of the first element in the list is null, as is the `next` field of the last element in the list.



Task 2: Efficient normal routing

Implement the function `starting_from_tail_send_oldest_packet()` so that it finds the oldest packet in the list using the tail pointer, removes it from the list, sends it, and frees its allocated memory.

Once the function has been implemented, you can test it by uncommenting lines 69–71 in `main.c`.

Task 3: Priority routing

Implement the function `in_priority_starting_from_tail_send_oldest_packet()`. Like the function in task 2, this function finds the packet to send starting from the tail. However, if there are any priority 1 packets in the list, it removes, sends and frees the oldest priority 1 packet. If there are no priority 1 packets in the list, it removes, sends and frees the oldest packet in the list, which will be a priority 2 packet. Your implementation will need to traverse the list from the tail towards the head, to find the oldest priority 1 packet.

Once the function has been implemented, you can test it by uncommenting lines 82–84 in `main.c`.

[10] Tâche 1 : Liste chaînée double

Modifiez `enqueue_packet()` pour que le champ `tail` de la structure `ends` pointe toujours à la fin de la liste et que le champ `prev` de chaque `Packet` pointe à l'élément précédent de la liste tel qu'illustré ci-dessous. Il s'agit d'une liste chaînée double. Notez que le champ `prev` du premier élément de la liste est nul, tout comme le champ `next` du dernier.

[15] Tâche 2 : Routage normal efficace

Implémentez la fonction `starting_from_tail_send_oldest_packet()` de façon à ce qu'elle trouve le paquet le plus ancien dans la liste en utilisant le pointeur de fin, l'enlève de la liste, le transmette et libère sa mémoire.

Lorsque la fonction sera implémentée, vous pouvez la tester en activant le code aux lignes 69–71 dans `main.c`.

[25] Tâche 3 : Routage prioritaire

Implémentez la fonction `in_priority_starting_from_tail_send_oldest_packet()`. Tout comme la fonction de la tâche 2, celle-ci trouve le paquet à envoyer à partir de la fin. Toutefois, s'il y a des paquets de priorité 1 dans la liste, elle enlève, transmet et libère la mémoire du plus ancien paquet de priorité 1. S'il n'y a pas de paquets de priorité 1 dans la liste, la fonction enlève, transmet et libère le plus ancien paquet dans la liste, qui en sera nécessairement de priorité 2. Votre implémentation doit traverser la liste de la fin au début pour trouver le plus ancien paquet de priorité 1.

Lorsque la fonction sera implémentée, vous pouvez la tester en activant le code aux lignes 82–84 dans `main.c`.

Bonus task setup

IMPORTANT! Do not attempt the bonus task in the Eclipse project that contains your solution to tasks 1, 2 and 3. Create a new project as follows and do all work on the bonus task there.

1. Right click on your original project name in Eclipse and choose *Copy* from the menu.
2. Right click again and choose *Paste*.
3. Name your new project using your last name plus “-bonus”. E.g, if your last name were Smith, the project name would be smith-bonus

Bonus task

You may have noticed that the `in_priority...` routing function in task 3 has the same inefficiency as the original `send_from_head...` function: it needs to traverse the linked list to find the oldest priority 1 packet in the routing queue. In the worst case, this means traversing the entire list for every send.

The bonus task is to modify the system so that the `Ends` structure also contains a pointer to the oldest priority 1 packet in the queue, and that each priority 1 packet in the queue points to the next oldest priority 1 packet. You must then modify the `in_priority...` function to use these pointers to efficiently send priority 1 packets.

In implementing the bonus task you **are permitted to modify** `router.c` and `router.h` in any way you deem necessary. **However**, your implementation must not break any of the functions already implemented: you may add pointers to the `Packet` and `End` structures but you may not remove any, and any functions that manipulate the list must maintain all the original pointer relationships appropriately.

When you are finished

Ensure your name is in the header comment of the files and on your notes pages. Leave the computer logged in and your project open in Eclipse, and leave your notes pages on the keyboard.

End of the examination

[10]

(bonus)

Préparation de tâche boni

IMPORTANT ! N’essayez pas de résoudre la tâche boni dans le projet Eclipse qui contient votre solution aux tâches 1, 2 et 3. Créez plutôt un nouveau projet comme suit et faites-y tout le travail sur la tâche boni.

1. Cliquez-droit sur le nom de votre projet original dans Eclipse et choisissez *Copy* dans le menu contextuel.
2. Cliquez-droit de nouveau et sélectionnez *Paste*.
3. Nommez votre nouveau projet en utilisant votre nom de famille et « -bonus ». Par exemple, si votre nom de famille était Tremblay, le nom du projet serait tremblay-bonus.

Tâche boni

Vous aurez peut-être remarqué que la fonction de routage `in_priority...` de la tâche 3 présente la même inefficacité que la fonction de départ `send_from_head...` en ce qu’elle doit traverser la liste pour trouver le plus ancien paquet de priorité 1 dans la file. Dans le pire cas, il est possible qu’elle doive traverser la liste au complet.

La tâche boni consiste à modifier le système afin que la structure `Ends` contienne aussi un pointeur au plus ancien paquet de priorité 1 de la file qui lui-même pointe au prochain plus ancien paquet de priorité 1 et ainsi de suite. Vous devez ensuite modifier la fonction `in_priority...` pour qu’elle utilise ces pointeurs pour envoyer les paquets de priorité 1 de façon efficace.

Lorsque vous implémenterez la tâche boni, **vous pouvez modifier** `router.c` et `router.h` à votre guise. **Toutefois**, votre implémentation ne doit pas briser le fonctionnement des fonctions déjà implémentées : vous pouvez ajouter des pointeurs aux structures `Packet` et `End`, mais vous ne pouvez pas en enlever. De plus, toutes les fonctions manipulant la liste doivent maintenir les relations entre les pointeurs originales de façon appropriée.

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.

Fin de l’examen