



Piscine C

Rush 00

Staff 42 [piscine@42.fr](mailto:piscine@42.fr)

*Résumé: Ce document est le sujet du rush 00 de la piscine C de 42.*

# Table des matières

I	Consignes	2
II	Préambule	4
III	Sujet commun	5
IV	Rush 00	6
V	Rush 01	8
VI	Rush 02	9
VII	Rush 03	10
VIII	Rush 04	11

# Chapitre I

## Consignes

- Chaque membre du groupe peut inscrire le groupe en soutenance.
- Le groupe doit être inscrit en soutenance.
- Toute demande de précision sur un des sujets compliquera le sujet.
- Vous devez suivre la procédure de rendu pour tous vos exercices.
- Le sujet peut changer jusqu'à une heure avant le rendu.
- Les exercices sont très précisément ordonnés du plus simple au plus complexe. En aucun cas nous ne porterons attention ni ne prendrons en compte un exercice complexe si un exercice plus simple n'est pas parfaitement réussi.
- La Moulinette compile avec les flags -Wall -Wextra -Werror, et utilise `cc`.
- Si votre programme ne compile pas, vous aurez 0.
- Les exercices du rush sont à réaliser par groupe de 2, 3 ou 4.
- Le numéro du rush imposé pour votre groupe suivra la règle suivante : rang alphabétique de la première lettre du login du team leader (de 1 à 26) modulo 5.
- Vous devrez donc réaliser le sujet indiqué avec les binômes imposés et vous présenter en soutenance à l'heure dite avec tous vos binômes.
- Lors de la soutenance, le projet devra être terminé. Les soutenances servent à présenter et à expliquer votre travail dans les moindres détails.
- Chaque membre du groupe devra parfaitement être au courant du travail réalisé, chacun des membres sera interrogé, la note du groupe étant basée sur les moins bonnes explications.
- Évidemment, vous devez tout faire pour prendre contact avec vos binômes : téléphone, mail, pigeon voyageur, séance de spiritisme, etc. Aucune excuse ne sera acceptée en ce qui concerne les problèmes de groupe.
- Si après avoir vraiment tout essayé un de vos binômes reste injoignable : réalisez votre rush on s'arrangera en soutenance. Même si c'est le chef de groupe : vous

avez tous accès au dépôt.

- Vous pouvez, à titre optionnel, réaliser plusieurs sujets pour avoir un éventuel bonus.
- La Moulinette n'est pas très ouverte d'esprit. Elle ne cherche pas à comprendre le code qui ne respecte pas la Norme. La Moulinette utilise le programme **norminette** pour vérifier la norme de vos fichiers. Comprendre par là qu'il est stupide de rendre un code qui ne passe pas la **norminette**.



Il faudra absolument avoir le sujet obligatoire réalisé parfairement pour prétendre aux sujets bonus.



Pour cette journée, la norminette doit être lancée avec le flag `-R CheckForbiddenSourceHeader`. La Moulinette l'utilisera aussi.

# Chapitre II

## Préambule

Voici les paroles du générique de Minus et Cortex :

Minus : Dis Cortex, tu veux faire quoi cette nuit ?

Cortex : La même chose que chaque nuit, Minus : tenter de conquérir le monde !

C'est Minus et Cortex

C'est Cortex et Minus

L'une est plein d'astuce

L'autre un vrai nimbus

Deux souris diaboliques

Du génie génétique

Quelles canailles,

Ces p'tites souris cobayes, -bayes, -bayes, -bayes,  
-bayes, -bayes, -bayes, -bayes !

Dans leur tête elles projettent

Des plans sur la comète

Pour partir à la conquête

De toute la planète

C'est Cortex et Minus

C'est Minus et Cortex

Qui ont le réflexe

De vouloir sans complexes

Tendre une souricière

À la Terre toute entière

Quelles canailles,

Ces p'tites souris cobayes, -bayes, -bayes, -bayes,  
-bayes, -bayes, -bayes, -bayes.

Plutôt que de conquérir le monde, vous allez vous employer à conquérir ce rush !

# Chapitre III

## Sujet commun

	Exercice : 00
	rush0X
Dossier de rendu : <i>ex00/</i>	
Fichiers à rendre : <b>main.c</b> , <b>ft_putchar.c</b> , <b>rush0X.c</b>	
Fonctions Autorisées : <b>write</b>	

- Les fichiers à rendre seront le **main.c**, un **ft\_putchar.c** et votre **rush0X.c**, où 0X correspondra au numéro du rush. Par exemple, **rush00.c**.
- Exemple de **main.c** :

```
int main()
{
    rush(5, 5);
    return (0);
}
```

- Vous devrez donc écrire la fonction **rush** prenant en paramètre deux variables de type entier nommées respectivement **x** et **y**.
- Votre fonction **rush** devra afficher à l'écran un rectangle de **x** caractères de largeur, et **y** caractères de hauteur.
- Votre **main** sera modifié en soutenance pour pouvoir changer les paramètres de l'appel à la fonction **rush**. Par exemple, ce genre de chose sera testé :

```
int main()
{
    rush(123, 42);
    return (0);
}
```

# Chapitre IV

## Rush 00

- rush(5,3) affichera ceci :

```
$>./a.out
o---o
|
o---o
$>
```

- rush(5, 1) ceci :

```
$>./a.out
o---o
$>
```

- rush(1, 1) ceci :

```
$>./a.out
o
$>
```

- rush(1, 5) ceci :

```
$>./a.out
o
|
|
|
|
$>
```

- rush(4, 4) ceci :

```
$> ./a.out
o--o
| |
| |
o--o
$>
```

# Chapitre V

## Rush 01

- rush(5,3) affichera ceci :

```
$>./a.out
/**\
* *
\*/
$>
```

- rush(5, 1) ceci :

```
$>./a.out
/**\
$>
```

- rush(1, 1) ceci :

```
$>./a.out
/
$>
```

- rush(1, 5) ceci :

```
$>./a.out
/
*
*
*
\$
$>
```

- rush(4, 4) ceci :

```
$>./a.out
/**\
* *
* *
\*/
$>
```

# Chapitre VI

## Rush 02

- rush(5,3) affichera ceci :

```
$>./a.out
ABBBA
B   B
CBBBC
$>
```

- rush(5, 1) ceci :

```
$>./a.out
ABBBA
$>
```

- rush(1, 1) ceci :

```
$>./a.out
A
$>
```

- rush(1, 5) ceci :

```
$>./a.out
A
B
B
B
C
$>
```

- rush(4, 4) ceci :

```
$>./a.out
ABBA
B   B
B   B
CBBC
$>
```

# Chapitre VII

## Rush 03

- rush(5,3) affichera ceci :

```
$>./a.out
ABBBC
B B
ABBBC
$>
```

- rush(5, 1) ceci :

```
$>./a.out
ABBBC
$>
```

- rush(1, 1) ceci :

```
$>./a.out
A
$>
```

- rush(1, 5) ceci :

```
$>./a.out
A
B
B
B
A
$>
```

- rush(4, 4) ceci :

```
$>./a.out
ABBC
B B
B B
ABBC
$>
```

# Chapitre VIII

## Rush 04

- rush(5,3) affichera ceci :

```
$>./a.out
ABBBC
B   B
CBBBA
$>
```

- rush(5, 1) ceci :

```
$>./a.out
ABBBC
$>
```

- rush(1, 1) ceci :

```
$>./a.out
A
$>
```

- rush(1, 5) ceci :

```
$>./a.out
A
B
B
B
C
$>
```

- rush(4, 4) ceci :

```
$>./a.out
ABBC
B   B
B   B
CBBA
$>
```



Piscine C

Rush 01

Staff 42 [piscine@42.fr](mailto:piscine@42.fr)

*Résumé: Ce document est le sujet du Rush 01 de la piscine C de 42.*

# Table des matières

I	Consignes	2
II	Préambule	3
III	Sujet	4
IV	Annexe	6
V	Bonus	8

# Chapitre I

## Consignes

- Chaque membre du groupe peut inscrire le groupe en soutenance.
- Le groupe doit être inscrit en soutenance.
- Toute demande de précision sur le sujet compliquera le sujet.
- Vous devez suivre la procédure de rendu pour le sujet.
- Le sujet peut changer jusqu'à une heure avant le rendu.
- La Moulinette compile avec les flags -Wall -Wextra -Werror, et utilise cc.
- Si votre programme ne compile pas, vous aurez 0.
- Vous devrez donc réaliser le sujet indiqué avec les binômes imposés et vous présenter en soutenance à l'heure dite avec tous vos binômes.
- Lors de la soutenance, le projet devra être terminé. Les soutenances servent à présenter et à expliquer votre travail dans les moindres détails.
- Chaque membre du groupe devra parfaitement être au courant du travail réalisé, chacun des membres sera interrogé, la note du groupe étant basée sur les moins bonnes explications.
- Évidemment, vous devrez tout faire pour prendre contact avec vos binômes : téléphone, mail, pigeon voyageur, séance de spiritisme, etc. Aucune excuse ne sera acceptée en ce qui concerne les problèmes de groupe.
- Si après avoir vraiment tout essayé un de vos binômes reste injoignable : réalisez rush on s'arrangera en soutenance. Même si c'est le chef de groupe : vous avez tous accès au dépôt.
- Bien sûr, votre travail devra être à la Norme, soyez très rigoureux.
- Bon travail à tous.

# Chapitre II

## Préambule

Voici ce que Wikipédia a à dire à propos du paresseux à deux doigts :

Le paresseux à deux doigts est réputé être l'animal le plus lent au monde. L'adulte pèse en moyenne 4 à 8 kg et a la taille d'un petit chien : 60 à 85 cm de long avec une queue de 1,4 à 3,3 cm. Il présente un cou court, 4 longs membres d'égale longueur terminés par deux griffes incurvées à l'avant et trois à l'arrière ayant jusqu'à 7 cm de long. La tête est courte et plate, avec un nez retroussé, des oreilles rudimentaires et de grands yeux.

Ils passent près de 80 % de leur temps à dormir. Les paresseux se déplacent très lentement. Ils dorment, mangent, s'accouplent, donnent naissance et élèvent leurs petits sans jamais descendre de leur arbre, dans lequel ils restent accrochés aux branches, la tête en bas. Ils ne vont que très rarement à terre, pour changer d'arbre quand la nourriture s'y fait rare ou pour déféquer, une fois par semaine.

Ils sont parfaitement camouflés dans les arbres, leur corps brun verdâtre ramassé sur lui-même passant facilement pour un nid de termites ou une excroissance de bois au yeux des prédateurs. En cas d'attaque, ils se défendent avec dents et griffes bien qu'ils soient d'un tempérament pacifique, comptant plus sur leur capacité à se fondre dans le décor. Ils restent immobiles pendant de longues heures mais se grattent régulièrement, trahissant ainsi parfois leur présence.

Contrairement aux paresseux, vous n'allez pas avoir beaucoup de temps pour dormir ce week-end.

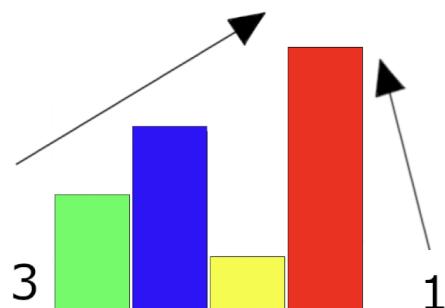
Quel dommage.

# Chapitre III

## Sujet

	Exercice : 00
	rush-01
Dossier de rendu : <i>ex00/</i>	
Fichiers à rendre : <b>Tous les fichiers nécessaires</b>	
Fonctions Autorisées : <b>write, malloc, free</b>	

- Votre code source sera compilé par la commande : `cc -Wall -Wextra -Werror -o rush-01 *.c`
- Votre dossier de rendu devra contenir tout ce qui sera nécessaire à la compilation de votre programme.
- Créez un programme qui résoud le problème suivant :
- Sur une map de 4 par 4, placez des caisses de taille 1 à 4 de manière à ce que chaque ligne et colonne voie le bon nombre de caisses de chaque point de vue possible.
- Exemple : la caisse de taille 3 cache ici la caisse de taille 1, ce qui fait qu'il n'y a que 3 boîtes visibles de la gauche. De la droite, la caisse de taille 4 cache toutes les autres caisses, il y a donc une seule caisse de visible.



- Chacun des vues (deux par ligne et deux par colonne) aura une valeur donnée,

allant de 1 à 4. Votre programme doit placer les caisses correctement, tout en faisant attention à n'avoir qu'une seule caisse de chaque hauteur sur chaque ligne et colonne.

- Si il y a plusieurs solution, vous devez afficher la première que vous trouverez.
- Le programme sera lancé de la façon suivante :

```
> ./rush-01 "col1up col2up col3up col4up col1down col2down col3down col4down row1left row2left  
row3left row4left rowright row2right row3right row4right"
```

- (cf. annexe 1)
- col1up correspond à la valeur pour le point de vue du haut de la colonne de gauche. Chaque valeur doit être comprise entre 1 et 4.
- Vous devrez afficher en sortie la résolution comme suit :

```
>./rush-01 "4 3 2 1 1 2 2 2 4 3 2 1 1 2 2 2" | cat -e  
1 2 3 4$  
2 3 4 1$  
3 4 1 2$  
4 1 2 3$
```

- (cf. annexe 2 et 3)
- En cas d'erreur vous écrirez seulement "Error" suivi d'un retour à la ligne.

# Chapitre IV

## Annexe

Ce qui suit est une représentation artistique de votre programme. Vous devez, bien entendu, respecter les consignes de rendu telles que décrites dans la partie précédente. Ces visualisations ont pour unique but de vous aider à apprêhender ce sujet.

- Annexe 1 :

	col1up	col2up	col3up	col4up	
row1left					row1right
row2left					row2right
row3left					row3right
row4left					row4right
	col1down	col2down	col3down	col4down	

- Représentation du lancement du programme avec col\_up, col\_down, row\_left et row\_right
- Annexe 2 :

	4	3	2	1	
4				1	
3				2	
2				2	
1				2	
	1	2	2	2	

- En remplaçant les col\* et row\*, on obtient ceci.

- Annexe 3 :

	4	3	2	1	
4	1	2	3	4	1
3	2	3	4	1	2
2	3	4	1	2	2
1	4	1	2	3	2
	1	2	2	2	

- Votre programme doit remplir les cases intérieures, et renvoyer la réponse tel que demandé dans le sujet.

# **Chapitre V**

## **Bonus**

- Ce projet peut être fait sur différentes tailles. Si votre programme gère une carte de 6 par 6 par exemple, vous pouvez la rendre en lieu et place du projet pour obtenir des points bonus.
- Vous pouvez aussi rendre un programme qui gère toutes les cartes (3 à 9) pour encore plus de bonus !



Piscine C

Rush 02

Staff 42 [piscine@42.fr](mailto:piscine@42.fr)

*Résumé: Ce document est le sujet du rush 02 de la piscine C de 42.*

# Table des matières

I	Consignes	2
II	Préambule	3
III	Le sujet	4
IV	Bonus	6

# Chapitre I

## Consignes

- Le groupe sera automatiquement inscrit en soutenance.
- Toute demande de précision sur le sujet complexifiera probablement le sujet.
- Vous devez suivre la procédure de rendu pour tous vos exercices.
- Le sujet peut changer jusqu'à une heure avant le rendu.
- La Moulinette compile avec les flags -Wall -Wextra -Werror, et utilise `cc`.
- Si votre programme ne compile pas, vous aurez 0.
- Vous devez rendre un Makefile, qui compile votre projet à l'aide des règles `$NAME clean` et `fclean`
- Vous devrez donc réaliser le sujet indiqué avec les binômes imposés et vous présenter en soutenance à l'heure dite avec tous vos binômes.
- Lors de la soutenance, le projet devra être terminé. Les soutenances servent à présenter et à expliquer votre travail dans les moindres détails.
- Chaque membre du groupe devra parfaitement être au courant du travail réalisé, chacun des membres sera interrogé, la note du groupe étant basée sur les moins bonnes explications.
- évidemment, vous devez tout faire pour prendre contact avec vos binômes : téléphone, mail, pigeon voyageur, séance de spiritisme, etc. Aucune excuse ne sera acceptée en ce qui concerne les problèmes de groupe.
- Si après avoir vraiment tout essayé un de vos binômes reste injoignable : réalisez votre rush on s'arrangera en soutenance. Même si c'est le chef de groupe : vous avez tous accès au dépôt.
- Bien sûr, votre travail devra être à la Norme : soyez très rigoureux. Bon travail !

# Chapitre II

## Préambule

Voici la recette du Quatre-Quart :

4 Ingrédients (Pour 1 gateau) :

- 4 oeufs
- 200g de sucre
- 200g de farine
- 200g de beurre demi-sel

Préparation :

- Préchauffez le four à 180°C.
- Commencez par faire fondre le beurre pour lui laisser le temps de refroidir.
- Fouettez les œufs avec le sucre pendant 5 minutes environ.

Il faut que le mélange devienne clair et mousseux.

- Ajoutez la farine (et éventuellement l'arôme de votre choix) et fouettez quelques secondes, juste le temps de l'incorporer.

Si vous fouettez trop, vous allez faire retomber toute votre pâte.

- Ajoutez le beurre fondu et mélangez avec une maryse ou une cuillère en bois.

- Beurrez votre moule à cake et versez la préparation à l'intérieur.

- Plantez la lame d'un couteau dans la pâte dans toute la longueur du cake.

- Enfournez 10 minutes à 180°C, puis baissez la température du four à 145°C.
- Laisser cuire encore 40 minutes.

Le dessus du quatre-quarts doit être légèrement doré, mais pas trop.

FIGURE II.1 – Ca devrait ressembler à ça.

# Chapitre III

## Le sujet

	Exercice : 00
	rush-02
Dossier de rendu : <i>ex00/</i>	
Fichiers à rendre : <b>Makefile et tous les fichiers nécessaires</b>	
Fonctions Autorisées : <b>write, malloc, free, open, read, close</b>	

- Vous devez réaliser un programme qui prend un nombre en argument en entrée et qui le converti en sa valeur écrite.
- Nom de l'exécutable : **rush-02**
- Votre code source sera compilé par la commande :

```
make fclean  
make
```

- Votre programme peut recevoir jusqu'à 2 arguments :
  - Si il n'y a qu'un seul argument, il s'agit de la valeur que vous devez convertir.
  - Si il y a 2 arguments, le premier est le dictionnaire à utiliser, et le deuxième est la valeur à convertir.
- Si l'argument n'est pas un entier positif valide, vous devrez renvoyer "Error" suivi d'un "\n".
- Si le dictionnaire ne vous permet pas de convertir la valeur demandée, vous devrez renvoyer "Error" suivi d'un "  
n"
- Pour des soucis d'harmonisation, votre programme parlera anglais.
- Votre programme doit parser le dictionnaire passé en ressources. Les valeurs données doivent être utilisées pour imprimer les résultats. Celles-ci pourront être mo-

difiées.

- Toute mémoire allouée sur la heap (avec malloc(3)) doit être libérée proprement.
- Le dictionnaire suivra les règles suivantes :

```
[a number] [0 to n spaces]:[0 to n spaces][n'importe quel caractere imprimable]\n
```

- Plusieurs espaces doivent toujours être remplacés par un unique espace.
- Le dictionnaire aura toujours au moins les clés données dans le dictionnaire de référence. Leur valeur peut être modifiée, des entrées peuvent être rajoutées, mais les clés initiales ne peuvent pas être retirées.
- Les entrées du dictionnaire peuvent être rangées dans n'importe quel ordre.
- Il peut y avoir des lignes vides dans le dictionnaire.
- Si vous avez la moindre erreur lors du parsing du dictionnaire, vous devez afficher "Dict Error n". Votre programme doit quitter ensuite proprement.

- Example :

```
$> ./rush-02 42 | cat -e
forty two$
$> ./rush-02 0 | cat -e
zero$
$> ./rush-02 10.4 | cat -e
error$
$> ./rush-02 100000 | cat -e
one hundred thousand$
$> grep "20" numbers.dict | cat -e
20      : hey      everybody !$
$> ./rush-02 20 | cat -e
hey everybody !$
```

# Chapitre IV

## Bonus

- Utiliser `,`, `-`, `and` pour être syntaxiquement correct.
- Avoir la possibilité de changer de langage (ex. le faire en français). Vous pouvez à cet effet rajouter votre propre dictionnaire qui contiendra les entrées nécessaires.sa
- Utiliser `read` pour lire l'entrée standard lorsqu'il n'y a pas d'argument
- Si vous avez d'autres idées et qu'elles apportent quelque chose au projet, allez-y !



# Projet Piscine C

BSQ

42 staff [staff@42.fr](mailto:staff@42.fr)

*Résumé: Trouveras-tu le plus grand carré ?*

# Table des matières

I	Préambule	2
II	Sujet	4
III	Consignes	6
IV	Notation	7
V	Annexe	8

# Chapitre I

## Préambule

Extraits de Life, The Universe, and Everything :

« Important facts from Galactic history, number one :

(Reproduced from the Siderial Daily Mentioner's Book of popular Galactic History.)

The night sky over the planet Krikkit is the least interesting sight in the entire Universe. »

« The Krikkit Wars belonged to the ancient past of the Galaxy, and Zaphod had spent most of his early history lessons plotting how he was going to have sex with the girl in the cybercubicle next to him, and since his teaching computer had been an integral part of this plot it had eventually had all its history circuits wiped and replaced with an entirely different set of ideas which had then resulted in it being scrapped and sent to a home for Degenerate Cybermats, whither it was followed by the girl who had inadvertently fallen deeply in love with the unfortunate machine, with the result (a) that

Zaphod never got near her and (b) that he missed out on a period of ancient history that

would have been of inestimable value to him at this moment. »

« The game you know as cricket, [Slartibartfast] said, and his voice still seemed to be wandering lost in subterranean passages, is just one of those curious freaks of racial memory which can keep images alive in the mind aeons after their true significance has been lost in the mists of time. Of all the races on the Galaxy, only the English could

possibly revive the memory of the most horrific wars ever to sunder the Universe and transform it into what I'm afraid is generally regarded as an incomprehensibly dull and

pointless game. »

« Although it has been said that on Earth alone in our Galaxy is Krikkit (or cricket) treated as fit subject for a game, and that for this reason the Earth has been shunned, this does only apply to our Galaxy, and more specifically to our dimension. In some of the higher dimensions they feel they can more or less please themselves, and have been playing a peculiar game called Brockian Ultra-Cricket for whatever their transdimensional equivalent of billions of years is. »

Saviez-vous que d'autres variantes du Krikkit existent ? Le Brockian Simple-Qricket a la particularité de se jouer sur un terrain de superficie variable. La seule constante est sa forme : un carré parfait. À travers les âges, de nombreuses techniques ont été adoptées afin de pouvoir optimiser l'espace disponible. Le BSQ, l'un des premiers jeux

écoresponsable, demande en effet de respecter l'environnement. Aucun obstacle ne peut

être déplacé pour la création du terrain de jeu, que ce soit arbre, pierre, panneau de signalisation, ancre de vaisseau ou, lors d'une rencontre particulière entre l'équipe des Administrés Légaux Vogons et celles des Titans de Titans, secteur galactique entier.

L'ironie étant bien entendue le destin du Galactic Sector ZZ9 Plural Z Alpha peu après la

victoire des Vogons.

# Chapitre II

## Sujet

- Le plus grand carré :
  - Il s'agit de trouver le plus grand carré possible sur un plateau en évitant des obstacles.
  - Un plateau vous est transmis dans un fichier passé en argument du programme.
  - La première ligne du plateau contient les informations pour lire la carte :
    - Le nombre de lignes du plateau ;
    - Le caractère "vide" ;
    - Le caractère "obstacle" ;
    - Le caractère "plein".
  - Le plateau est composé de lignes de '**caractère "vide"**' et de '**caractère "obstacle"**'.
  - Le but du programme est de remplacer les '**caractère "vide"**' par des '**caractère "plein"**' pour représenter le plus grand carré possible.
  - Dans le cas où il y en a plusieurs solutions, on choisira de représenter le carré le plus en haut à gauche.

- Carte valide :
  - Toutes les lignes doivent avoir la même longueur.
  - Il y a au moins une ligne d'au moins une case.
  - À la fin de chaque ligne il y a un retour à la ligne.
  - Les caractères présent dans la carte doivent être uniquement ceux présenté à la première ligne.
  - En cas de carte invalide vous afficherez sur la sortie d'erreur : `map error` suivi d'un retour à la ligne puis il passera traitement du plateau suivant.
- Exemple de fonctionnement :

```
%>cat example_file
9.ox
.....
....o.....
.....o.....
.....
....o.....
.....o.....
.....
.....o.....o....
..o.....o.....
%>./bsq example_file
.....XXXXXX.....
....OXXXXXXX.....
....XXXXXXXo.....
....XXXXXX.....
....OXXXXXXX.....
....XXXXXXX...o....
....XXXXXX.....
....o.....o.....
..o.....o.....
%>
```



C'est bien un carré. Même si cela n'y ressemble pas visuellement

# Chapitre III

## Consignes

- L'exécutable doit s'appeler `bsq` et se trouver dans le répertoire principal.
- Votre projet doit être à la Norme.
- Vous ne pouvez utiliser que les éléments vus durant votre Piscine.
- Le répertoire doit avoir un fichier auteur dans lequel vous devez mettre vos logins.

```
$>cat auteur  
login_1:login_2  
$>
```

- Le programme peut prendre de 1 à N fichiers en paramètre.
- Si aucun argument il lira sur l'entrée standard.
- Votre Makefile devra compiler le projet.
- Vous ne pouvez utiliser que les fonctions `exit`, `open`, `close`, `write`, `read`, `malloc` et `free`.
- Vous pouvez poser vos questions dans le forum.

# Chapitre IV

## Notation

- La notation de la BSQ s'effectue en deux temps :
  - Nous commençons par tester **las funcionalidades** (sur 10 points). Votre programme doit fonctionner.
  - La deuxième partie est **la optimización** du code. Elle est notée sur deux parties indépendantes : la vitesse d'exécution (5 points) la mémoire utilisé (5 points).
    - Elle n'est vérifiée que si toute la première partie est correcte.
    - Toutes les BSQ ayant validé l'intégralité de la première partie seront en compétition sur les deux tests d'optimisations.
    - Le meilleur projet remportera l'intégralité des points de cette partie.
    - Les projets suivants remporteront des points en fonction de leur classement : pour la vitesse d'exécution, les plus rapides auront plus de points que les plus lents. Et pour la mémoire, ceux qui ont utilisé le moins de mémoire auront plus de points que ceux qui en utilisent beaucoup.

Bon courage à tous !

# Chapitre V

## Annexe

- Générateur de plateaux perl

```
#!/usr/bin/perl

use warnings;
use strict;

die "program x y density" unless (scalar(@ARGV) == 3);

my ($x, $y, $density) = @ARGV;

print "$y.\n";
for (my $i = 0; $i < $y; $i++) {
    for (my $j = 0; $j < $x; $j++) {
        if (int(rand($y) * 2) < $density) {
            print "o";
        }
        else {
            print ".";
        }
    }
    print "\n";
}
```