



Piscine C++ - d06

IOStream, String et objets

David “Joachim” Pineau pineau_d@epitech.eu

Abstract: Ce document est le sujet du d06

Table des matières

I	Remarques générales	2
II	Exercice 0	4
III	Exercice 1	5
IV	Exercice 2	7
V	Exercice 3	9
VI	Exercice 4	11
VII	Exercice 5	13
VIII	Exercice 6	17

Chapitre I

Remarques générales

- Si vous faites la moitié des exercices car vous avez du mal, c'est normal. Par contre, si vous faites la moitié des exercices par flemme et vous tirez à 14h, vous AUREZ des surprises. Ne tentez pas le diable.
- Les noms de fichiers qui vous sont imposés doivent être respectés A LA LETTRE, de même que les noms de fonctions.
- Toute fonction implémentée dans un header ou header non protégé signifie 0 à l'exercice.
- Toutes les sorties se font sur la sortie standard et sont terminées par un retour à la ligne sauf si le contraire est précisé explicitement.
- Les noms de fichiers qui vous sont imposés doivent être respectés À LA LETTRE, de même que les noms de classes et de fonctions membres / méthodes.
- Rappelez-vous que vous faites du C++ et non plus du C. Par conséquent, les fonctions suivantes sont INTERDITES, et leur utilisation sera sanctionnée par un -42 :
 - `*alloc`
 - `*printf`
 - `free`
- De façon générale, les fichiers associés à une classe seront toujours `NOM_DE_LA_CLASSE.h` et `NOM_DE_LA_CLASSE.cpp` (avec le nom de la classe écrit en minuscule, s'il y a lieu).
- Les répertoires de rendus sont `ex00`, `ex01`, ..., `exN`
- Toute utilisation de `"friend"` se soldera par un -42, `no questions asked`.
- Lisez attentivement les exemples, ils peuvent requérir des choses que le sujet ne dit pas...


- Vous n'avez bien évidemment droit qu'aux outils de C++ que vous aurez vu depuis le début de la piscine. Toute lib externe étant donc proscrite.
- Ces exercices vous demandent de rendre beaucoup de classes, mais la plupart sont relativement courtes si vous faites ça intelligemment. Donc, halte à la flemme!
- Lisez **ENTIÈREMENT** le sujet d'un exercice avant de le commencer!
- **RÉFLÉCHISSEZ**. Par pitié.

COMPILATION DES EXERCICES :

- La moulinette compile votre code avec les flags : `-W -Wall -Werror` .
- Chacun de vos includes doit pouvoir être inclus indépendamment des autres. Il devra donc contenir tous les includes dont il dépendra.
- Tous les fichiers d'include seront inclus dans le main de correction.
- Notez bien qu'aucun de vos fichiers ne doit contenir de fonction `main` sauf si le contraire est explicité. Nous utiliserons notre propre fonction `main` pour compiler et tester votre code.
- Le sujet peut-être modifié jusqu'à 4h avant le rendu!
- Le compilateur à utiliser est `g++` !
- Vous devez utiliser les file streams pour résoudre certains exercices. La vidéo de cours sur les streams vous explique le fonctionnement des IOStreams. Il en existe un grand nombre d'implémentations, dont les file streams. Ce type particulier de stream remplace la manipulation de fichiers en mode `C` à base de `*open` et `close` .

Chapitre II

Exercice 0

	Exercice : 00	points : 2
IOStream : my_cat		
Répertoire de rendu: (DÉPOT SVN - piscine_cpp_d06-promo-login_x)/ex00		
Compilateur : g++	Flags de compilation: -Wall -Wextra -Werror	
Makefile : Oui	Règles : all, clean, fclean, re	
Fichiers a rendre : Makefile, et les fichiers de votre programme		
Remarques : Pour cet exercice, rendez un programme complet, main inclus		
Fonctions Interdites : *alloc, free, *printf, open, fopen, le mot-clé using		

Votre Makefile devra générer un binaire nommé : `my_cat`

Vous devez écrire une commande `CAT(1)` simplifiée. Celle-ci devra prendre en paramètre un ou plusieurs fichiers, et n'aura pas à gérer l'entrée standard.

En cas d'erreur, (fichier non trouvé, `permission denied`, etc...), vous devrez systématiquement afficher sur la sortie d'erreur :

```
1 my_cat: <file>: No such file or directory
```


Où `file` sera remplacé par le nom du fichier qui n'a pu être lu.

Dans le cas où aucun paramètre n'est donné à votre `my_cat`, vous devrez afficher sur la sortie standard :

```
1 my_cat: Usage : ./my_cat file [...]
```

Chapitre III

Exercice 1

	Exercice : 01	points : 2
Conversion de température		
Répertoire de rendu: (DÉPOT SVN - piscine_cpp_d06-promo-login_x)/ex01		
Compilateur : g++	Flags de compilation: -W -Wall -Werror	
Makefile : Oui	Règles : all, clean, fclean, re	
Fichiers a rendre : Makefile, et les fichiers de votre programme		
Remarques : Pour cet exercice, rendez un programme complet, main inclus		
Fonctions Interdites : *alloc, free, *printf, open, fopen, *scanf - le mot clé using		

Votre Makefile devra générer un binaire nommé : `my_convert_temp`

Le but de cet exercice est d'écrire un programme qui convertira les températures de l'unité **Celsius** à **Fahrenheit** , et inversement.

La formule de conversion est la suivante (Oui on sait la formule est fausse) :

```
1 Celsius = 5.0 / 9.0 * ( Fahrenheit - 32 )
```

Le programme devra demander sur l'entrée standard (séparées par un ou plusieurs espaces) :

- une température
- une échelle


Exemple :

```
>./my_convert_temp
-10 Celsius
      14.000      Fahrenheit
>./my_convert_temp
46.400 Fahrenheit
      8.000      Celsius
```

Tous les résultats doivent être affichés dans deux colonnes alignées à droite, paddées sur 16, avec une précision au 1000ème.

Chapitre IV

Exercice 2

	Exercice : 02	points : 3
Hopital : Le patient		
Répertoire de rendu: (DÉPOT SVN - piscine_cpp_d06-promo-login_x)/ex02		
Compilateur : g++	Flags de compilation: -W -Wall -Werror	
Makefile : Non	Règles : n/a	
Fichiers a rendre : sickkoala.h, sickkoala.cpp		
Remarques : n/a		
Fonctions Interdites : *alloc, free, *printf, open, fopen - mot-clé using		

Vous allez maintenant commencer à travailler sur une simulation de la santé des Koalas. Pour commencer, nous avons besoin de patients. Vous allez donc mettre en place une classe SickKoala. Voici les informations qui vous permettront de coder cette classe :

- Il ne peuvent être instanciés sans nom `string`
- Lors de leur destruction, ils affichent sur la sortie standard :

```
1  Mr.[nom]: Kreooogg !! Je suis gueri!!!
```

- Ils possèdent la fonction membre `poke` ne prenant aucun paramètre et ne renvoie rien. Lors de l'appel, cette fonction membre affiche :

```
1  Mr.[nom]: Gooooeeerrk !! :'(
```

- Ils possèdent la fonction membre `takeDrug` qui prend une `string` en paramètre et renvoie `true` si la `string` correspond à une des possibilités suivantes :
 - `mars` (insensible à la casse). La fonction affiche alors :


```
1      Mr.[nom] : Mars, et ca Kreog !
```

- Buronzand (sensible à la casse). La fonction affiche alors :

```
1      Mr.[nom] : Et la fatigue a fait son temps !
```

Dans le cas contraire, la fonction renvoie `false` et affiche :

```
1      Mr.[nom] : Goerk !
```

- Il arrive que les SickKoalas délirent lors de trop fortes fièvres. Afin de simuler cela, les Sickkoalas auront la fonction membre `overDrive` ne renvoyant rien et prenant en paramètre une `string`. Cette fonction membre affichera la chaîne passée en paramètre, précédée de "Mr.[nom] ", et où toutes les occurrences de "Kreog !" sont remplacées par "1337 !" .
Par exemple, la chaîne :

```
1      Kreog ! Ca boume ?
```

Deviendra :

```
1      Mr.[nom] : 1337 ! Ca boume ?
```




Pour toutes les sorties de cet exercice, vous veillerez à remplacer tous les [nom] par le nom du SickKoala.



Tous les affichages seront suivis d'un retour à la ligne, à moins que le contraire ne soit précisé.

Chapitre V

Exercice 3

	Exercice : 03	points : 3
Hopital : L'infirmière		
Répertoire de rendu: (DÉPOT SVN - piscine_cpp_d06-promo-login_x)/ex03		
Compilateur : g++	Flags de compilation: -W -Wall -Werror	
Makefile : Non	Règles : n/a	
Fichiers a rendre : koalanurse.h, koalanurse.cpp, sickkoala.h, sickkoala.cpp		
Remarques : n/a		
Fonctions Interdites : *alloc, free, *printf, open, fopen - mot-clé using		

Nous avons désormais des malades. Il nous faut rajouter une infirmière pour les soigner.

Vous allez donc coder une infirmière koala : la KoalaNurse.

Voici les infos dont vous disposez pour créer votre KoalaNurse :

- Chaque KoalaNurse possède un identifiant numerique (ID), qui sera précisé à la création de l'objet (on ne pourra créer une infirmière sans lui donner une ID)
- Lors de sa destruction, une infirmière exprime son soulagement de la facon suivante :

```
1 Nurse [ID]: Enfin un peu de repos !
```

- Elle peut donner un médicament à un patient avec la fonction membre `giveDrug` qui prend en paramètres :
 - une `string` (le médicament)
 - un pointeur sur le patient.

Cette fonction membre ne renvoie rien.

L'infirmière donne alors le médicament au malade.

- L'infirmière peut lire le rapport d'un médecin avec la fonction membre `readReport` prenant en paramètre un nom de fichier `string` .
 - Le nom du fichier sera constitué du nom du koala malade avec l'extension `.report` .
 - Le fichier contiendra le nom du médicament à donner au malade.

La fonction membre renvoie le nom du médicament (`string`) Et affichera sur la sortie standard :

```
1 Nurse [ID]: Kreog ! Il faut donner un [medicament] a Mr.[nomPatient] !
```

Si le fichier `.report` n'existe pas ou n'est pas valide, ne rien afficher et renvoyer une `string` vide.

- L'infirmière peut pointer ses horaires avec la fonction membre `timeCheck` qui ne prend pas de paramètres et ne renvoie rien.
Elle appellera cette fonction membre quand elle commencera à travailler et lorsqu'elle s'arrêtera (parce qu'elle est une travailleuse consciencieuse).
Lorsqu'elle pointe à l'arrivée, elle affiche :

```
1 Nurse [ID]: Je commence le travail !
```

Lorsqu'elle repart :

```
1 Nurse [ID]: Je rentre dans ma foret d'eucalyptus !
```

Il vous appartient de trouver un moyen de déterminer quand elle arrive et quand elle repart. Sachez simplement que par défaut elle n'est pas encore à l'hôpital. La KoalaNurse étant par vocation humanitaire, elle acceptera tout travail. Même hors de l'hôpital.


Seul un appel à la fonction membre `timeCheck` permettra de modifier le status de travail de la KoalaNurse : si elle ne travaillait pas, elle se met au travail; et si elle travaillait elle s'arrête.



Dans cet exercice, vous veillerez à remplacer tous les [ID] par l'id de la KoalaNurse qui fait l'affichage.

Chapitre VI

Exercice 4

	Exercice : 04	points : 3
Hopital : Le Docteur		
Répertoire de rendu: (DÉPOT SVN - piscine_cpp_d06-promo-login_x)/ex04		
Compilateur : g++	Flags de compilation: -W -Wall -Werror	
Makefile : Non	Règles : n/a	
Fichiers a rendre : koalanurse.h, koalanurse.cpp, sickkoala.h, sickkoala.cpp, koaladoctor.h, koaladoctor.cpp		
Remarques : n/a		
Fonctions Interdites : *alloc, free, *printf, open, fopen, srand, srandom - mot-clé using		

Vous allez commencer par modifier vos classes précédentes.

- Ajoutez une fonction membre `getName` dans la classe `SickKoala` ne prenant aucun paramètre et renvoyant le nom du malade (`string`)

Nous avons maintenant le malade, l'infirmière qui le soigne, et il nous faut le médecin qui va donner des directions a l'infirmière. Vous allez donc coder une simulation de celui-ci sous la forme d'une classe `KoalaDoctor` .

Voici ce que nous savons au sujet du `KoalaDoctor` :

- Il doit être instancié avec un nom (`string`). Lors de la construction il affichera sur la sortie standard :

```
1 Dr.[nom]: Je suis le Dr.[nom] ! Comment Kreoggez-vous ?
```

- Il peut ausculter un patient avec la fonction membre `diagnose` qui prend un pointeur sur le patient à ausculter en paramètre. Cette fonction membre affiche sur la sortie standard :

```
1 Dr.[nom]: Alors, qu'est-ce qui vous goerk, Mr.[nomPatient] ?
```

Puis elle appelle la fonction membre `poke` du `SickKoala`.

Il va alors écrire son rapport pour les Nurses dans un fichier nommé `[nomPatient].report` qui contiendra le nom du médicament à donner, choisi aléatoirement parmi 5 médicaments :

- `mars`
- `Buronzand`
- `Viagra`
- `Extasy`
- `Feuille d'eucalyptus`

Pour cela vous utiliserez `random() % 5` , que vous appliquerez à la liste ci-dessus, dans l'ordre donné. La fonction `srandom` sera bien évidemment appelée dans le main de correction.

- Il pointe ses horaires avec la fonction membre `timeCheck` (ne prend pas de paramètre et ne renvoie rien) lorsqu'il commence à travailler et lorsqu'il arrête, parce que c'est un travailleur consciencieux.
Lorsqu'il commence le travail, il affiche :

```
1 Dr.[nom]: Je commence le travail !
```

Lorsqu'il quitte le travail, il affiche :

```
1 Dr.[nom]: Je rentre dans ma foret d'eucalyptus !
```


Le `KoalaDoctor` étant de vocation humanitaire, il acceptera tout travail même hors de l'hôpital.



Dans cet exercice, pour les sorties à générer, tous les `[nom]` sont à remplacer par le nom du `KoalaDoctor` qui génère la sortie (nom avec lequel vous avez initialisé votre instance de `KoalaDoctor`) Pour les `[nomPatient]` , il faudra les remplacer par le nom du `SickKoala` que votre instance de `KoalaDoctor` est en train de manipuler.

Chapitre VII

Exercice 5

	Exercice : 05	points : 3
Hopital : une manière de gérer tout ça		
Répertoire de rendu: (DÉPOT SVN - piscine_cpp_d06-promo-login_x)/ex05		
Compilateur : g++	Flags de compilation: -W -Wall -Werror	
Makefile : Non	Règles : n/a	
Fichiers à rendre : koalanurse.h, koalanurse.cpp, sickkoala.h, sickkoala.cpp, koaladoctor.h, koaladoctor.cpp, sickkoalalist.h, sickkoalalist.cpp, koalanurselist.h, koalanurselist.cpp, koaladoctorlist.cpp, koaladoctorlist.h		
Remarques : Ici, le récursif pourrait réduire le temps de dev...		
Fonctions Interdites : *alloc, free, *printf, open, fopen, srand, srandom - mot-clé using		

Pour cet exercice, il va nous falloir modifier les classes `KoalaNurse` et `KoalaDoctor`

- Ajoutez une fonction membre `getID` à la classe `KoalaNurse` . Elle ne prend aucun paramètre et renvoie un `int` .
- Ajoutez une fonction membre `getName` à la classe `KoalaDoctor` qui ne prend aucun paramètre et renvoie une `string` .

Nous avons maintenant besoin d'observer tout ce beau monde à l'oeuvre en harmonie. Il est nécessaire de pouvoir gérer plusieurs patients, docteurs ou infirmières en même temps. Pour y parvenir, nous allons coder une petite liste pour chacun.



Quand ici nous parlons d'un maillon de la chaine, nous parlons d'un objet de `*List`.

Nous allons maintenant construire les listes.

- La classe `SickKoalaList`

- À la construction, prend un pointeur sur `SickKoala` . Ce pointeur peut être `NULL`
- Possède une fonction membre `isEnd` . Cette fonction membre ne prend pas de paramètres et renvoie un booléen valant `true` si le `SickKoalaList` est le dernier maillon de la liste.
- Possède une fonction membre `append` . Cette fonction prend en paramètre un pointeur sur un `SickKoalaList` et ne renvoie rien. Le maillon passé en paramètre est ajouté à la fin de la liste chaînée.
- Possède une fonction membre `getFromName` qui prend en paramètre une `string` et renvoie le pointeur sur le premier `SickKoala` dont le nom correspond à la `string` passée en paramètre.
- Possède une fonction membre `remove` prenant en paramètre un pointeur sur `SickKoalaList` et supprime de la liste le `SickKoalaList` correspondant à ce pointeur.
Cette fonction membre renvoie le pointeur sur le premier maillon de la liste.
- Possède une fonction membre `removeFromName` qui prend en paramètre une `string` . Cette fonction membre retire de la liste le premier `SickKoala` dont le nom correspond à la `string` passé en paramètre.
Cette fonction membre renvoie le pointeur sur le premier maillon de la liste.
- Possède une fonction membre `dump` ne prenant aucun paramètre et ne renvoie rien. Cette fonction membre affiche le nom de tous les `SickKoalas` contenus dans l'ordre de parcours de la liste (begin → end) :

```
1 Liste des patients : [nom1], [nom2], ..., [nomX].
```

Dans le cas où l'élément est absent, le nom affiché sera `[NULL]` .

- Possède une fonction membre `getContent` ne prenant pas de paramètre qui renvoie le pointeur sur l'élément contenu dans le maillon actuel (`SickKoalaList`).
- Possède une fonction membre `getNext` ne prenant pas de paramètre qui renvoie le pointeur sur le maillon suivant. S'il n'y en a pas, elle renvoie 0 (`NULL`)

- La classe `KoalaNurseList` :

- Lors de la construction, prend un pointeur sur une `KoalaNurse` . Ce pointeur peut être `NULL` .
- Possède une fonction membre `isEnd` . Cette fonction membre ne prend pas de paramètres et renvoie un booléen valant `true` si le `KoalaNurseList` est le

dernier maillon de la liste.

- Possède une fonction membre `append` . Cette fonction prend en paramètre un pointeur sur un `KoalaNurseList` et ne renvoie rien. Le maillon passé en paramètre est ajouté à la fin de la liste chaînée.
- Possède une fonction membre `getFromID` qui prend en paramètre un `int` et renvoie le pointeur sur le premier `KoalaNurse` dont le numéro correspond à la `int` passé en paramètre.
- Possède une fonction membre `remove` prenant en paramètre un pointeur sur `KoalaNurseList` et supprime de la liste le `KoalaNurseList` correspondant à ce pointeur.
Cette fonction membre renvoie le pointeur sur le premier maillon de la liste.
- Possède une fonction membre `removeFromID` qui prend en paramètre un `entier` . Cette fonction membre retire de la liste le premier `KoalaNurse` dont l'ID correspond à l'entier passé en paramètre.
Cette fonction membre renvoie le pointeur sur le premier maillon de la liste.
- Possède une fonction membre `dump` ne prenant aucun paramètre et ne renvoie rien. Cette fonction membre affiche le nom de tous les `KoalaNurse` contenus dans l'ordre de parcours de la liste (begin→ end) :

```
1 Liste des infirmieres : [nom1], [nom2], ..., [nomX].
```

Dans le cas où l'élément est absent, le nom affiché sera `[NULL]` .

- La classe `KoalaDoctorList` :

- Lors de la construction, prend un pointeur sur une `KoalaDoctor` . Ce pointeur peut être `NULL` .
- Possède une fonction membre `isEnd` . Cette fonction membre ne prend pas de paramètres et renvoie un booléen valant `true` si le `KoalaDoctorList` est le dernier maillon de la liste.
- Possède une fonction membre `append` . Cette fonction prend en paramètre un pointeur sur un `KoalaDoctorList` et ne renvoie rien. Le maillon passé en paramètre est ajouté à la fin de la liste chaînée.
- Possède une fonction membre `getFromName` qui prend en paramètre une `string` et renvoie le premier `KoalaDoctor` dont le nom correspond à la `string` passé en paramètre.
- Possède une fonction membre `remove` prenant en paramètre un pointeur sur `KoalaDoctorList` et supprime de la liste le `KoalaDoctorList` correspondant à ce pointeur. Cette fonction membre renvoie le pointeur sur le premier maillon de la liste.


- Possède une fonction membre `removeFromName` qui prend en paramètre une `string` . Cette fonction membre retire de la liste le premier `KoalaDoctor` dont le nom correspond à la `string` passé en paramètre. Cette fonction membre renvoie le pointeur sur le premier maillon de la liste.
- Possède une fonction membre `dump` ne prenant aucun paramètre et ne renvoie rien. Cette fonction membre affiche le nom de tous les `KoalaDoctors` contenus dans l'ordre de parcours de la liste (`begin -> end`) :

```
1  Liste des medecins : [nom1], [nom2], ..., [nomX].
```

Dans le cas où l'élément est absent, le nom affiché sera `[NULL]` .

Chapitre VIII

Exercice 6

	Exercice : 06	points : 4
L'Hopital		
Répertoire de rendu: (DÉPOT SVN - piscine_cpp_d06-promo-login_x)/ex06		
Compilateur : g++	Flags de compilation: -W -Wall -Werror	
Makefile : Non	Règles : n/a	
Fichiers a rendre : koalanurse.h, koalanurse.cpp, sickkoala.h, sickkoala.cpp, koaladoctor.h, koaladoctor.cpp, sickkoalalist.h, sickkoalalist.cpp, koalanurselist.h, koalanurselist.cpp, koaladoctorlist.h, koaladoctorlist.cpp, hopital.h, hopital.cpp		
Remarques : n/a		
Fonctions Interdites : *alloc, free, *printf, open, fopen - mot-clé using		

Nous pouvons gérer plusieurs patients, infirmières et docteurs. Nous pouvons donc avancer et gérer un hopital!!

Vous allez donc nous le coder, mais cette fois-ci, plus d'assistanat!

Vous allez donc devoir déduire les fonctions membres de l'hopital à partir du main de test ci-dessous.

L'hopital doit faire tourner un peu le travail entre les nurses et les medecins.

Pour cet exercice, il est possible que vous ayez des modifications à faire sur vos classes, c'est vous qui gérez, du moment que les modifications respectent au moins les sujets précédents!

Voici un main de test avec la sortie attendue :

```
1 #include <iostream>
2 #include <string>
3 #include <cstdlib>
4
5 #include "sickkoala.h"
6 #include "koalanurse.h"
7 #include "koaladoctor.h"
8 #include "sickkoalalist.h"
9 #include "koalanurselist.h"
10 #include "koaladoctorlist.h"
11 #include "hopital.h"
12
13 int main()
14 {
15     srandom(42);
16
17     KoalaDoctor cox("Cox");
18     KoalaDoctor house("House");
19     KoalaDoctor tired("Boudur-Oulot");
20     KoalaDoctorList doc1(&cox);
21     KoalaDoctorList doc2(&house);
22     KoalaDoctorList doc3(&tired);
23
24     KoalaNurse a(1);
25     KoalaNurse b(2);
26     KoalaNurseList nurse1(&a);
27     KoalaNurseList nurse2(&b);
28
29     SickKoala cancer("Ganepar");
30     SickKoala gangrene("Scarface");
31     SickKoala rougeole("RedFace");
32     SickKoala variole("Varia");
33     SickKoala fracture("Falter");
34     SickKoalaList sick1(&cancer);
35     SickKoalaList sick2(&gangrene);
36     SickKoalaList sick3(&rougeole);
37     SickKoalaList sick4(&variole);
38     SickKoalaList sick5(&fracture);
39
40     {
41         Hospital stAnne;
42
43         stAnne.addDoctor(&doc1);
44         stAnne.addDoctor(&doc2);
45         stAnne.addDoctor(&doc3);
46         stAnne.addSick(&sick1);
47         stAnne.addSick(&sick2);
48         stAnne.addSick(&sick3);
49         stAnne.addSick(&sick4);
```

```
50     stAnne.addSick(&sick5);
51     stAnne.addNurse(&nurse1);
52     stAnne.addNurse(&nurse2);
53
54     stAnne.addSick(&sick4);
55
56     stAnne.run();
57 }
58
59 if (nurse1.isEnd() && sick1.isEnd() && doc1.isEnd())
60     std::cout << "Lists cleaned up." << std::endl;
61 else
62     std::cerr << "You fail ! Boo !" << std::endl;
63
64 return (0);
65 }
```

Sortie attendue :

```
1 Sortie attendue :
2 Dr.Cox: Je suis le Dr.Cox ! Comment Kreoggez-vous ?
3 Dr.House: Je suis le Dr.House ! Comment Kreoggez-vous ?
4 Dr.Boudur-Oulot: Je suis le Dr.Boudur-Oulot ! Comment Kreoggez-vous ?
5 [HOSPITAL] Doctor Cox just arrived !
6 Dr.Cox: Je commence le travail !
7 [HOSPITAL] Doctor House just arrived !
8 Dr.House: Je commence le travail !
9 [HOSPITAL] Doctor Boudur-Oulot just arrived !
10 Dr.Boudur-Oulot: Je commence le travail !
11 [HOSPITAL] Patient Ganepar just arrived !
12 [HOSPITAL] Patient Scarface just arrived !
13 [HOSPITAL] Patient RedFace just arrived !
14 [HOSPITAL] Patient Varia just arrived !
15 [HOSPITAL] Patient Falter just arrived !
16 [HOSPITAL] Nurse 1 just arrived !
17 Nurse 1: Je commence le travail !
18 [HOSPITAL] Nurse 2 just arrived !
19 Nurse 2: Je commence le travail !
20 [HOSPITAL] Debut du travail avec :
21 Liste des medecins : Cox, House, Boudur-Oulot.
22 Liste des infirmieres : 1, 2.
23 Liste des patients : Ganepar, Scarface, RedFace, Varia, Falter.
24
25 Dr.Cox: Alors, qu'est-ce qui vous goerk, Mr.Ganepar ?
26 Mr.Ganepar: Gooeeeeerrk !! :'(
27 Nurse 1: Kreog ! Il faut donner un Buronzand a Mr.Ganepar !
28 Mr.Ganepar: Et la fatigue a fait son temps !
29 Dr.House: Alors, qu'est-ce qui vous goerk, Mr.Scarface ?
30 Mr.Scarface: Gooeeeeerrk !! :'(
31 Nurse 2: Kreog ! Il faut donner un mars a Mr.Scarface !
32 Mr.Scarface: Mars, et ca Kreog !
33 Dr.Boudur-Oulot: Alors, qu'est-ce qui vous goerk, Mr.RedFace ?
34 Mr.RedFace: Gooeeeeerrk !! :'(
35 Nurse 1: Kreog ! Il faut donner un Buronzand a Mr.RedFace !
36 Mr.RedFace: Et la fatigue a fait son temps !
37 Dr.Cox: Alors, qu'est-ce qui vous goerk, Mr.Varia ?
38 Mr.Varia: Gooeeeeerrk !! :'(
39 Nurse 2: Kreog ! Il faut donner un Buronzand a Mr.Varia !
40 Mr.Varia: Et la fatigue a fait son temps !
41 Dr.House: Alors, qu'est-ce qui vous goerk, Mr.Falter ?
42 Mr.Falter: Gooeeeeerrk !! :'(
43 Nurse 1: Kreog ! Il faut donner un Viagra a Mr.Falter !
44 Mr.Falter: Goerk !
45 Nurse 1: Je rentre dans ma foret d'eucalyptus !
46 Nurse 2: Je rentre dans ma foret d'eucalyptus !
47 Dr.Cox: Je rentre dans ma foret d'eucalyptus !
48 Dr.House: Je rentre dans ma foret d'eucalyptus !
49 Dr.Boudur-Oulot: Je rentre dans ma foret d'eucalyptus !
```

```
50 Lists cleaned up.  
51 Mr.Falter: Kreooogg !! Je suis gueriiii !  
52 Mr.Varia: Kreooogg !! Je suis gueriiii !  
53 Mr.RedFace: Kreooogg !! Je suis gueriiii !  
54 Mr.Scarface: Kreooogg !! Je suis gueriiii !  
55 Mr.Ganepar: Kreooogg !! Je suis gueriiii !  
56 Nurse 2: Enfin un peu de repos !  
57 Nurse 1: Enfin un peu de repos !
```