Université Paris Diderot
Licence 3 Informatique

Conduite de Projet
Année 2017–2018

Examen

3 mai 2018

Instructions Justifiez vos réponses. Les documents sont interdits. Tout dispositif électronique et/ou de communication est interdit. La durée de l'examen est de 3 heures.

Exercice 1 [Processus de développement, 3 points]

a) Décrivez brièvement trois processus de développement de votre choix.

b) Décrivez brièvement trois bonnes pratiques introduite par la méthode agile *Extreme Programming* (XP).

Exercice 2 [Conception, 5 points] On souhaite réaliser un moteur de recherche pour le Web nommé Zoogle. Ce logiciel doit fournir un robot d'indexation qui, à partir d'une liste donnée de pages web (accessibles via différents protocoles, comme `http`, `https`, `ftp`, etc.), télécharge ces pages et les analyse (en supportant des formats différents, comme HTML, PDF, texte simple, documents bureautiques, etc.). L'analyse des pages doit : d'une part, trouver des nouveaux liens à suivre pour permettre une indexation récursive du Web ; d'autre part, extraire les mots présents dans chaque page pour former un index permettant d'effectuer des recherches textuelles parmi toutes les pages indexées. L'interface utilisateur de Zoogle sera un site Web avec un champ de recherche qui permet d'entrer des mots-clés, et qui retournera une liste de pages Web contenant ces mots-clés. Les pages les plus pertinentes devront être retournées en premier.

a) Proposez une architecture logiciel pour Zoogle, sous la forme d'une liste de modules. Pour chaque module, donnez son nom et une courte description de ses responsabilités.

b) Présentez l'architecture que vous avez conçue pour Zoogle sous la forme d'un graphe orienté ayant comme noeuds les modules, et comme arcs les dépendances entre ces modules (*rappel* : un module A dépend d'un module B si l'implémentation de A se sert de l'implémentation de B, par exemple via un appel de fonction).

Veillez à bien prendre en compte la modularité, la maintenabilité, et la possibilité des extensions futures dans vos choix d'architecture.

Exercice 3 [*Make*, 4 points]

a) Quelle est l'utilité de l'outil Make et des Makefiles ? Quels sont les avantages et les inconvénients de Make par rapport à une gestion plus manuelle du processus de compilation ?

b) On considère le `Makefile` suivant :

```
edit : main.o kbd.o command.o display.o insert.o search.o files.o utils.o
	cc -o edit main.o kbd.o command.o display.o \
	           insert.o search.o files.o utils.o
```

```
main.o : main.c defs.h
        cc -c main.c
kbd.o : kbd.c defs.h command.h
        cc -c kbd.c
command.o : command.c defs.h command.h
        cc -c command.c
display.o : display.c defs.h buffer.h
        cc -c display.c
insert.o : insert.c defs.h buffer.h
        cc -c insert.c
search.o : search.c defs.h buffer.h
        cc -c search.c
files.o : files.c defs.h buffer.h command.h
        cc -c files.c
utils.o : utils.c defs.h
        cc -c utils.c
clean :
        rm edit main.o kbd.o command.o display.o \
           insert.o search.o files.o utils.o
```

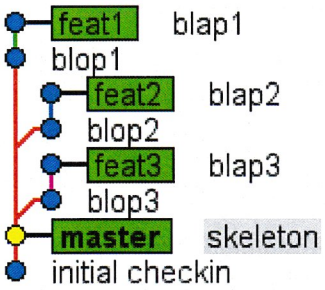
Donnez le diagramme des dépendances correspondant à ce `Makefile`, *i.e.*, un graphe dont chaque noeud correspond à une cible et a comme fils ses prérequis.

c) Si tous les fichiers `*.c` et `*.h` existent, mais si aucun des fichiers `*.o` (ou `edit`) n'est encore présent sur le disque, que se passera-t-il lors de l'exécution de `make` ? Donnez, dans l'ordre exact, la liste des commandes que `make` exécutera successivement.

d) Si, après une première compilation complète, vous modifiez `buffer.h` puis exécutez à nouveau `make`, que se passera-t-il ? Donnez, dans l'ordre exact, la liste des commandes que `make` exécutera successivement.

Exercice 4 [Git, 3 points]

Écrivez une liste de commandes `git` produisant un dépôt Git dont l'historique ressemble le plus possible à celui représenté dans la figure ci-contre—chaque noeud correspond à un *commit*, les textes sur la droite à des messages de *commit*, et les étiquettes vertes à des noms de branches. Le contenu du dépôt n'est pas important (vous pouvez par exemple considérer qu'il ne contient qu'un seul fichier `toto.txt`), mais les *commits*, branches et *merges* doivent correspondre à ceux de la figure.



Exercice 5 [Tests, 5 points]

a) Écrivez, dans le format d'un fichier `.h`, l'interface C minimale d'un type de données abstrait *file double d'entiers* (ou *dequeue*), qui permet d'enfiler et de défiler au début et à la fin d'une file, en $O(1)$ dans tous les cas.

b) Proposez au moins 10 tests unitaires pour ce type de données, en donnant une courte description (= 1 phrase) de chaque test. Par exemple : « si on ajoute 42 au début d'une file existante, le prochain élément défilé au début sera 42 ».

c) Donnez l'implémentation, avec la librairie de test Check, d'au moins 5 tests parmi les tests unitaires proposés.