Université Paris Diderot
Licence 3 Informatique

Conduite de Projet
Année 2016–2017

Examen

17 mai 2017

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, 4 points]

a) Qu'est-ce qu'un processus de développement logiciel ? Donnez une définition générale, ainsi que la liste des activités que l'on retrouve dans la plupart des processus de développement logiciel.

b) Décrivez brièvement les deux processus de développement suivants : en cascade ; itératif (ou *évolutif*). Quels sont les avantages et les inconvénients de chacun d'entre eux ?

Exercice 2 [Conception, 5 points] On souhaite réaliser un logiciel d'indentation automatique nommé WizDent. Ce logiciel prendra en entrée un répertoire contenant un ou plusieurs fichiers source, et produira en sortie une copie du répertoire dans lequel chaque fichier source aura été indenté correctement selon des règles prédéfinies. Le logiciel doit au minimum supporter les fichiers C (extension `.c` ou `.h`) et Python (extension `.py`), mais ses développements futurs pourront supporter d'autres langages de programmation.

a) Proposez une architecture logiciel pour WizDent, 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 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).

Exercice 3 [*Make*, 3 points]

a) Dans le cadre de *Make*, expliquez les notions suivantes : règle ; cible ; prérequis ; commande.

b) On considère le `Makefile` suivant :

```
foo: main.o high.o low.o utils.o
	gcc -o foo main.o high.o low.o utils.o
main.o: main.c high.h utils.h
	gcc -c -o main.o main.c high.h utils.h
high.o: high.c low.h utils.h
	gcc -c -o high.o high.c low.h utils.h
low.o: low.c utils.h
	gcc -c -o low.o low.c utils.h
utils.o: utils.c utils.h
	gcc -c -o utils.o utils.c utils.h
clean:
	-rm -f foo main.o high.o low.o utils.o
```

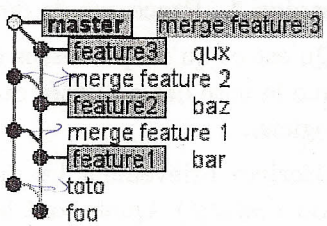
Donnez le diagramme de dépendances correspondant à ce `Makefile` (*ie.* un graphe dont chaque noeud correspond à une cible et a comme fils ses prérequis)

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

d) Si, après une première compilation complète, vous modifiez `utils.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) Expliquez l'approche *Test Driven Development* (développement piloté par les tests) appliquée au développement de logiciel. Quels sont les avantages et les inconvénients de cette approche ?

b) Écrivez, dans le format d'un fichier `.h`, l'interface C minimale d'un type de données abstrait *graphe orienté*, où les sommets seront étiquetés par des chaînes de caractères. 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 le sommet "foo" au graphe vide, on obtient un graphe avec 1 sommet et 0 arcs »).

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