Université Paris Diderot
Licence 3 Informatique

Conduite de Projet
Année 2016–2017

Examen

27 juin 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 [Conception (théorie), 4 points]

a) Expliquez les notions de cohésion et interdépendance entres modules d'un logiciel. Quel critère qualitatif empirique on utilise pour juger de la qualité de la modularisation d'un logiciel lors de sa conception ?

b) Donnez une description du patron d'architecture « filtres et tuyaux ».

Exercice 2 [Conception (exercice), 5 points] On souhaite réaliser un éditeur de texte pour développeurs, nommé Absence.

Au delà de fonctionnalités classiques d'édition de texte (insertion et suppression de caractères, copier-coller, etc.) Absence doit être capable de gérer les activités typiques de la programmation. Une liste non exhaustive de ces activités comprend : la coloration syntaxique ; l'indentation automatique ; la complétion automatique de membres de structures, classes, etc. ; la compilation, en appelant un outil externe ; le débogage, à travers l'exécution pas à pas. Le comportement exact de ces activités dépend du langage de programmation utilisé.

a) Proposez une architecture logiciel pour Absence, sous la forme d'une liste de modules, en respectant les meilleures pratiques de réutilisation du code et d'extensibilité. 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 un petit projet de développement en C formé par les fichiers montrés dans la sortie de `ls` suivante :

```
$ ls
buffer.c            ui-curses.h
buffer.h            ui-gtk.c
main.c              ui-gtk.h
search.c            ui.c
search.h            ui.h
ui-curses.c
```

En termes de dépendances entres modules :

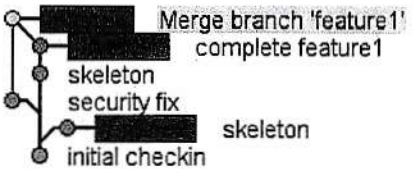
— `ui-curses` et `ui-gtk` dépendent, chacun, de `ui`, `buffer`, `search` ;
— `search` dépend de `buffer`
— `main` dépend de `ui-gtk` et `ui-curses`

Écrivez le Makefile plus court et moins répétitif possible pour ce projet. Le Makefile doit permettre de compiler et *linker* tous les modules ensemble, pour obtenir un exécutable final nommé `editor`. Tous les modules doivent être compilés avec le compilateur `gcc` et les options `-Wall -Werror`.

c) Donnez le diagramme de dépendances (i.e., un graphe dont chaque noeud correspond à une cible et a comme fils ses prérequis) correspondant au `Makefile` de la question précédente.

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 la différence entre tests en boite blanche et tests en boite noir.
Quand les uns sont plus appropriés et utiles que les autres ?

b) Écrivez, dans le format d'un fichier `.h`, l'interface C minimale d'un type de données abstrait *liste d'entiers*.
Proposez au moins 10 tests unitaires pour ce type de données, en donnant une courte description (= 1 phrase) de chacun. Par exemple : « si on ajoute l'entier 17 en tête à la liste qui contient seulement l'entier 19, on obtient une liste de longueur 2, avec 17 comme premier élément et 19 comme dernier ».

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