EXAMEN DE GÉNIE LOGICIEL
Cours B5

Session I 13 mars 1999
Arcueil

La question de cours et l'exercice doivent impérativement être traités sur deux copies distinctes.

Prenez le temps de bien lire l'énoncé.

Tous documents et calculettes sont autorisés.

Barème :

Question de cours 12 points

Q1 4
Q2 3
Q3 3
Q4 2

Bonus

Q5 2
Q6 2

Exercice 8 points

I Partie syntaxique

Q1.1 4
Q1.2 4

Bonus

II Partie sémantique

Q 4

TABLES OU DICTIONNAIRES

Les tables ou dictionnaires, encore appelées tables de symboles en compilation, forment un type de données abstrait constitué d'un ensemble de doublets (clé, valeur), sur lequel sont définies les opérations :

créer une table vide, insérer un doublet dans la table, y supprimer une clé, rechercher l'information associée à une clé, modifier cette information.

Dans les tables arborescentes, il n'y a aucun rapport entre la clé d'un élément et sa localisation (son "adresse") dans la table. En vue de réduire la complexité *moyenne* des primitives d'accès, sinon à une constante, au moins à un très petit nombre de comparaisons de clés, on envisage d'organiser la table à l'aide d'une structure permettant d'établir un lien entre les deux entités "clé" et "adresse" : on parle alors d'adressage *calculé*, ou fonctionnel.

Soit K le domaine des clés. On définit une fonction d'adressage h de K dans l'espace des adresses de la table :

$$\forall c \in K \xrightarrow{\ h\ } a = h(c)$$

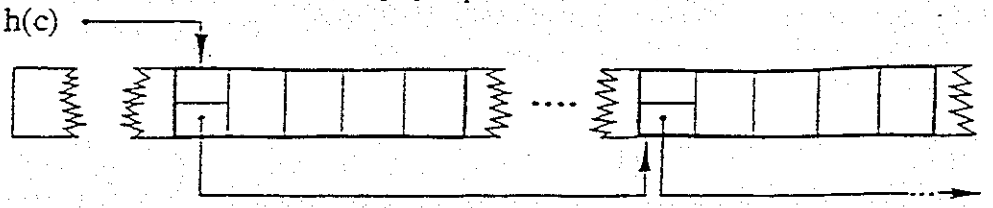
Pour que ce procédé soit opérationnel, la table doit être adressable, donc supporter l'accès direct vrai (en temps constant), avec un espace d'adresses A connu a priori. C'est le cas d'un vecteur, ou d'un fichier à accès DIRECT.

Il arrivera qu'à deux clés distinctes c et c' corresponde une même adresse a dans la table : $h(c) = h(c') = a$.
On dit que la fonction h réalise alors une *collision*, et que les deux clés c et c' sont *synonymes* relativement à h.

Le hachage ouvert

On l'appelle ainsi parce qu'on "ouvre" la table en associant à chaque adresse une liste linéaire de clés synonymes, rangée hors de la table principale. Cette méthode est particulièrement prisée lorsque la table réside sur disque magnétique.

en vue de minimiser le nombre d'accès disque, on gère les listes de synonymes de manière mixte "contiguë-chaînée", sous forme d'une liste chaînée de *blocs* (ou enregistrements "physiques") :

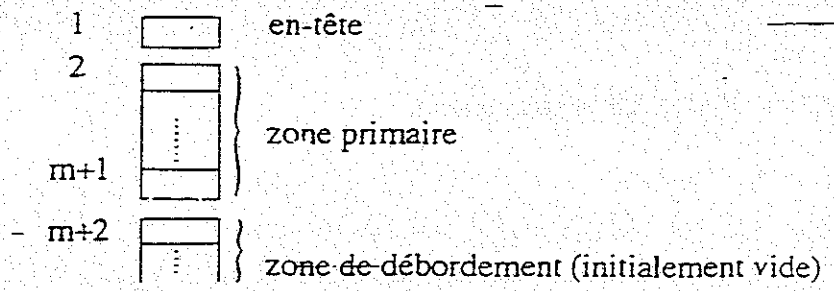


La taille b des blocs, en nombre d'enregistrements logiques (clé, valeur) (b = 4 dans le schéma ci-dessus), dépend du rapport entre la taille optimale des blocs physiques, et la longueur d'un enregistrement : elle est donc laissée au choix de l'utilisateur (c'est la constante générique Facteur_de_Blocage du module ci-après). Ainsi, chaque accès disque permet d'obtenir jusqu'à b clés synonymes.

en toute rigueur, la suppression d'une clé dans un bloc devrait être compensée s'il y a lieu par l'apport d'une autre clé extraite du bloc successeur, et ainsi de suite. Les décalages, trop coûteux (une lecture-écriture pour chaque bloc), sont souvent omis : la suppression laisse alors un bloc incomplet.

Cela pourrait allonger indûment les listes de blocs synonymes : aussi, pour compenser, opère-t-on l'insertion dans le premier bloc incomplet rencontré. A son tour, ce procédé impose d'abandonner l'ordre entre clés au sein d'une liste de synonymes.

Globalement, le fichier est divisé en deux zones contiguës : une zone primaire de taille m (les premiers blocs des m listes de synonymes), et une zone de débordement illimitée pour les blocs supplémentaires. On peut encore introduire un article d'en-tête pour mémoriser des informations générales sur le fichier :



A la création de la table, le fichier est automatiquement formaté selon ce schéma.

On désire procéder au test unitaire, par la méthode des couvertures, de la procédure VAL-CLE ci-jointe, incluse dans le module HACHAGE-EXTERNE. Les spécifications sont indiquées en commentaire. On fournit par ailleurs l'essentiel des primitives exportées, qui utilisent toutes VAL-CLE.

I PARTIE SYNTAXIQUE

Q1.1 -Etablir le graphe de commande de la procédure VAL-CLE.

Q1.2 -Calculer son nombre cyclomatique. Comment peut-on qualifier la complexité de cette procédure?

N.B : on assimilera les instructions RAISE à des RETURN simples. Par ailleurs, on ignorera la séquence EXCEPTION des lignes 148 et 149.

II PARTIE SEMANTIQUE

Q En déduire une politique de test permettant une couverture exhaustive des arcs (ou liens). Préciser la sémantique de chaque jeux de données utilisé pour le test.

Comment procèderiez-vous si l'on vous demandait maintenant une couverture 100% des nœuds (ou sommets)?

P.J : Le texte de la procédure Ada VAL-CLE et son contexte.

```
--*********************************************************
--*   T A B L E   G E R E E   P A R   H A C H A G E   *
--*    Version implementee sur Fichier a Acces Direct.    *
--*********************************************************

generic.                              -- parametres de compilation.
  type VALEUR is private;
  type  CLEF  is private;
  -- N.B.: L'egalite sur les CLEFs est l'op. predefinie.

  with function NUMERIC (c:CLEF)return NATURAL;    -- f. de hachage.

  F_TABLE: string;       -- nom "systeme" du fichier-table.
  FACTEUR_DE_BLOCAGE: positive := 1;              -- nb. cles/bloc.

  -- Pour la creation de la table seulement:

  TAILLE_ZONE_PRIMAIRE:positive:=23;              -- nb. premier.

package HACHAGE_EXTERNE is

  CLEF_ABSENTE, CLEF_PRESENTE:exception;
  ERR_ACCES: exception;          -- erreur lors d'un acces.


  -- PRIMITIVE D'INSERTION:
     -- Insere dans la table le doublet (CLE, VAL);
     -- Leve CLEF_PRESENTE si la CLE y figurait deja.
  procedure INSERER (VAL:VALEUR; CLE:CLEF);



  -- PRIMITIVE DE SUPPRESSION:
     -- Supprime l'element de CLE donnee dans la table.
     -- Leve CLEF_ABSENTE si cette cle n'y figurait pas.
  procedure SUPPRIMER (CLE: CLEF);



  -- PRIMITIVE DE CONSULTATION:
     -- Recupere la valeur associee a la CLE dans la table.
     -- Leve CLEF_ABSENTE si la cle n'y figure pas.
  function RECHERCHE (CLE: CLEF) return VALEUR;



  -- PRIMITIVE DE MODIFICATION:
     -- Associe la nouvelle valeur NEW_VAL a la CLE dans la table.
     -- Leve CLEF_ABSENTE si la cle n'y figure pas.
  procedure MODIFIER (CLE:CLEF; NEW_VAL:Valeur);

end HACHAGE_EXTERNE;
```

```
--*********************************************************
--*    IMPLEMENTATION D'UN OBJET TABLE GERE PAR HACHAGE   *
--*            Version Fichier a Acces Direct.            *
--*********************************************************

with DIRECT_IO;
package body HACHAGE_EXTERNE is

  type ELEMENT is record
        VAL: VALEUR;
        KEY: CLEF;
  end record;
  subtype ARTICLE is positive range 1..FACTEUR_DE_BLOCAGE;
  type SYNONYMES is array (Article) of element;

  NIL: constant :=0;
```



```
type BLOCK is record
        long_liste : natural:=0;     -- d'Elts (ou T_Z_P).
        suivant : natural:=NIL;      -- chainage des blocs.
        liste_syno : Synonymes;
      end record;

entete, bloc_donnees: block;
TAILLE: natural renames entete.long_liste;     -- de la zone primaire.

NB_ARTICLES: natural renames bloc_donnees.long_liste;

package dir_io is new DIRECT_IO (element_type => block);
use dir_io;

FDAT: FILE_TYPE;
subtype INDEX is positive_Count range 2..COUNT'last;

-- Variables globales:

blk_cle: Positive_Count;                       -- pour VAL_CLE.
art_cle: positive range 1..Facteur_de_Blocage+1;


              -- *************
              -- * SSP LOCAUX *
              -- *************


-- Fonction de dispersion dans la table:
-- (avec decalage a cause du bloc Entete de no. 1).

function HACH_CODE (CLE: clef) return INDEX is
begin
  return Index (NUMERIC (cle) mod Taille + 2);
end;


-----------------------------------------------------------------
-- Positionne la "tete de lecture-ecriture" sur l'article du bloc
-- contenant la CLE (ou sur la 1ere "case" disponible sinon),
-- et calcule les coordonnees (Blk_cle, Art_cle)
-- de l'enregistrement associe.
-- Leve CLEF_ABSENTE le cas echeant.

procedure VAL_CLE (CLE: CLEF) is
          -------

  bloc_insert:Block;                   -- de sauvegarde (Insertion).
  bloc_num: Positive_Count:=1;         -- val. init. conventionnelle.

begin
  blk_cle:= Hach_Code (cle);

  begin                                -- bloc (=Sequence).

    loop
      READ (Fdat,bloc_donnees, blk_cle);

      -- Recherche si CLE figure dans le bloc lu:

      for i in 1..Nb_Articles loop
          if Bloc_Donnees.Liste_Syno(i).key = cle then
            art_cle:=i;
            return;
          end if;
      end loop;

      EXIT when Bloc_Donnees.Suivant = NIL;
```

```
      if bloc_num=1 and Nb_Articles < FACTEUR_DE_BLOCAGE then
         bloc_num:=blk_cle;                    -- pour l'insertion.
         art_cle:=Nb_Articles +1;
         bloc_insert:=Bloc_donnees;            -- sauvegarde.
      end if;
      blk_cle:= Index(Bloc_Donnees.Suivant);
    end loop;

  exception
    when others => raise Err_Acces;
  end;                                 -- (fin du bloc).

  -- En mode normal, l'execution ne se poursuit ici que pour l'insertion.
  -- Pour les autres primitives, ill s'agit d'un cas d'erreur.

  if bloc_num > 1 then                 -- num. du bloc ou inserer.
     Bloc_Donnees:=Bloc_Insert;        -- restauration.
     blk_cle:=bloc_num;
  else
     art_cle:=Nb_Articles + 1;
  end if;

  raise CLEF_ABSENTE;
end VAL_CLE;  ------------------------------------------

              -- ****************
              -- * SSP PUBLICS *
              -- ****************


procedure INSERER (VAL:VALEUR; CLE:CLEF) is   -- PRIMITIVE D'INSERTION.
begin
    VAL_CLE (cle);
    raise CLEF_PRESENTE;
exception
    when CLEF_ABSENTE => begin                -- cas normal!!!
       if art_cle > FACTEUR_DE_BLOCAGE then
          Bloc_Donnees.Suivant:=Natural(SIZE (Fdat) + 1);
          WRITE (Fdat, Bloc_Donnees, To => blk_cle);
          blk_cle:= Index(Bloc_Donnees.Suivant);
          art_cle:=1;
          Nb_Articles:=0;              -- du nouveau bloc.
          Bloc_Donnees.Suivant:=NIL;
       end if;
       Bloc_Donnees.Liste_Syno(art_cle):=(val,cle);
       Nb_Articles:=Nb_Articles+1;
       WRITE (Fdat, Bloc_Donnees, To => blk_cle);
    end;
end Inserer;


procedure SUPPRIMER (CLE: CLEF) is       -- PRIMITIVE DE SUPPRESSION.
begin
  VAL_CLE (cle);
  if art_cle < Nb_Articles then
     Bloc_Donnees.Liste_Syno(art_cle):=
                  Bloc_Donnees.Liste_Syno(Nb_Articles);
  end if;
  Nb_Articles:=Nb_Articles-1;
  WRITE (Fdat, Bloc_Donnees, To => blk_cle);
end;


-- PRIMITIVE DE CONSULTATION:

function RECHERCHE (CLE:CLEF) return VALEUR is
begin
  VAL_CLE (cle);
```



```
    return Bloc_Donnees.Liste_Syno(art_cle).Val;
  end;


  -- PRIMITIVE DE MODIFICATION:

  procedure MODIFIER (CLE:CLEF; NEW_VAL:Valeur) is
  begin
      VAL_CLE (cle);
      Bloc_Donnees.Liste_Syno(art_cle).Val:= NEW_VAL;
      WRITE (Fdat, Bloc_Donnees, To => blk_cle);
  end;

end HACHAGE_EXTERNE;
```

EXERCICE DE TEST

On souhaite écrire une procédure générale pour saisir, de manière interactive, une valeur (entière, par exemple) devant, le cas échéant, respecter une contrainte d'intervalle (fermé ou ouvert) de définition.

L'utilisateur lui transmet le message d'invite expliquant le sens de la valeur attendue et, de manière *optionnelle,* la u les bornes de son intervalle de définition.
La procédure sera une fonction renvoyant la valeur lue.
La fonction READ-INT ci-jointe réalise, le traitement spécifié.

On désire procéder au test unitaire, par la méthode des couvertures, e cette fonction (READ-INT). Les spécifications sont précisées dans les commentaires.

I. PARTIE SYNTAXIQUE

3 points Q1.1 -Etablir le graphe de commande de la fonction READ-INT
1 point Q1.2 -Calculer son nombre cyclomatique. Quel sera le nombre maximum de chemins nécessaires pour effectuer la couverture totale des arcs du graphe ?

II. PARTIE SEMANTIQUE

4 points Q2 Définir un ensemble de chemins permettant une couverture exhaustive des arcs (ou liens) du graphe de commande. Préciser avec soin la sémantique de chaque chemin choisi en fonction des paramètres d'entrée de READ-INT et de la valeur saisie au clavier (variable RESULTAT)

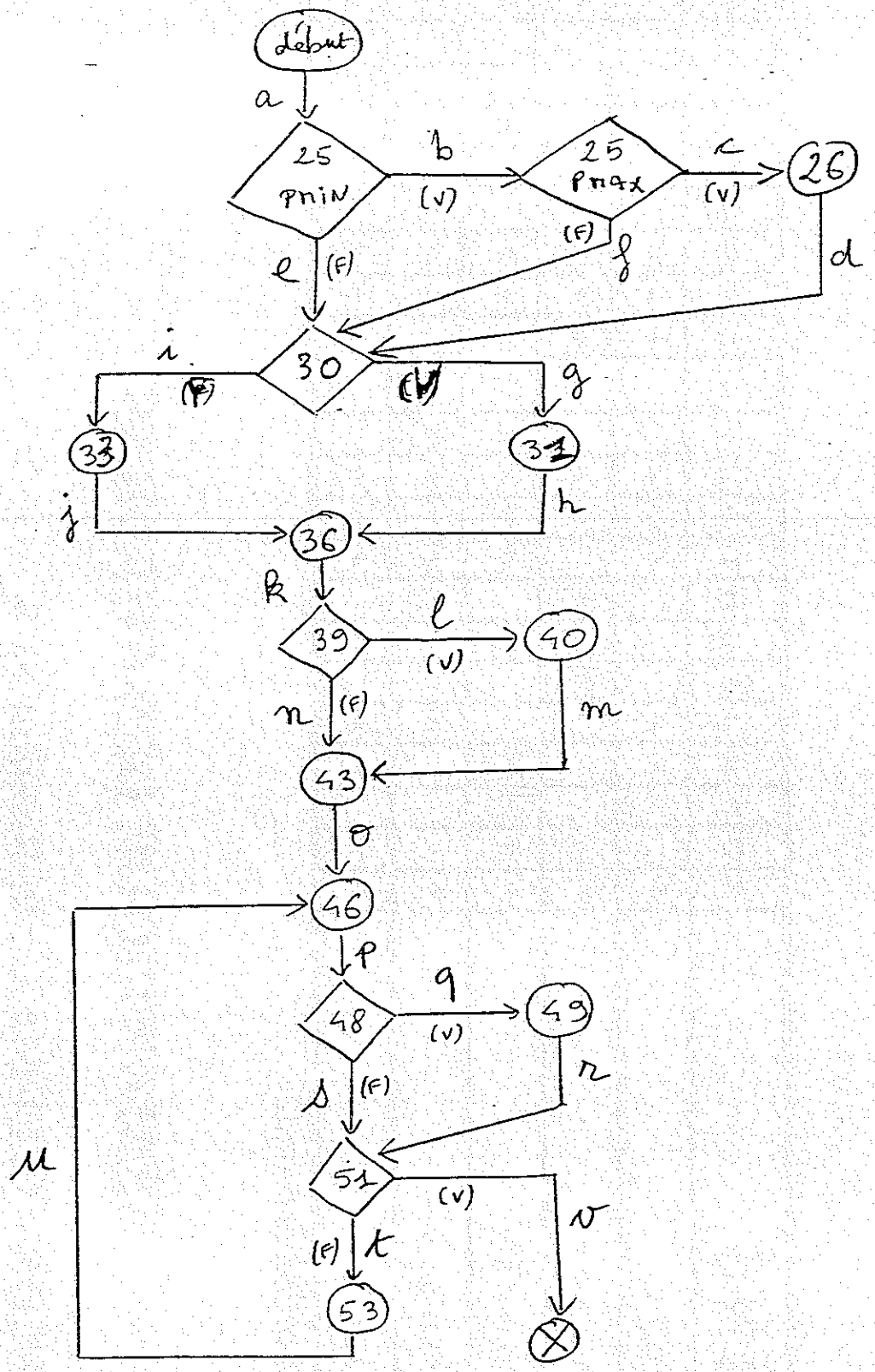
N.B : On utilisera impérativement la grille ci-jointe dans la réponse à cette question. Les nombres d'arcs et de chemins qui y figurent ne préjugent bien sûr en rien des nombres effectifs d'arcs de votre graphe ni de chemins nécessaires pour le recouvrir…

Que concluez-vous du nombre de chemins par rapport au nombre cyclomatique?

Comment procéderiez vous si l'on vous demandait maintenant une couverture totale des nœuds (ou sommets) du graphe?

```
1   -- Cette fonction affiche a l'ecran le MeSsaGe de l'utilisateur,
2   -- puis lit au clavier un nombre entier >= 0.
3   -- Le nombre lu est le cas echeant confronte a l'intervalle
4   -- [LUMIN, LUMAX], et la lecture iteree en cas d'erreur de saisie.
5
6   -- LUMIN >=0 ==> le nombre lu doit etre >= LUMIN.
7   -- LUMAX >=0 ==> le nombre lu doit etre <= LUMAX.
8   -- Les 2 parametres < 0 ==> aucune autre contrainte que: nb. >= 0.
9   -- Si les 2 param. sont >=0 mais LUMIN > LUMAX, ils sont IGNORES.
10
11 FUNCTION READ_INT (MSG: STRING; LUMIN, LUMAX: INTEGER := -1) RETURN Natural;
12
13 WITH Text_io; USE Text_io;
14
15 FUNCTION READ_INT (MSG: STRING; LUMIN, LUMAX: INTEGER := -1) RETURN Natural IS
16
17     PACKAGE Int_Io IS NEW INTEGER_IO (INTEGER); USE Int_Io;
18     Resultat, LUMINat: INTEGER;
19     PMIN, PMAX, OK: Boolean;
20
21 BEGIN    -- corps de laM-  fonction:
22
23     PMIN:= LUMIN >= 0; PMAX:= LUMAX >= 0;
24
25     IF (PMIN AND PMAX) THEN
26         PMIN := LUMIN <= LUMAX;
27         PMAX := PMIN;
28     END IF;
29
30     IF PMIN THEN          -- LUMINAT = valeur minimale du nombre lu.
31         LUMINAT:= LUMIN;
32     ELSE
33         LUMINAT:= 0;
34     END IF;
35
36     PUT ("*** L'entier demande doit etre >= ");
37     PUT (LUMINAT,1);
38
39     IF (PMAX) THEN
40         PUT (" et <= ");
41         PUT (LUMAX,1);
42     END IF;
43     NEW_LINE;
44     PUT (MSG & " ? ");
45     LOOP
46         GET (Resultat);
47         OK := Resultat >= LUMINAT;
48         IF PMAX THEN
49            OK := OK AND Resultat <= LUMAX;
50         END IF;
51         EXIT WHEN  OK;
52
53         PUT ("Contrainte(s) non satisfaite(s): recommencez. ? ");
54     END LOOP;
55     RETURN Resultat;
56  END READ_INT;
```

GRAPHE DE COMMANDE



Nb. cyclomatique = 1 + Nb. de décisions binaires simples

= 1 + 6 = 7

= A − N + 2.P

= 22 − 17 + 2.1 = 7

CHEMINS		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v
1	a e i j k n o p s v	*	.	.	.	*	.	.	.	*	*	*	.	.	*	*	*	.	.	*	.	.	*
2	a b f g h k n o p s v	*	(*)	.	.	.	(*)	(*)	(*)	.	.	*	.	.	*	*	*	.	.	*	.	.	*
3	a b c d i j k n o p s v	*	*	(*)	(*)	.	.	.	.	*	*	*	.	.	*	*	*	.	.	*	.	.	*
4	a e i j k l m o p q r v	*	.	.	.	*	.	.	.	*	*	*	(*)	(*)	.	*	*	(*)	(*)	.	.	.	*
5	a e i j k n o p s t u . . .	*	.	.	.	*	.	.	.	*	*	*	.	.	*	*	*	.	.	*	(*)	(*)	.

(*) ==> arcs nouvellement visités.

SÉMANTIQUE DES CHEMINS

1. **LUMIN < 0; LUMAX < 0; Resultat >= 0**
2. **LUMIN >= 0; LUMAX < 0; Resultat >= 0**
3. **LUMIN >= 0; LUMAX >= 0; LUMIN > LUMAX; Resultat >= 0**
4. **LUMIN < 0; LUMAX >= 0; 0 <= Resultat <= LUMAX**
5. **LUMIN < 0; LUMAX < 0; Resultat < 0, ...**

EXERCICE DE TEST

La fonction S-BINOMIALE ci-jointe simule le tirage pseudo-aléatoire d'une observation selon la loi de probabilité discrète dite binomiale de paramètres N et P (ses 2 paramètres d'entrée).

On désire procéder au test unitaire, par la méthode des couvertures, de cette fonction (S-BINOMIALE). Les spécifications exactes sont précisées dans les commentaires.

I. PARTIE SYNTAXIQUE

3 points Q1.1 -Etablir le graphe de commande de la fonction S-BINOMIALE

1 point Q1.2 -Calculer son nombre cyclomatique. Quel sera le nombre maximum de chemins nécessaires pour effectuer la couverture totale des arcs du graphe ?

II. PARTIE SEMANTIQUE

4 points Q2 Définir un ensemble de chemins permettant une couverture exhaustive des arcs (ou liens) du graphe de commande. Préciser avec soin la sémantique de chaque chemin choisi en fonction des paramètres d'entrée de S-BINOMIALE

N.B1 : Dans cette question on ne tiendra pas compte des lignes 51-53, le résultat d'un appel de la fonction RANDOM (du paquetage Sutil) étant par définition pseudo aléatoire, donc indépendant des données...

N.B2 : On utilisera impérativement la grille ci-jointe dans la réponse à cette question. Les nombres d'arcs et de chemins qui y figurent ne préjugent bien sûr en rien des nombres effectifs d'arcs de votre graphe ni de chemins nécessaires pour le recouvrir...

Que concluez-vous du nombre de chemins par rapport au nombre cyclomatique?

Comment procéderiez vous si l'on vous demandait maintenant une couverture totale des nœuds (ou sommets) du graphe?

P.J : le texte de la fonction S-BINOMIALE, plus la grille de réponse.

```
 1 with sutil; use sutil;          --> RANDOM, SQRT, GAUSS.
 2
 3 -- La fonction S_BINOMIALE simule une observation pseudo-aleatoire
 4 -- selon la loi binomiale B (n, p).
 5 -- Le resultat est un entier entre 0 et n.
 6 -- Methode utilisee: on effectue n tirages pseudo-independants selon
 7 -- la loi de Bernoulli B(1, p), et on en fait la somme.
 8
 9 -- Cas particuliers:
10
11 -- 1) L'approximation gaussienne asymptotique:
12 --       B (n,p) = N (n.p, sqrt(n.p.q))   (ou: q = 1 - p)
13 --       est utilisee si la variance n.p.q > 20
14
15 -- 2) Cas d'erreur sur les parametres d'entree N et P:
16
17 --       a) N = 0 ou P <= 0 : la fonction renvoie une masse de Dirac 0
18 --       b) P >= 1 : la fonction renvoie une masse de Dirac N
19
20 function S_BINOMIALE (N: natural; P: float) return NATURAL IS
21
22   XMOY, SIGMA2: FLOAT;  -- moyenne & variance de B (n, p)
23   SBINOM: natural;
24
25 BEGIN            -- corps de la fonction.
26
27         -- Controle des cas d'erreur:
28
29   IF N = 0 OR P <= 0.0 THEN
30         RETURN 0;
31   END IF;
32
33   IF P >= 1.0 THEN
34         RETURN N;
35   END IF;
36
37   -- Calculs normaux:
38
39   XMOY   := FLOAT(N) * P;                -- moyenne  de B (n, p)
40   SIGMA2 := XMOY * (1.0 - P);            -- variance de B (n, p)
41
42   IF SIGMA2 > 20.0 THEN          -- utiliser l'approximation normale.
43      LOOP                        -- obtenir une valeur entre 0  et N
44        SBINOM := INTEGER( GAUSS (Xmoy, SQRT (Sigma2)));
45        EXIT WHEN Sbinom >= 0 AND Sbinom <= N;
46      END LOOP;
47
48    ELSE                   -- somme de N aleas B (1, p) "independants"
49         SBINOM := 0;
50         FOR tirage IN 1..N LOOP
51                IF RANDOM < P THEN
52                      Sbinom := Sbinom + 1;
53                END IF;
54         END LOOP;
55     END IF;
56
57     RETURN SBINOM;
58
59 END S_BINOMIALE;
```

CHEMINS	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	....
1																											
2																											
3																											
4																											
5																											
6																											
7																											
8																											
9																											
10																											

NUMÉRO DE COPIE:

Examen Cours B5 - Génie Logiciel

Année universitaire 2002-2003 ; Deuxième session : 29 mars 2003

Etude de cas 12 points (+ bonus 4 points)

La société Power Inc. (PI) réalise, maintient, adapte un ensemble de systèmes informatiques destinés à la surveillance et au contrôle-commande de différentes catégories d'équipements (Haute tension / HT, Moyenne tension / MT, Basse tension / BT) permettant la distribution de l'énergie (SC2DE – Système de Contrôle Commande de Distribution d'Energie). Elle est en charge également de participer au « trading » de l'énergie électrique qui s'est instauré entre les différents producteurs et consommateurs européens. Ce trading consiste à vendre de l'énergie en cas de sur-capacité et/ou à acheter de l'énergie en cas de sous-capacité (par exemple en cas de maintenance de gros équipements de production ou de pannes). Les contrats correspondants ont valeur exécutoire et doivent être honorés quel que soit l'état du système électrique, y compris en cas de défaillance de la production où il faut alors négocier en temps réel l'achat d'énergie. Le principe de ce trading est tout à fait comparable à ce qui se fait en matière de systèmes boursier sur le marché des actions.

La première version de ce système d'information qu'il a fallu mettre en place et expérimenter très rapidement n'est pas informatisée. Les échanges et les contrats se font par téléphones, fax et accessoirement par quelques messages Internet avec les différents partenaires.

En parallèle à cette mise en œuvre manuelle, Power Inc. a lancé les études préalables pour une informatisation de ce système sous la forme d'un projet Power-Trade-Contract (ou PTC). Power Inc. n'a aucune expérience de ce type de système d'information. De plus, la réglementation européenne et les exigences des partenaires font que le **délai de réalisation demandé** par la direction générale de Power Inc. a été **fixé à 1 an** pour l'ensemble des études et la réalisation d'une première version industrielle.

Cette 1ère version a été précédée par la réalisation d'une maquette-prototype (NB : la différence maquette prototype n'a pas été clarifiée, ni par FABLOG, ni par PI ; il y a donc une ambiguïté sur la nature de ce projet), confiée à la société FABLOG, qui a fait l'objet d'un marché de sous-traitance de gré à gré, sans mise en concurrence.

Convaincue par ses conseillers de l'applicabilité de la méthode RAD (*Rapid Application Development*) à ce type de situation, la direction informatique de Power Inc. a décidé de la prendre comme méthode de développement pour le projet PTC. De plus, pour des raisons de rapidité de développement, l'utilisation des NTI (Java, Internet et le web, les environnements et méthodes objets sont privilégiés, par exemple Websphere ou Java studio, par rapport aux méthodes plus traditionnelles).

L'expérience de PI en matière de pilotage-développement de systèmes est fondée sur le développement de très gros systèmes (500 à 1.000 KLS) de contrôle-commande plutôt temps-réel, avec de fortes contraintes de disponibilité et de sûreté ; ainsi que sur le développement de systèmes d'aides à la décision pour la planification de la production d'énergie (mais sans contrainte de délai particulière) qui mettent en œuvre des modèles mathématiques sophistiqués permettant la prévision des consommations. Elle n'a jamais réalisé de systèmes d'information au sens traditionnel du terme

Le schéma d'organisation pour cette réalisation est le suivant :

- Un client final, maître d'ouvrage (MOA) du projet, dont le rôle est d'exprimer le besoin (et de financer le projet) ; le MOA représente les usagers réels de ce futur SI qui n'ont pas de compétence particulière en informatique, et encore moins en développement de système d'information.
- Un maître d'œuvre industriel (MOI : Power Inc.) dont le rôle est : 1) la mise en forme des spécifications du système en vue d'une consultation auprès de sociétés tierces (CdC fonctionnel) ; 2) la sélection de la société retenue pour la réalisation ; 3) le pilotage industriel du projet et le suivi de la réalisation chez le sous-traitant (FABLOG) ; 4) la recette finale, en relation avec la MOA et les usagers réels.
- Un sous-traitant qui développe l'intégralité du logiciel de cette application. Le sous-traitant est libre d'organiser son équipe de développement comme bon lui semble sans en référer au MOI. Le sous-traitant doit cependant être certifié ISO 9000.

Rappel mathématique

La solution d'une équation de type $a = x^n$ est $x = a^{\frac{1}{n}} = \sqrt[n]{a}$ que l'on a sur la plupart des calculettes. Pour être sûr de vos calculs, vous pouvez toujours vérifier que $\left(\sqrt[n]{a}\right)^n = a$.

Question N°1 : étude de la réalisation de la maquette-prototype (4 points)

Le délai de réalisation pour la maquette-prototype a été fixé à 7 mois. Au terme de ces 7 mois la version industrielle sera mise en chantier.

Pour cette réalisation, FABLOG a décidé de mettre à la disposition de PI 2 ou 3 ingénieurs très expérimentés (selon les nécessités de la réalisation, mais ne connaissant pas le métier du trading de l'énergie) pour modéliser les besoins et développer la maquette-prototype.

Comprenant qu'il s'agit d'une maquette, les experts de FABLOG ont choisi de programmer la maquette dans un langage de maquettage « maison » appelé Visu-UML, de type Visual-Basic, basé sur le concept de diagrammes états-transitions du langage UML, avec des aides graphiques très puissantes pour définir les écrans de l'IHM de l'application. Dans ce langage, on estime que 100 LS de Visu-UML correspondent à environ 300 LS de JAVA/C++.

a) Compte tenu du délai imposé (7 mois) quel est l'effort, selon le modèle COCOMO, ou selon votre propre appréciation, que l'on peut consacrer à ce projet ? Quel est le volume de code Visu-UML correspondant à cet effort ?

NB : vous devez tenir compte du fait que ***les ingénieur sont des experts*** ; on considérera que *le projet est de type S*.

b) En vous servant des tableaux de ventilation de l'effort par phase-activité du modèle, donner une répartition des efforts par type d'activité. Proposer un profil de charges et d'effectifs correspondant aux efforts à fournir (c'est la planification de ce projet de maquettage).

c) Compte tenu du fait qu'il s'agit d'une maquette, quelles sont les phases et/ou activités de développement qui peuvent être simplifiées, voire purement et simplement annulée. Que peut-on en déduire pour ce qui concerne les efforts estimés en a).

d) Selon vous, quelles peuvent être les conséquences de l'ambiguïté maquette-prototype ? Peut-on espérer, au terme de la réalisation, disposer d'une architecture de l'application PTC permettant de démarrer immédiatement la réalisation industrielle ? Justifiez et expliquez vos arguments.

Question N°2 (3 points) - *Lire attentivement l'annexe N°2.*

Entre la fin de réalisation de la maquette-prototype et le démarrage du développement industriel qui est un contrat réalisé pour un montant forfaitaire, il s'est écoulé 3 mois. Les experts de FABLOG qui ont travaillé sur la maquette ont été de ce fait réaffectés à d'autres projets. FABLOG souhaite rentabiliser l'investissement réalisé pour le développement de la maquette-prototype en mettant en place une nouvelle équipe industrielle qui est décrite en annexe.
Le premier travail de cette nouvelle équipe est de mettre au propre, de façon définitive le cahier des charges fonctionnelles (CDCF) du projet PTC. Ce travail de spécification fait partie de la conception générale.
a) En lisant attentivement l'historique, à combien peut-on évaluer l'effort qui a été nécessaire à la réalisation de ce CDCF ? Expliquez votre raisonnement.
b) En faisant l'hypothèse que le travail des 2 développeurs pendant cette période, ajouté à l'effort précédent, revient à terminer la phase de conception générale, à combien peut-on estimer l'effort de développement de l'application PTC ?

NB : pour cela vous devez comptabiliser les efforts de PI et ceux des experts de FABLOG (le chef de projet et l'analyste ; les 2 développeurs quant à eux examinent le code source de la maquette) qui ont pris en compte la version livrée le 1/02 et participés aux réunions.
Vous remarquerez également que selon la table 5-2, la phase CG représente 16% de l'effort total.

Question N° 3 : Industrialisation de la maquette (5 points)

Le traducteur du langage Visu-UML dispose d'une option qui permet de générer un programme JAVA sémantiquement équivalent au programme initialement écrit en Visu-UML. Il a déjà été indiqué que le facteur d'expansion de cette traduction est dans un rapport 1 à 3 (c'est à dire que 1 ligne de Visu-UML génère en moyenne 3 lignes de JAVA).
La version industrielle consiste à :
a) faire ce qui n'était pas requis pour la maquette (documentation de conception, dossier de programmation, tests unitaires complets avec mesure d'un taux de couverture, tests d'intégration complets).
b) réaliser le complément de développement demandé pour cette version industrielle conformément à la mise au propre du CDCF (nouveaux écrans, traces d'instrumentation, pré et post-conditions pour l'intégrité des traitements, optimisation de certains accès à la base de données, etc.).
c) mettre en place une nouvelle équipe pleinement opérationnelle.
La nouvelle équipe qui se met en place, et qui récupère la maquette-prototype réalisée par les experts de FABLOG, peut considérer que les conditions de cette réutilisation reposent sur 3 hypothèses.
H1 : les experts ont parfaitement travaillé. La maquette se révèle être un vrai prototype, c'est à dire que l'architecture est réutilisable en l'état. En fait les experts, dans ce cas, auraient anticipé une équipe industrielle correspondant à un profil nominal au sens COCOMO.
H2 : Seul 50% du code réalisé est réutilisable ; ce code correspond à la partie MODEL du pattern MVC. Le reste est à refaire intégralement.
H3 : Rien n'est en fait réutilisable. Ce qui a été réalisé par les experts est une maquette, et non un prototype. L'architecture n'est pas conforme au pattern MVC.
Lorsque le nouveau chef de projet est nommé, son hypothèse de travail est plutôt H1. De ce fait il propose à la DG de FABLOG de constituer une équipe dont le profil est nominal au sens

COCOMO (voir le tableau COCOMO 8-2 dans votre cours). Sur cette base, il a estimé le travail de terminaison à 267hj.
Cependant, pour des raisons de stratégie industrielle, la DG de FABLOG a consenti à PI une remise tout a fait exceptionnelle ; cette version industrielle sera facturé 200hj (bien qu'elle ait été estimée à 267hj), soit une remise de 25%. Pour compenser ce manque à gagner, la DG de FABLOG impose au chef de projet une équipe de débutants en argumentant auprès de son chef de projet que le travail d'architecture et que les difficultés ont déjà été résolues.

a) Dans l'hypothèse H1 quelle est le volume de code de l'application PTC (c'est à dire la maquette traduite en JAVA auquel il faut rajouter les développements correspondant à 267hj) ? Pour ces 267hj, quelle est la productivité : est-ce 4KLS/HA comme l'indique le vade-mecum ou le mode S de COCOMO, ou beaucoup plus (ce qui serait normal, car l'architecture est déjà faite, et les principales difficultés résolues) ? Justifier votre réponse et vos arguments.
b) Pensez-vous que la DG de FABLOG est dans son rôle d'imposer à son chef de projet une équipe qui ne correspond pas à ses souhaits ?
c) A partir de quelle date, dans l'historique, le chef de projet prend-il conscience qu'il n'est pas dans l'hypothèse H1, mais plutôt dans H2 ou H3 ? Quels sont les symptômes ? Quel est le rôle du Directeur de projet arrivé en mars ?
d) Reconstituer très soigneusement l'historique de la version, semaine par semaine, en faisant apparaître les augmentations d'effectifs, en tenant compte des heures supplémentaires, des week-end travaillés, etc. A partir de cette analyse, quelle est le coût réel de développement de ce qui a été livré en Août ? Quel est selon vous la perte d'effort occasionné par le fait que l'équipe ne s'est pas aperçue tout de suite qu'il y avait un grave problème de réutilisation ?
e) La version livrée en août totalise 72 KLS. Quelles sont les impasses et les simplifications que FABLOG a du faire, compte tenu des délais, et compte tenu de l'équipe réelle, pour arriver à livrer un code qui fonctionne a peu près correctement ? A combien peut-on estimer ce qui a pu être récupéré de la maquette ? Qu'est ce qui reste à faire pour que la maintenance se fasse dans de bonnes conditions ?

Bonus

Question N°4 : Amélioration de la testabilité avec le pattern MVC (2 points)

L'application PTC est constituée de 10 écrans comportant en moyenne 10 champs valeurs (numériques et/ou alpha-numériques) nécessitant une saisie par l'utilisateur de l'application ; chacun de ces champs doit être vérifié à la saisie, et éventuellement expliqué avec des « aides en lignes ». Ces 10 écrans (gérés par le composant « View » de MVC) sont organisés comme suit :

- 5 écrans pour saisir les données spécifiques aux transactions énergétiques,
- 3 écrans de consultation des données tarifaires pour préparer les contrats en fonction des caractéristiques du clients (ces écrans permettent de spécifier les accès aux données tarifaires),
- 2 écrans d'édition pour éditer et expédier les contrats.

A priori, toutes les combinaisons d'écrans sont autorisées par le contrôleur MVC.
Quelle est le nombre de combinaisons que doit gérer le « Controller » de MVC sachant que toute opération nécessite 3 types d'actions : une saisie de données de transactions, une consultation tarifaire et une édition ?
Combien faut-il prévoir de messages d'erreurs si l'on veut que l'utilisateur soit bien guidé dans les différentes saisies à effectuer ?

Question N° 5 : Assurance qualité et effort de tests (2 points)

Donner une hypothèse de chiffrage du coût de fabrication de scénarios de tests permettant de vérifier que tous les messages d'erreurs et les aides en lignes sont conformes aux besoins des usagers ? Peut-on faire l'hypothèse que les 3 type d'actions (saisie, consultation, édition) sont indépendantes ?

NB : il faut considérer chaque écran, et les données qui leur sont associées, en provoquant des actions erronées sur ces données. Ces actions erronées déclencheront, via le « Controller » des messages indiquant la nature de l'erreur, et ce qu'il y a lieu de faire (voir les automates vus dans le cours conception ; c'est la même logique de contrôle).

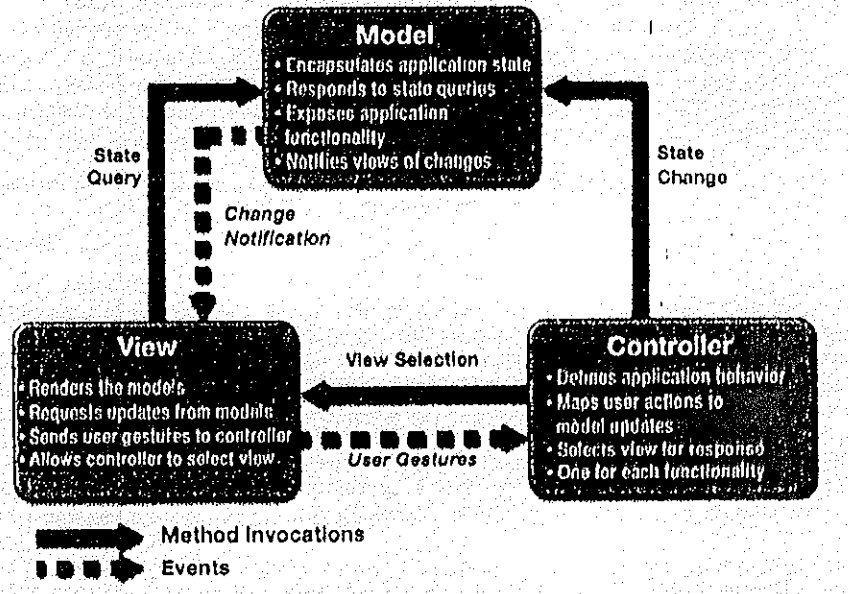
Combien de scénarios d'enchaînements des écrans faut-il prévoir pour valider correctement la combinatoire autorisée par le « Controller » ? Ce nombre vous paraît-il compatible avec l'effort de test et d'intégration fait dans l'industrialisation du projet.
Donner une façon de chiffrer le travail de fabrication d'un scénario qui enchaînerait 5 écrans successifs (NB : il faut renseigner tous les champs données de ces écrans).

193

ANNEXES

Annexe N°1

Principes du design pattern MVC



View : Son rôle principal est de présenter les modèles, d'envoyer les actions utilisateur au module de contrôle.
Controller : C'est lui qui définit le comportement de l'application en fonction des actions utilisateur, il assure également les correspondances entre les actions utilisateurs et les modifications. Enfin il sélectionne les vues pour la réponse.
Model : Répond aux requêtes en notifiant les changements. C'est ce module qui "connaît" les fonctionnalités de l'application.

Ce design pattern permet de rendre indépendantes les fonctionnalités de l'application (Model) de la présentation (View) et du comportement de l'application (Controller).

Annexe N°2 : Documents FABLOG

a) Historique de la version

Dates	Actions
10/01/2002	Power Inc. (PI) donne son accord à FABLOG pour développer une version 2002 de l'application sur une base de 267 hj (coût supporté par PI de 200 hj). Ce chiffrage ne tient pas compte des évolutions 2002 qui seront chiffrées suite à la rédaction d'une nouvelle version du cahier des charges système.
01/02/2002	PI livre à FABLOG la nouvelle version du Cahier des Charges Fonctionnel (dont la rédaction a nécessité 2 personnes à temps plein pendant 6 semaines).
Semaine du 4/02	PI organise 4 réunions de 2 heures pour répondre aux questions fonctionnelles de FABLOG où sont présents des experts métiers de PI.
06/02 au 11/02	Réunion toute une journée avec le client et participation de FABLOG pour répondre à toutes les questions fonctionnelles préparées par FABLOG (sur place chez PI, avec accès sans problème au client et aux experts métier).
15/02/2002	Le Cahier des Charges Fonctionnel est complété par PI avec les remarques de FABLOG.
18/02/2002 ↓ 3 semaines	FABLOG prend acte que PI a répondu à toutes les questions fonctionnelles, et présente le planning de développement (base du planning de la commande) et chiffre les évolutions 2002. Fablog va même jusqu'à annoncer que les évolutions demandées seront légèrement moins coûteuse que prévues.
07/03/2002	Annonce d'une dérive de 4 hj + 1er renforcement de l'équipe (2 personnes). L'équipe initiale était constituée de : 1 chef de projet + 1 analyste + 2 développeurs + demande d'autorisation pour le soir et week-end. PI indique que c'est un signe de retard car ces ressources sont utilisées tout de suite.
13/03/2002	PI indique que l'avancement de la rédaction des spécifications présente des signes de retard (2 à 3 jours de retard + détection d'incohérences). ***Fablog nomme un Directeur de projet.*** FABLOG indique qu'il y a une dérive de 50% sur la réalisation du lot 1.
20/03/2002	FABLOG annonce 20 hj de retard et annonce un 2ème renforcement de l'équipe (+3 personnes). Donc maintenant, il y a 9 personnes dans l'équipe.
27/03/2002	FABLOG annonce 40 hj de retard et annonce un 3ème renforcement de l'équipe (+4 personnes). Actuellement l'équipe est constituée de 9 développeurs et 4 personnes pour l'encadrement dont 1 directeur de projet de très haut niveau arrivé en mars.
Août 2002	Fin des travaux ; le code JAVA livré totalise 72KLS.

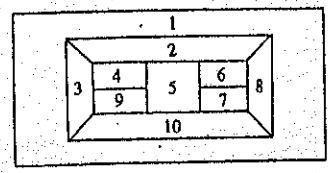
2. Maturité de l'équipe FABLOG (bref résumé des CV, sans les noms ...)

Fonction	Formation	Expérience	Années d'exp.	Remarques
1-Directeur de projet	IFP (Institut Français du Pétrole)	Java, Websphere, Oracle, Gestion de projet	12	Pas à 100% du temps
2-Chef de projet	ENSAIA (Agronomie)	Gestion de projet	4	S'occupe des tests actuellement
3-Analyste expérimenté	INSA	Java, Websphere, Oracle	3	A fait la plupart des écrans IHM
4-Expert logiciel	DUT Informatique	Oracle, peu de Java	7	Pas à 100% du temps
5-Ing. d'étude et dév.	ENCCF (chimie de Clermont Ferrand)		0,5	
6-Ing. d'étude et dév.	DUT génie informatique	Un peu de Java	0,5	
7-Expert technique Oracle	Ingénieur école biologie industrielle	Unix, Oracle, PL/SQL, web application server (?), Apache (serveur Web "open source")	4	Pas à 100% du temps
8-Nouveau chef de projet (arrivé le 25/03/2002)	DESCAF (commercial et financier)	MVS, Unix, PL/SQL, gestion de projet	7	
9-Ing. d'étude et dév. (arrivé le 11/03/2002)	Ecole des Mines de Douai		0,5	
10-Stagiaire (arrivé le 18/03/2002)	?			N'est pas compté dans l'effectif
11-Ing. d'étude et dév. (arrivé le 15/04/2002)	DESS mécanique des fluides	Unix, Linux, NT, C/C++, PL/SQL, Java, JavaScript, HTML, Oracle, Websphere	1,5	
12-Concepteur/développeur (arrivé le 15/04/2002)	EPITA	NT, Unix, C/C++, HTML, Java, site Web	8	
13-Expert logiciel	?	Java, JavaScript, HTML, Oracle, NT, Unix	4,5	Pas à 100% du temps
14-Architecte technique	Ecole Centrale (Lille)	Java, JavaScript, HTML, Oracle, NT, Unix	1,5	

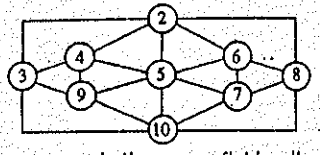
Examen session II 29 mars 2003 **Partie II**

EXERCICE DE TEST

Voici un exemple de carte ou schéma divisé en zones homogènes (les nombres sont les numéros arbitrairement affectés aux régions).



Une carte se modélise par un graphe symétrique, dit planaire, dont les sommets représentent les régions, et les arêtes le lien de voisinage entre deux régions.
Par exemple, pour la partie centrale de la carte ci dessus :



La procédure LIRCARTE ci-jointe permet de lire sur un fichier d'entrée (de type texte) les données d'une carte C. Le paramètre de sortie NBZON représente le nombre de zones sur la carte; sa valeur est suivie, dans le fichier, pour chaque zone ZONLU de 2 à NBZON, du nombre NBVOIS de ses voisins de numéro <ZONLU, puis des NBVOIS numéros de ses voisins. Un certain nombre d'erreurs de saisie possibles, dont la fin prématurée de fichier, sont testées par la procédure de lecture.

On désire procéder au test unitaire, par la méthode des couvertures, de cette procédure LIRCARTE.
Les spécifications sont précisées dans les commentaires.

I. PARTIE SYNTAXIQUE

3 points Q1.1 -Etablir le graphe de commande de la procédure LIRCARTE
1 point Q1.2 -Calculer son nombre cyclomatique. Quel sera le nombre maximum de chemins nécessaires pour effectuer la couverture totale des arcs du graphe ?

II. PARTIE SEMANTIQUE

4 points Q2 Définir un ensemble de chemins permettant une couverture exhaustive des arcs (ou liens) du graphe de commande. Préciser avec soin la sémantique de chaque chemin choisi en fonction des données du fichier.

N.B : On utilisera impérativement la grille ci-jointe dans la réponse à cette question. Les nombres d'arcs et de chemins qui y figurent ne préjugent bien sûr en rien des nombres effectifs d'arcs de votre graphe, ni de chemins nécessaires pour le couvrir...

Que concluez-vous du nombre de chemins par rapport au nombre cyclomatique?

Comment procéderiez vous si l'on vous demandait maintenant une couverture totale des nœuds (ou sommets) du graphe?

P.J : le texte de la procédure LIRCARTE, plus la grille de réponse (obligatoire).

195

CORRIGÉ

Examen Cours B5 - Génie Logiciel

Année universitaire 2002-2003 ; Deuxième session

Etude de cas 12 points (+ bonus 4 points)

Question N°1 (4 points)

Point a)
On peut raisonner de 3 façons :
1) Empirisme pur (raisonnement d'expert, car cela requiert une grande expérience pour ne pas dire de bêtises !!!) : le 1er expert X1 démarre la prestation et commence à réfléchir et à organiser le travail. A T0+1mois, le second expert X2 arrive et peut immédiatement démarrer sur les hypothèses de travail mises en place par X1. A T0+2 mois, le 3ème et dernier expert X3 arrive et démarre immédiatement.
Au total on a : $7 + 6 + 5 = 18hm$ d'effort.

2) Avec l'équation du vade-mecum $\frac{7}{12} = 0{,}5 \times \sqrt[3]{Eff}$, soit $Eff = 1{,}17^3 = 1{,}6ha = 19hm$

3) Avec l'équation COCOMO $7 = 2{,}5(Eff)^{0{,}38}$, soit $Eff = \sqrt[0{,}38]{2{,}8} = 15hm = 1{,}25ha$
Notons ici que l'équation COCOMO, comme celle du vade-mecum, ne prend pas en compte le fait qu'il ne s'agit pas d'un projet nominal qui comporte toutes les phases prévues par le cycle de développement. Le mode de calcul empirique est le meilleur.
Pour calculer le volume de code, il faut tenir compte du fait que le travail de maquettage est essentiellement un travail de programmation, et que ce travail est fait par des experts. Sur cette base, le vade-mecum donne, avec la règle 40-20-40, une productivité de $4KLS \times 5 = 20KLS/ha$.
Soit un volume de code $\frac{18}{12} \times 20 = 30KLS$ de Visu-UML, si l'on considère que les 18hm sont exclusivement consacrés à la programmation, ce qui est évidemment très excessif. En utilisant une distribution de Pareto, soit 80% de l'effort pour la programmation et le reste pour des réunions et/ou de la réflexion, on obtient seulement : $24KLS$.
Pour utiliser correctement COCOMO, il faut raisonner par phases (ou mieux encore par activités, mais cela fait beaucoup plus de calcul)

Ventilation de l'effort par phase pour la maquette
Ce § traite le point c).
En utilisant le tableau de ventilation 5-2 du cours, on obtient :

CG	16%,	pas de conception générale pour la maquette
CD	24%,	très peu pour une maquette, soit 5%.
P-TU	38%,	Très peu de TU, soit 20% de programmation ; mais on ne programme pas tout ! donc on va tomber aux alentours de 10 à 15%.
I-T	22%	Pas (ou très peu) d'intégration ; on ne fait la validation que sur quelques cas nécessaires à la démonstration de la maquette, soit 7% pour TU+IT.

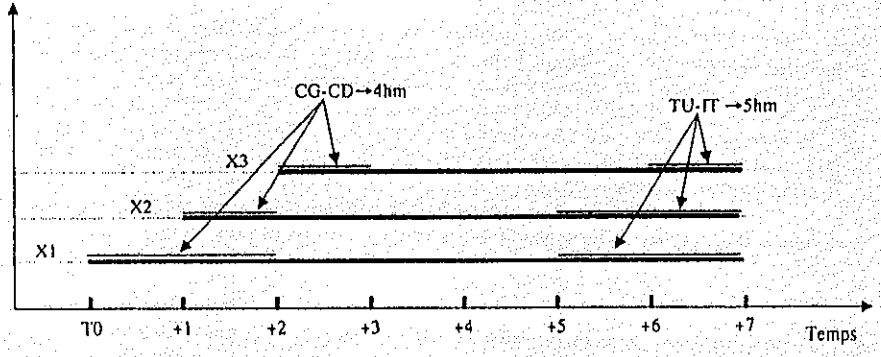
La base 100 est réduite à 20 – 25 maximum, soit une productivité moyenne de 4 à 5 fois supérieure à 350 LS/hm.

On retombe sur des chiffres très proches du calcul empirique (ce qui montre la solidité du modèle).

Point b)
Compte tenu de ce qui précède, on peut considérer que les poids relatifs pour une maquette-prototype de ce type sont :

CG-CD	:	6	→	24%, soit 4hm (essentiellement sur la partie IHM, et un sous ensemble du Controller MVC)
P	:	12	→	48%, soit 9hm
TU-IT	:	7	→	28%, soit 5hm
Total	:	25		soit 4 fois moins coûteux que le produit industriel fini.

On peut faire le schéma suivant pour le plan projet :



Le point c) a été traité avec a).

Point d).
L'ambiguïté maquette prototype est très préjudiciable à la suite du projet.
Dans une maquette on ne se préoccupe pas de réutilisation. Ce qui est important est la vitesse avec laquelle on va pouvoir montrer une partie de la réalisation au client (ici la maîtrise d'ouvrage) ; on peut considérer qu'une maquette d'IHM bien faite correspond à une spécification validée des IHM, du point de vue client (reste bien sûr à la valider logiquement) ; on peut dire que cela correspondrait à la partie « View » du pattern MVC. Dans un prototype, qui est en fait une réalisation incomplète, on traite complètement le problème de l'architecture (par exemple un pattern comme MVC complet ; d'où l'intérêt de réutiliser un pattern déjà validé logiquement).
La distribution des efforts sera donc encore différente, soit par exemple :

CG-CD :	2	→	28%,	(modélisation des écrans du composant View de MVC, sans les traitements d'erreurs, inutile pour le démonstrateur)
P :	4	→	58%,	avec une génération automatique de code à partir des spécifications d'écrans.
TU-IT :	1	→	14%,	Pas d'intégration.
Total :	7			

On remarquera que entre la maquette IHM et le produit industriel, on est dans un rapport $\frac{100}{7} \approx 15$. Ce rapport aurait du être connu des experts, lesquels auraient du informer la DG de

Fablog sur les conditions du passage au produit industriel, et la mettre en garde des risques encourus avec une équipe de débutants.

A titre de comparaison, regardons le nombre de KLS obtenu avec différents niveaux d'expertise selon COCOMO pour un effort de $18KLS$:

Nominal : $18 = 2{,}4(KLS)^{1.05}$, soit $KLS = \left(\frac{18}{2{,}4}\right)^{\frac{1}{1.05}} = 7{,}5^{0.952} \approx 7KLS$

1er niveau d'expertise :

$$18 = 1{,}2(KLS)^{1.05}, \text{ soit } KLS = \left(\frac{18}{1{,}2}\right)^{\frac{1}{1.05}} = 15^{0.952} \approx 13{,}2KLS$$

2ème niveau d'expertise (8 à 10 ans d'expérience) :

$$18 = 0{,}8(KLS)^{1.05}, \text{ soit } KLS = \left(\frac{18}{0{,}8}\right)^{\frac{1}{1.05}} = 22{,}5^{0.952} \approx 19{,}4KLS$$

Les simplifications apportées pour un prototype permettraient de ×2 ou 3 les *KLS*.

Question N°2 (3 points)

Effort de réalisation du CDCF
En lisant attentivement l'énoncé, on peut faire le décompte suivant :
Pour PI :
- 6 semaines à 2 personnes, soit $6 \times 5 \times 2 = 60hj$
- 4 réunions avec Fablog, soit 5 personnes ; en comptant les temps de préparation et de comptes rendus (CR) de réunions, on a : $4 \times 5hj = 20hj$
- 1 réunion de 1 journée à 10 personnes + préparation et CR, soit $10+5 \approx 15hj$

Soit 95hj, dont 15 pour Fablog (participation de Fablog aux réunions).
Pour Fablog :
La période de consolidation de la spécification de conception générale se déroule du 10/01 au 15/02, soit 20 jours ouvrés, pour 2 personnes. Sur cette période l'effort de Fablog se monte donc à 40hj, dont 15hj consommés dans les réunions.
L'effort pour la réalisation du CDCF est donc : $95 + 40 - 15 = 120hj$.
En incluant à cet effort le travail des 2 programmeurs pour examiner la maquette et finaliser la conception générale, soit $40hj$ de plus, on obtient un total de $160hj$ correspondant à la phase CG du modèle COCOMO.
Sachant que cette phase représente environ 16% du total, on peut estimer le coût de développement à $\frac{160}{16} \times 100 = 1000hj$, soit $4{,}5ha$ pour une équipe de *profil nominal* selon COCOMO.
Si l'on fait l'hypothèse que l'on est seulement à 12% de l'effort total, on a un effort total de $\frac{160}{12} \times 100 = 1334hj$.
Cette fourchette est à comparer avec les résultats obtenus à la question 3.

Question N° 3 : Analyse du dossier technique et historique de la réalisation (5 points)

Point a).

Dans l'hypothèse H1, on récupère la maquette écrite en Visu-UML. La conversion de ce code en Java donne un programme de conversion brute de $24 \times 3 = 72KLS$.
A ce code, il faut rajouter ce qui a été produit dans les 267hj d'effort complémentaire prévu par Fablog.
Dans le profil du vade-mecum (40-20-40), la productivité moyenne est de 4KLS toutes phases comprises. Or, dans notre hypothèse, la conception est déjà faite ; les 267hj correspondent donc à du code testé et intégré, soit une productivité de $\frac{4}{0,6} = 6,7KLS/ha$ au lieu des 4 du mode complet. On remarquera que si on ne fait pas l'intégration, la productivité passe à 20 !

Avec 267hj, on peut donc produire $\frac{267}{220} \times 6,7 = 8,2KLS$ qui viendront s'ajouter au 72KLS.
Si une partie des 267hj sert à faire des compléments d'intégration sur les 72KLS converties, cela diminue d'autant les lignes de code nouvelles (il est évident qu'il faudra faire un gros complément d'intégration).

Point b).
Le DG de Fablog n'est évidemment pas dans son rôle en imposant le profil de l'équipe. Sa décision va mettre le chef de projet dans une situation extrêmement périlleuse, car sa décision revient, en fait, à proposer une équipe beaucoup moins mature et beaucoup moins productive.
Le manque de maturité peut faire en sorte que les jeunes programmeurs ne comprennent même pas ce que les experts ont réalisé, et ne sachent même pas réutiliser le code Visu-UML, d'où la tentation de tout jeter !!!

Point c).
Un effort de 267hj de production, à 4 personnes correspond à un délai d'environ 4 mois (à ce stade, nous sommes encore dans un projet de type RAD).
En début mars, soit 1 mois après le début réel des travaux d'industrialisation, on voit l'effectif de l'équipe s'accroître considérablement. On peut supposer que le chef de projet vient de prendre conscience que le code de la maquette n'est pas réutilisable en l'état : il est dans l'hypothèse H2, voire peut-être H3. Il est tout à fait symptomatique que quelque chose de très grave s'est produit puisque l'on passe d'un relatif optimisme (on est légèrement en avance le 18/02) à une augmentation très importante de l'effectif qui se produit en 2 étapes, en 2 semaines d'intervalle, avec nomination d'un directeur de projet (ce qui montre que le 1er diagnostique n'était pas le bon !).
Le rôle du Directeur de projet (très expérimenté, il a 12 ans d'expérience) est de statuer sur l'état réel du code de la maquette, et de faire le tri entre ce qu'il faut garder et ce qu'il faut jeter. Il est normal qu'avec son expérience et son ancienneté dans Fablog, il soit à même de prendre des décisions que ne pouvait pas prendre le chef de projet initial (3 ans d'expérience seulement, donc beaucoup moins crédible aux yeux de la DG).

Point d).
Sur la base des informations de l'énoncé du cas, on peut faire un décompte analytique, mois par mois, comme suit.
NB : il faut tenir compte du fait que tout le monde n'est pas à 100%. Les mois sont comptés à 20j.

Mois	Effectif	Charges/Effort	Remarques
Février	4	70hj	Pas à 100% sur le mois
Mars	6	100hj	Pas à 100% sur le mois
Avril	13	220hj	Sur avril-juin, 30% d'heures sup. soit :
Mai	13	220hj	660×1,3 = 858hj réel. Horaire de jour + week-end.
Juin	13	220hj	NB : difficile de faire plus sur une longue période !!!
Juillet	13	220hj	Sur juillet-Août 50% d'heures sup. soit
Août	6 (ou 9)	80hj (pour 6) 120hj (pour 9)	300×1,5=450hj réel. En août, il y a qq. jours de vacances.
Total brut		1130hj 1170hj	
Total normalisé		1478hj	Il faut tenir compte des heures sup. + week-ends pour se conformer aux hypothèses du modèle.
Total efficace		**1342*hj***	Voir ci-dessous.

Le total normalisé donne un chiffre brut intégrant les heures supplémentaires.
Perte d'effort
Pour se conformer aux règles du modèle COCOMO, il faut estimer ce qui correspond effectivement à la refonte du programme initial ; on peut estimer que sur la période février-mars **80% (distribution de Pareto) a été perdu**, soit en normalisé 34hj de travail effectif. Selon ces hypothèses, on aurait un ***effort normalisé efficace*** de 1478-136=1342hj.

Soit un effort de $\frac{1342}{220} = 6,1ha$. En productivité basique vade-mecum, cela fait :

$$6,1 \times 4 = 24,4KLS$$

en productivité expert, on aurait : $6,1 \times 12 = 74KLS$.
En intégrant le fait que l'équipe est essentiellement constituée de débutants, on serait plutôt aux alentours de 30 *KLS*.
Avec de ***vrais experts***, il n'y a aucun problème à produire 74KLS, avec peut-être qq. impasses sur la documentation ! mais l'essentiel est le délai et la qualité du produit livré ; le reste viendra après. Le problème est que l'équipe n'est pas formée de vrais experts.

Point e).
La fabrication de 72KLS, dans le modèle nominal COCOMO nécessiterait un effort de

$$Eff = 2,4(72)^{1.05} = 214hm = 17ha$$

Mais l'équipe n'est pas nominale. Il y a certes des architectes expérimentés, mais les programmeurs sont vraiment débutants (5, 6 et 9 n'ont jamais réellement programmé !!!).
En faisant l'hypothèse que l'on a pu récupérer 50% (hypothèse H2), il faut produire 36KLS de code avec les 6,1ha d'effort calculés ci-dessus.

En nominal cela donne : $Eff = 2,4(36)^{1.05} = 103hm = 8,6ha$ pour tout faire. Si l'on tient compte que 4, 12, 13 et 14 connaissent bien MVC, on peut significativement réduire le coût de la conception (80% de réduction). Dans la table 5-2 CG + CD compte pour 40%, soit un effort nominal de 3,5ha. En faisant l'hypothèse que ce coût est réduit de 80% par la connaissance du pattern MVC, le coût CG+CD se ramène à $3,5 \times 0,2 = 0,7ha$; on a donc économisé 2,8ha, ce qui ramène le coût de fabrication des 36KLS à 5,4ha.
On pourrait affiner le calcul en intégrant que les programmeurs sont des débutants, ce qui augmenterait l'effort.
On a donc une explication de ce qui s'est passé, conforme au modèle.

197