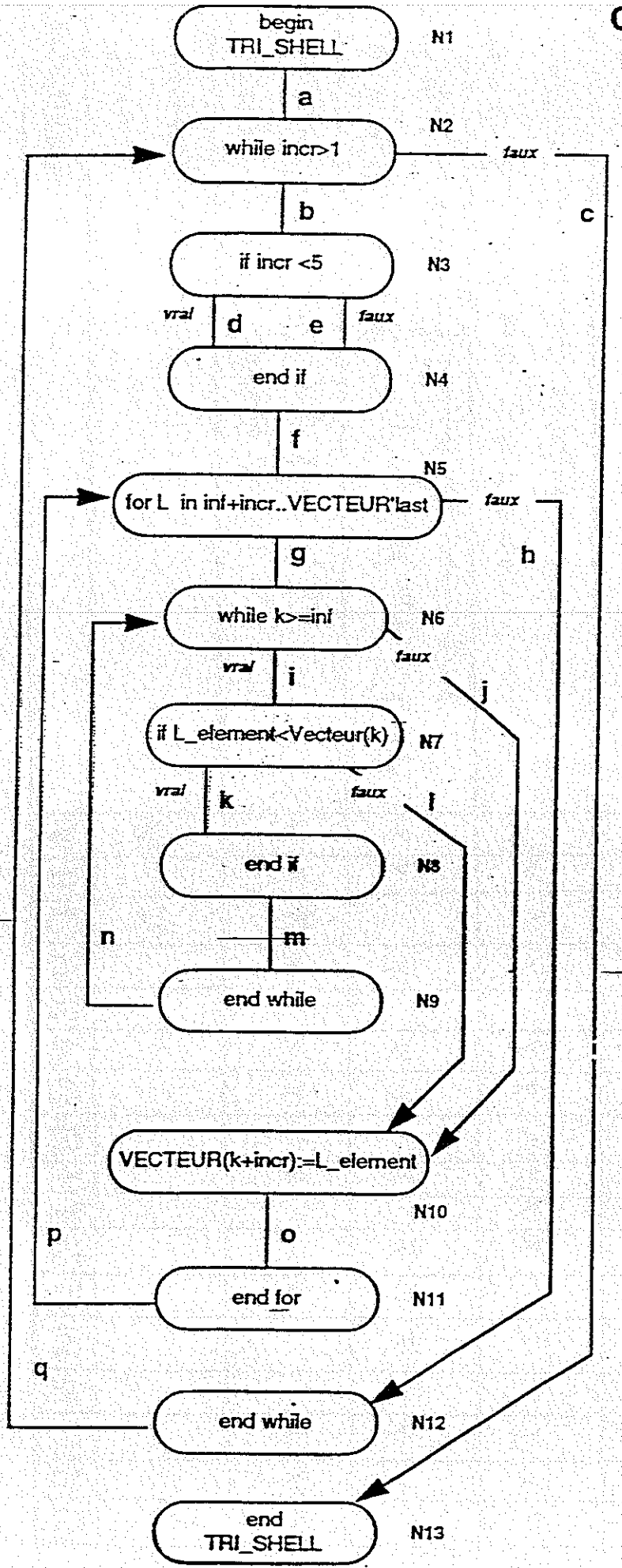


Correction 1ère Question

Nombre d'arcs: 17
Nonbre de nœuds: 13
Composante connexe: 1

Nombre cyclomatique: 6
(M = Nb_arc - Nb_nœud + 2)

Analyse des chemins pour C0-C1:
1. Vecteur vide: a-c
2. Vecteur <5:
-non trié:a-b-d-f-{g-i-{k ou l au moin 1 fois}
-o-p} sortie des boucles par h ou j
trié: idem sauf qu'on ne passe jamais par k
3. Vecteur > 5 quelconque pour passer par e

Conclusion:

Le nombre cyclomatique indique un nombre de tests supérieur à ce qui est nécessaire pour obtenir une couverture C0-C1

CHEMINS	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
a c Vecteur vide	X		X														
Vecteur à 2-4 éléments trié a b d f g i l o p z h q c	X	X	X	X		X	X	X	X			X			X	X	X
a b d f g i l o p g i k m n i l o p h q c Vecteur à 3 éléments tels que : 1 5 3	X	X		X		X	X		X		X	X	X	X	X	X	
a b d f g i k m n j o p h q c Vecteur à 2 éléments en désordre	X	X	X	X		X	X	X	X	X	X		X	X	X	X	X
a b e ... Vecteur quelconque à plus de 5 éléments	X	X			X												

N.B. : Tous les arcs ⇒ Tous les Nœuds.

Cnam

Génie Logiciel
EXAMEN 93-94

2ème session
14 Septembre 1994

Durée de l'épreuve : 3 heures
Tous documents autorisés

Chaque sujet doit être développé sur une copie séparée
vous devez donc rendre 3 copies

Notation :
- Synthèse de cours : /5
- Exercice de modélisation de système : /7
- Exercice de programmation : /5
+ note obtenue en ED/5
troncature à 20

1. Question de cours

Études de cheminement dans un programme Ada

A partir du programme Ada ci-joint, vous réaliserez les opérations suivantes :

1. Construire le graphe de contrôle associé à ce programme.

2. Calculer le nombre cyclomatique associé au graphe de contrôle

3. Déterminer, en examinant les conditions de chemins, l'ensemble des valeurs initiales qu'il faut fournir au programme de façon à vérifier que :

a) tous les noeuds du graphe sont parcourus au moins une fois (Couverture CO).

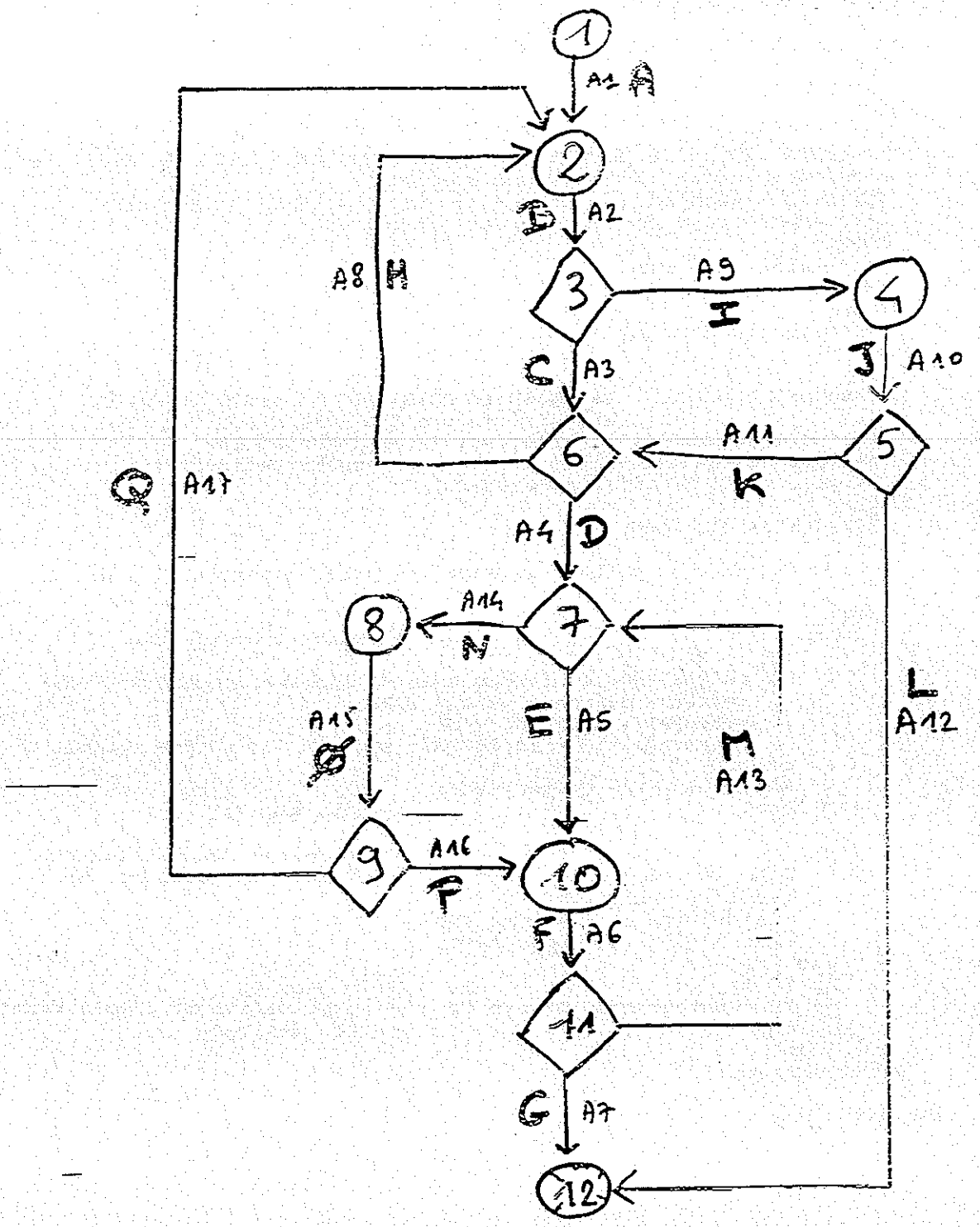
b) Tous les arcs du graphe sont parcourus au moins une fois (Couverture C1).

Dans les 2 cas, qu'en déduisez vous par rapport au nombre cyclomatique.

140

```
--*********************************
--*    LE PROBLEME DU SAC-A-DOS    *
--*********************************

with ENSEMBLE_DE;
generic  NB_CHARGES: positive;
package  SAC_A_DOS is
  subtype NUMERO_CHARGE is positive range 1..NB_CHARGES;
  subtype poids is positive;
  type LISTE_DE_POIDS is array (NUMERO_CHARGE) of poids;

  package p_ens is new ENSEMBLE_DE (numero_CHARGE);
  use p_ens;
  subtype ENS_CHARGE is ENSEMBLE;        -- renommage.
  function EST_VIDE (E: ens_CHARGE) return boolean;

  generic
    with procedure traiter (p: poids);
  procedure LISTE_DES_CHARGES (E: ens_charge; ch: LISTE_DE_POIDS);

  procedure DECOMPOSER (LA_MASSE:in POIDS; EN:in LISTE_DE_POIDS;
                        RESULTAT:in out ENS_CHARGE);

end sac_a_dos;
```

```
package  body SAC_A_DOS is

   function EST_VIDE (E: ens_CHARGE) return boolean is
   begin
     return E = ENS_VIDE;
   end;

   procedure LISTE_DES_CHARGES (E: ens_charge; ch: LISTE_DE_POIDS) is
     procedure trait_num (num:numero_charge);
     procedure enumerer_les_charges is new POUR_TOUS (trait_num);
     procedure trait_num (num:numero_charge) is
     begin
       TRAITER (CH (num));
     end;
   begin
     enumerer_les_charges (E);
   end;

   procedure DECOMPOSER (LA_MASSE:in POIDS; EN:in LISTE_DE_POIDS;
                         RESULTAT:in out ENS_CHARGE) is
     NP: natural range 0..en'last:=0;     -- pseudo-pile.
     residu: natural:=LA_MASSE;           -- masse residuelle.
     weight: LISTE_DE_POIDS renames EN;
   begin
     RESULTAT:= ENS_VIDE;

        -- La pile NP definit le sort des poids numerotes 1 a NP.

     loop
       loop                                -- Phase d'empilement.
            NP := NP + 1;
            if weight (NP) <= residu then -- inclure le poids no. NP
               residu:= residu - weight (NP);
               RESULTAT:= RESULTAT + NP;
               if residu = 0 then RETURN; end if;  -- succes obtenu.
            end if;
            exit when NP = weight'last;          -- echec...
       end loop;                           -- (fin empiler).

       loop                                -- On depile (partiellement).
         if DANS (NP, RESULTAT) then       -- exclure le poids no. NP
            RESULTAT:= RESULTAT - NP;
            residu:=residu + weight (NP);
            exit when NP < weight'last;
         end if;
         NP := NP - 1;                     -- On depile.
         if NP = 0 then RETURN; end if;    -- echec definitif ("pile vide").
       end loop;                           -- (fin depiler).
     end loop;
   end Decomposer;

end sac_a_dos;
```



$$M = A - N + 2p$$
$$= 17 - 12 + 2 \quad = 7$$
$$= 6 + 1$$

COUVERTURE C1 (ARCS)

CHEMINS	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1 a b i j l 1 poids = La_Masse	X	X							X	X		X					
2 a b c d e f g 1 poids > La_Masse	X	X	X	X	X	X	X										
3 a b c h b c d e f m e f g 2 poids > La_Masse	X	X	X	X	X	X	X	X					X				
4 a b i j k d n o p f g 1 poids < La_Masse	X	X		X		X	X		X	X	X			X	X	X	
5 a b i j k h b c d e f m n o q, etc. 1 poids < La_Masse, 1 poids > La_Masse	X	X	X	X	X	X		X	X	X	X		X	X	X		X

SAC-À-DOS

N.B. : Couv. des Arcs ⇒ Couv. des Nœuds

N.B.2 : ∃ boucle ⇒ Nb. de chemins (5) < M (7)

EXAMEN DE GENIE LOGICIEL

1ère session du 26 juin 1995

3 sujets (Durée 3 h.)

1. Exercice de conception

2 . Exercice de tests

3 . Question de cours

Tous les documents sont autorisés.

Notation :

- Exercice de tests sur 5 pts

Exercice de tests

La procédure ADA suivante (COPIER) implémente l'affectation "profonde" (par copie) d'une liste linéaire (simple) représentée sous forme de liste chaînée.

1) Etablir son graphe de commande

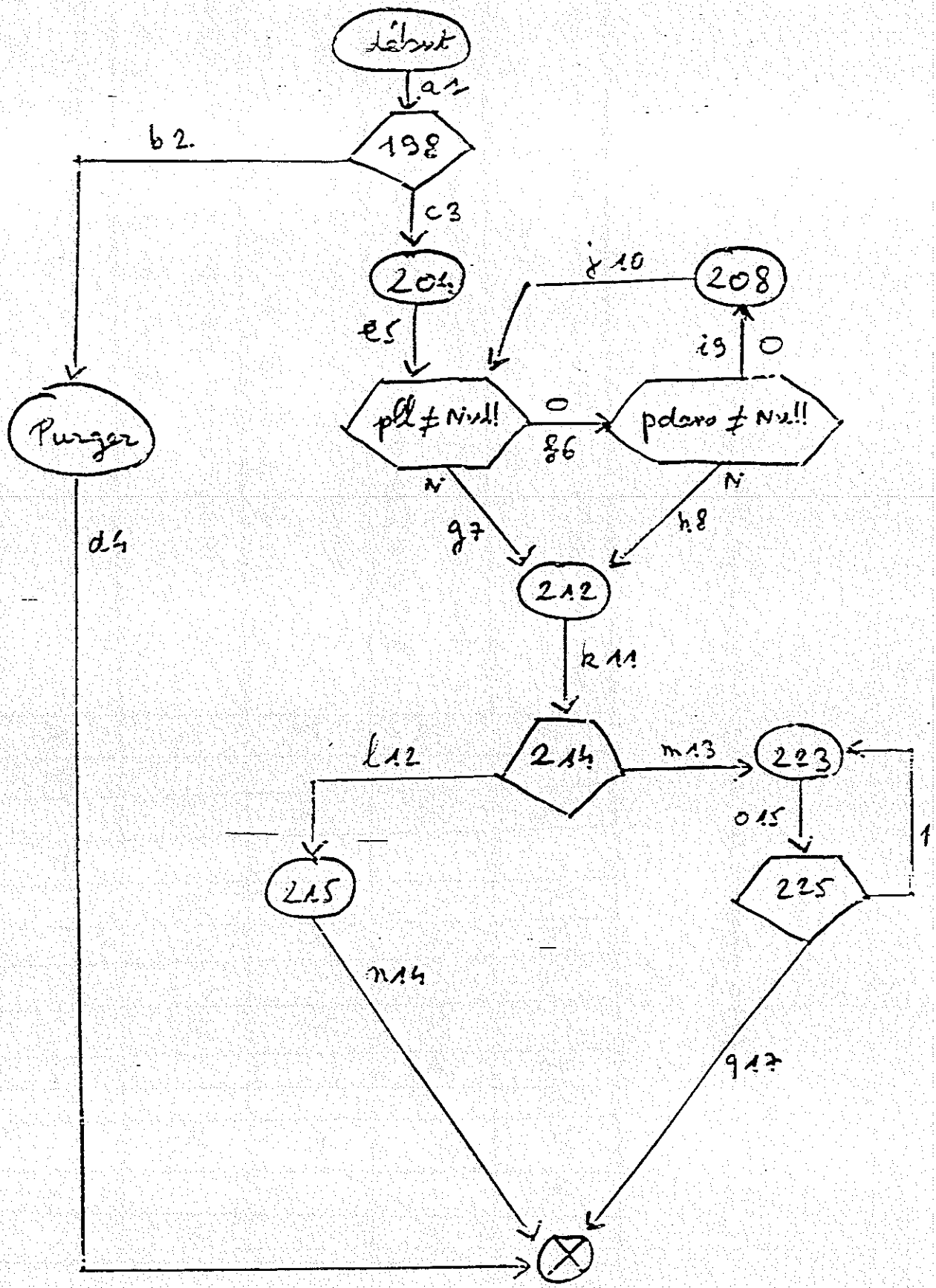
2) Calculer son nombre cyclomatique. Comment peut-on qualifier la complexité de cette procédure ?

3) En déduire une politique de tests permettant une couverture C2 (100 % des branches). Préciser la sémantique de chaque donnée utilisée pour le test (par exemple : liste vide, etc...)

Qu'en conclure par rapport au nombre cyclomatique ?

P.J. : COPIER

```
192     -- Affectation: DANS := LA_LISTE;
193
194     procedure COPIER (LA_LISTE: liste; DANS: in out liste) is
195       pll,pdans,pds_fin:ptr;
196       ldummy:liste;
197     begin
198       if EST_VIDE (La_Liste) then
199          PURGER (Dans);
200       else                          -- { La_Liste non vide. }
202          -- Affectation "sur place" entre cellules de meme rang:
203
204          pll:=La_Liste.debut; pdans:=Dans.debut;
205          pds_fin:=null;
206
207          while pll /= null and pdans /= null loop   -- { pll<>null }
208            pdans.contenu:=pll.contenu;
209            pds_fin:=pdans;
210            pll:=pll.suivant; pdans:=pdans.suivant;
211          end loop;
212          Dans.fin :=pds_fin;
213
214          if pll = null then          -- Tronquer Dans (la + longue):
215             ldummy.debut:=pdans;
216             PURGER (ldummy);         -- utilise l'implementation.
217             pds_fin.suivant:=null;-- marquage fin de liste (Dans).
218             Dans.long:=LONGUEUR (La_Liste);
219             PREMIER (Dans);          -- definit la TLE de Dans.
220
221          else     -- Completer Dans en l'allongeant:
222             loop
223               PROLONGER (Dans, pll.contenu);
224               pll:=pll.suivant;
225               exit when pll = null;
226             end loop;
227          end if;
228       end if;
229     end COPIER;
```



$M = A - N + 2p$

$= 17 - 13 + 2 = 6$

$= 5 + 1$

$M < 10$

$\Rightarrow$ SIMPLE % Test

CHEMINS	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
a b d La_liste Vide	X	X		X													
a c e g h k m o q La_liste Non Vide 1 élément, Dans Vide	X		X		X	X		X			X		X		X		X
a c e g h k m o p o q La_liste Non Vide 2 éléments, Dans Vide	X		X		X	X		X			X		X		X	X	X
a c e {g i j} g h k m o {p o} q Dans non Vide, + courte que la_liste non Vide	X		X		X	X		X	X	X	X		X		X	X	X
a c e {g i j} g k l m Dans non Vide, + longue que la_liste non Vide	X		X		X	X	X		X	X	X	X		X			

∃ boucles ⇒ Nb. de chemins (5) < au maximum N (6)

EXAMEN DE GENIE LOGICIEL

2ème session du
12 septembre1995

3 sujets (Durée 3 h.)

1. Exercice de conception

2 . Exercice de tests

3 . Question de cours

Tous les documents sont autorisés.

Notation :

- Exercice de tests sur 6 pts

Exercice de tests

La procédure ADA suivante (INSERER) implémente l'insertion d'un doublet (CLE, Valeur) dans un arbre binaire de recherche équilibré en hauteur (arbre AVL).

1) Etablir le graphe de commande de chacune des 2 procédures RECOLLER et INSERER.

N.B. : On assimilera un RAISE (susciter une exception) à un RETURN simple.

2) En déduire le nombre cyclomatique de INSERER. Que peut-on dire de la complexité de cette procédure ?

3) Définir une politique de tests permettant une couverture C2 (100% des branches).

N.B : On ne tiendra pas compte de RECOLLER pour cette question.

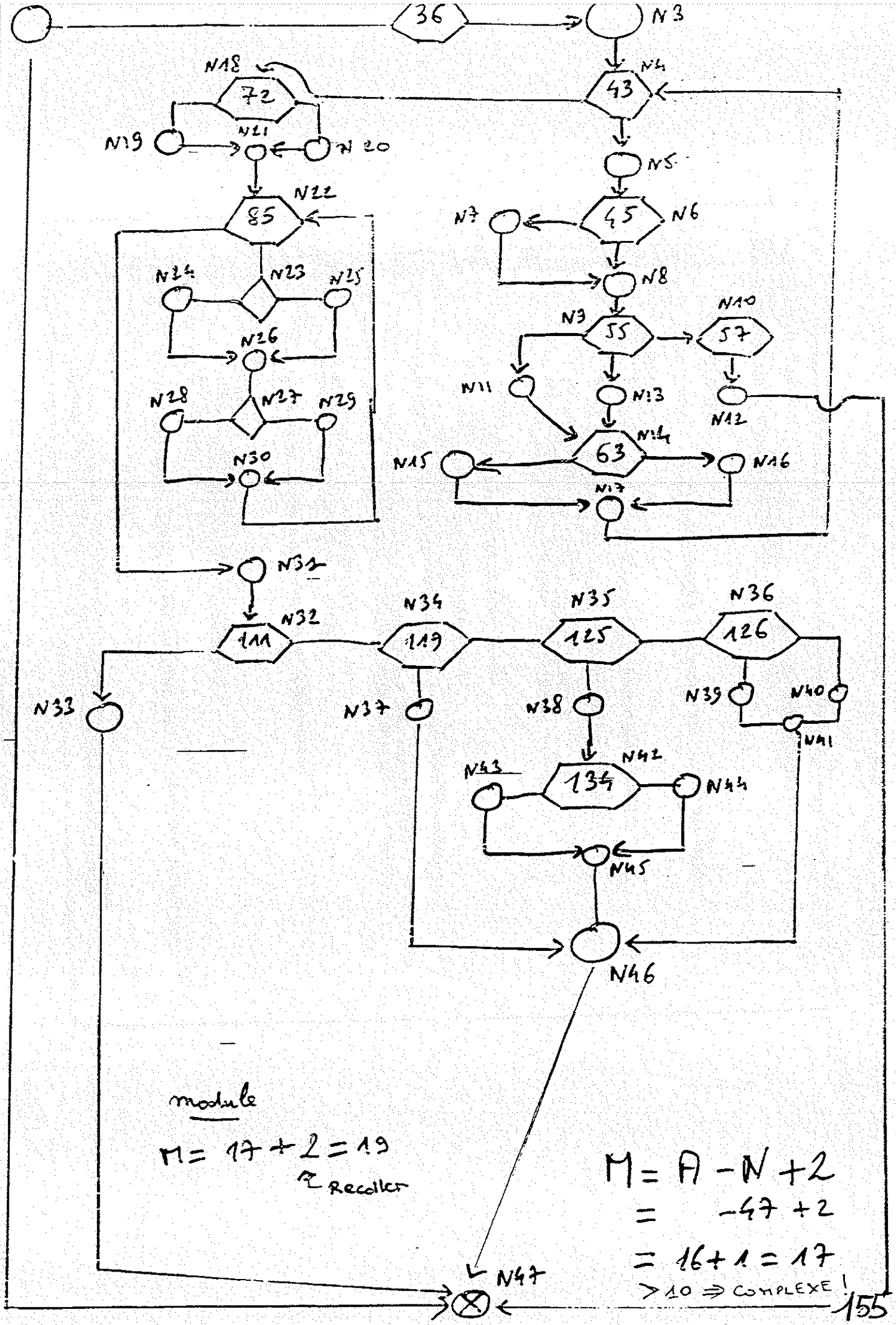
Qu'en conclure par rapport au nombre cyclomatique ?

INSERER Type TABLE geree par arbre equilibre en hauteur 17:36:34

```
 2  --*************************************************
 3  --*    INSERTION  DANS  UNE  TABLE  A. V. L.      *
 4  --*          (version iterative sans pile)        *
 5  --*************************************************
 6
 7  with FILES_BOOL;
 8  separate (TABLES_AVL)
 9
10  procedure INSERER (DANS:in out TABLE; VAL:VALEUR; CLE:CLEE) is
11      P_J_Deseq: TABLE;              -- plus jeune ancetre desequilibre.
12      elt_insere:constant element:=((val,0),cle);
13      elt_courant:element;
14      use FILES_BOOL;
15      f_mvts:FILE;
16
17      dr_p_pjd,premier_mvt,dr_desc,dr_pjd_f,tous_equil,fils_pjd,
18              pf_pjd: boolean:=true;
19      deseq_courant,desequil,deseq_f_pjd,des_ganew,des_drnew:signe;
20      deseq_pjd, d_pfpjd: signe:=0;
21
22      procedure RECOLLER is
23      begin
24          if dr_p_pjd then
25             D_COLLER (Dans,P_J_Deseq);
26          else
27             G_COLLER (Dans,P_J_Deseq);
28          end if;
29      end;
30
31  begin
```

```
33      -- Insertion proprement dite:
34      -- ============================
35
36        if EST_VIDE (Dans) then
37           D_PROLONGER (Dans,elt_insere);
38           return;
39        end if;
40
41        RACINE (Dans);
42        COUPER (Dans, Pour => P_J_Deseq);
43        while not EOA (P_J_Deseq) loop
44           elt_courant:=LIRE (P_J_Deseq);
45           if elt_courant.val.hg_hd /= 0 then
46              deseq_pjd :=elt_courant.val.Hg_hd;
47              tous_equil:=false;
48              Recoller;
49              COUPER (Dans, Pour => P_J_Deseq);
50              PURGER (f_mvts);
51              dr_p_pjd:=dr_desc; premier_mvt:=true;
52           end if;
53
54           dr_desc := elt_courant.key < cle;
55           if dr_desc then
56              A_DROITE (P_J_Deseq);
57           elsif cle < elt_courant.key then
58              A_GAUCHE (P_J_Deseq);
59           else
60              Recoller;
61              raise CLEF_PRESENTE;
62           end if;
63           if premier_mvt then
64              dr_pjd_f:=dr_desc;
65              premier_mvt:=false;
66           else
67              ENFILER (f_mvts,dr_desc);     -- a partir fils(rac(PJD)).
68           end if;
69        end loop;
70        -- (On est descendu au lieu d'insertion.)
71
72        if dr_desc then
73           D_PROLONGER (P_J_Deseq,elt_insere);
74        else
75           G_PROLONGER (P_J_Deseq,elt_insere);
76        end if;
77
78
79      -- Calcul des nouveaux "poids" sur le chemin d'insertion:
80      -- ======================================================
81
82        desequil:= -2*boolean'pos(dr_pjd_f)+1;          --{ +1 = gauche }
83        RACINE (P_J_Deseq);
84        deseq_courant:=desequil;
85        while not EST_VIDE (f_mvts) loop
86           if deseq_courant < 0 then
87              A_DROITE (P_J_Deseq);
88           else
89              A_GAUCHE (P_J_Deseq);
90           end if;
91           deseq_courant:= -2* PREMIER (f_mvts) + 1;
92           DEFILER (f_mvts);
93           Nouv_Equil (P_J_Deseq,deseq_courant);
94           if fils_pjd then
95              deseq_f_pjd:=deseq_courant;
96              fils_pjd:=false;
97           elsif pf_pjd then
98              d_pfpjd := deseq_courant;
99              pf_pjd:=false;
100          end if;
101       end loop;
```

```
104      -- Controle de l'equilibrage du + jeune ancetre desequilibre:
105      -- ============================================================
106
107      RACINE (P_J_Deseq);
108
109      -- 1er cas:aucun desequilibre initial sur le chemin d'insertion:
110
111      if deseq_pjd = 0 then
112         Nouv_Equil (P_J_Deseq, desequil);
113         COUPER (P_J_Deseq,Dans);
114         return;
115      end if;
116
117      -- 2eme cas: l'insertion ameliore l'equilibre:
118
119      if deseq_pjd + desequil = 0 then
120         Nouv_Equil (P_J_Deseq,0);
121
122      -- 3eme cas: L'insertion aggrave le desequilibre anterieur:
123      -- on retablit localement l'equilibre par rotation(s).
124
125      elsif deseq_f_pjd = desequil then   -- descente GG ou DD.
126            if desequil = 1 then
127               GG_rotation (p_j_deseq,0,0);
128            else
129               DD_rotation (p_j_deseq,0,0);
130            end if;
131      else
132         des_ganew:= (abs(d_pfpjd)-d_pfpjd)/2;
133         des_drnew:=-(abs(d_pfpjd)+d_pfpjd)/2;
134         if deseq_pjd > 0 then
135            D_G_Rotation (P_J_Deseq,des_ganew,des_drnew);
136         else
137            G_D_Rotation (P_J_Deseq,des_ganew,des_drnew);
138         end if;
139      end if;
140
141      Recoller;   --.le sous-chemin d'insert. a l'arbre principal.
142
143   end INSERER;


Procedure body TABLES_AVL.INSERER added to library
  CPU Time:              00:00:03.53    (2430 Lines/Minute)
  Compilation Complete
```



module

$M = 17 + 2 = 19$

↑ Recollet

$M = A - N + 2$

$= -47 + 2$

$= 16 + 1 = 17$

$> 10 \Rightarrow$ COMPLEXE !

EXAMEN DE GÉNIE LOGICIEL

1ère session du 14 février 1996

Exercice de tests

La procédure ADA suivante (constr-seq) implémente la création d'une matrice creuse (contenant beaucoup de zéros) représentée sous forme de multiliste chaînée.

1) Établir son graphe de commande

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

3) En déduire une politique de tests permettant une couverture C2 (100% des branches). Préciser la sémantique de chaque donnée utilisée pour le test (par exemple : matrice vide, etc....)

Qu'en conclure par rapport au nombre cyclomatique?

NB : on assimilera un RAISE (susciter une exception) à un RETURN simple

P.J. : CONSTR-SEQ (dans le module Matrice Creuse)

```
--=====================================
--=   Gestion de matrices creuses.    =
--=====================================

GENERIC

  -- Description de l'ens. des elts des matrices:

  TYPE valeur IS PRIVATE;          -- def. de l'ensemble.
  WITH FUNCTION "+" (v1,v2:valeur) RETURN valeur IS<>;
  zero:valeur;                     -- elt neutre de +

  base:NATURAL:=1;                 -- indice min. des matr.

PACKAGE matrice_creuse IS

  TYPE matrice(max_lig,max_col:POSITIVE)
               IS LIMITED PRIVATE;

-- 1 MATRICE est de type logique:
--    array (BASE:MAX_LIG, BASE:MAX_COL) of VALEUR;
-- Toute matrice est nulle lors de sa declaration.

-------------------------------------------------

PRIVATE
  TYPE maillon;
  TYPE ptr IS ACCESS maillon;
  TYPE ptab IS ARRAY(NATURAL RANGE<>) OF ptr;

  TYPE matrice (max_lig,max_col:POSITIVE) IS RECORD
          tetlig:ptab(base..max_lig):=(OTHERS=>NULL);
          tetcol:ptab(base..max_col):=(OTHERS=>NULL);
       END RECORD;

END matrice_creuse;

-- Corps du module:

PACKAGE BODY matrice_creuse IS

  TYPE maillon IS RECORD
       pligsuiv,pcolsuiv:ptr;
       lig,col:NATURAL;            -- coord. de la case.
       val:valeur;
       END RECORD;

-- ===   Procedures locales:   ===--

-- Proc generique construisant sequentiellement 1 matrice,
-- colonne par colonne:

  GENERIC
    WITH PROCEDURE init (nb_lig,nb_col:OUT NATURAL);
    WITH PROCEDURE initcol(col:NATURAL;EoCol:OUT BOOLEAN);
    WITH PROCEDURE avancer_col(EoCol:OUT BOOLEAN);
    WITH PROCEDURE getval (lig:OUT NATURAL;val:OUT valeur);
  PROCEDURE constr_seq (mdest:IN OUT matrice);

  PROCEDURE constr_seq (mdest:IN OUT matrice) IS
    p,p0,p_l_c:ptr;
    nb_lig,nb_col,lig:NATURAL;
    val:valeur;
    EoCol:BOOLEAN;
  BEGIN
```