POLYTECHNIQUE MONTRÉAL

Questionnaire
Contrôle Périodique 3

LOG3430

Sigle du cours

Identification de l'étudiant(e)		
Nom :	**Prénom :**	
Signature :	**Matricule :**	**Groupe :**

Sigle et titre du cours	*Groupe*	*Trimestre*
LOG3430 - Méthodes de test et de validation du logiciel	**Tous**	**20163**
Professeur	***Local***	***Téléphone***
Giuliano Antoniol	**L-2204**	

Jour	*Date*	*Durée*	*Heures*
Mercredi	**19 octobre 2016**	**1 heure**	

Documentation	*Calculatrice*	
☐ Aucune ☒ Toute ☒ Voir directives particulières	☐ Aucune ☒ Toutes ☐ Non programmable	**Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.**

Directives particulières

Toute documentation est permise, ainsi que les calculatrices, tablettes et ordinateurs à l'exception toutefois de tout dispositif connecté à Internet.

Important

Cet examen contient **1** exercice et **5** question sur un total de **6** pages **(excluant cette page)**

La pondération de cet examen est de **5** **%**

Vous devez répondre sur : ☒ le questionnaire ☐ le cahier ☐ les deux

Vous devez remettre le questionnaire : ☒ oui ☐ non

L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

Exercice 1 – 20 points

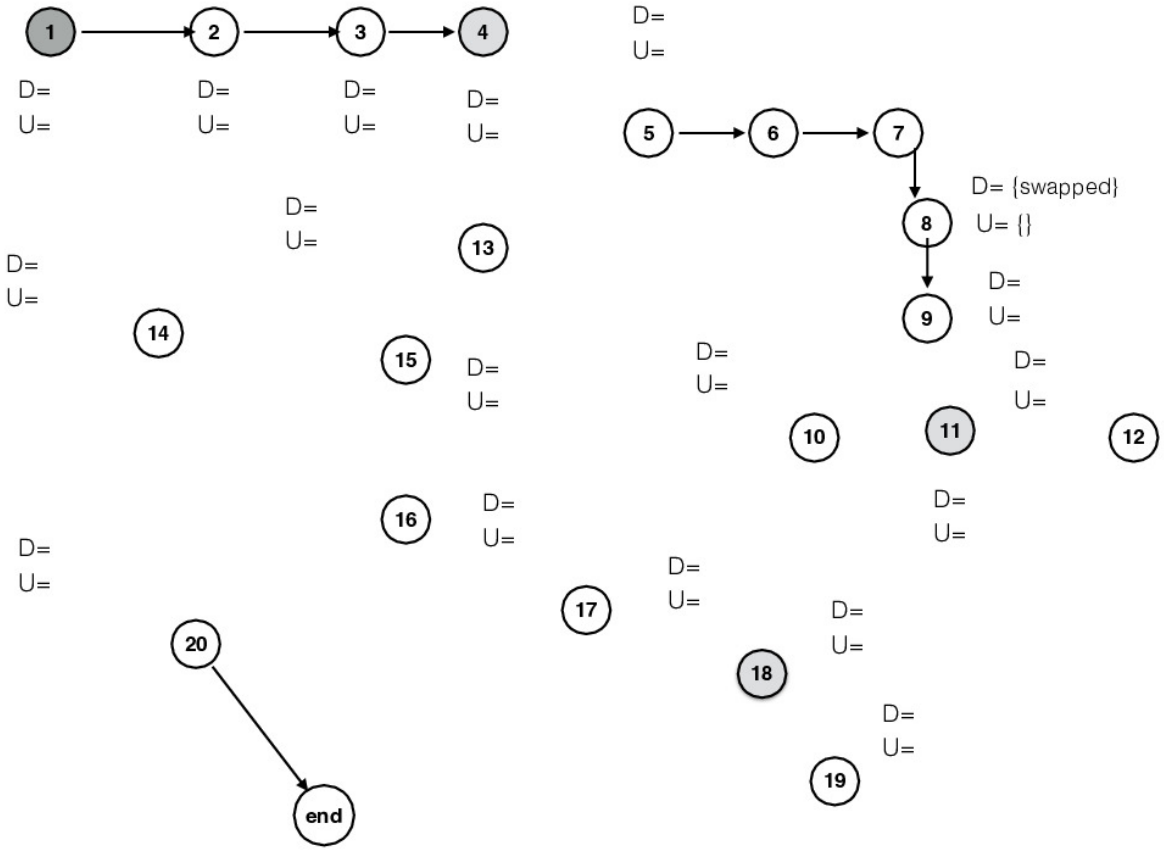
Considérez le programme suivant :

```
void  swap(int  *i, int *j) {  // line 1
  int t = *i; // line 2
  *i = *j; // line 3
  *j = t; // line 4
}


void cocktailShakerSort(int *  A, int size){ // line 5
int swapped, i; // line 6
  do{  // line 7
    swapped = 0; // line 8
    for (i =0 ; i < size - 2; i++) // line 9
      if (A[ i ] > A[ i + 1 ]){  // line 10
        swap( A + i , A + i + 1  ); // line 11
     swapped =1; // line 12
      }

    if (swapped ==0) // finish if no swaps occurred. -- line 13
      break; // line 14

    swapped = 0; // line 15
    for (i = size  - 2 ; i> 0; i--)  // line 16
      if (A[ i ] > A[ i + 1 ]) { // line 17
        swap(A + i , A+ i + 1  ); // line 18
     swapped = 1; // line 19
      }

  }while (swapped); // if no elements have been swapped, then it is sorted -- line 20
}
```

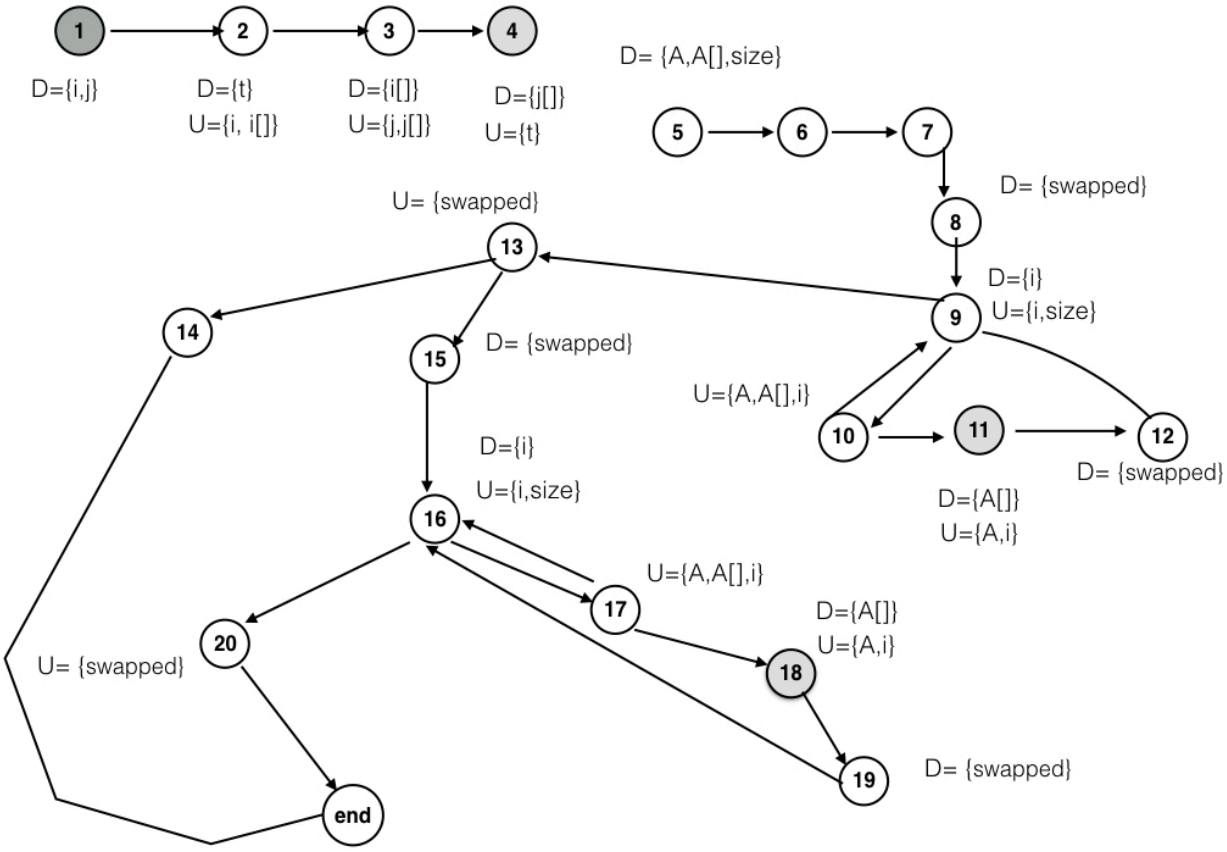
Q1 : Complétez le *Graphe de flux de contrôle* suivant, où les numéros de nœuds correspondent aux numéros de lignes. Les nœuds d'appel, entrées et sorties, sont en gris; le nœud end indique la fin du calcul et donc le point de return de la procédure `cocktailShakerSort`. Indiquez :

a. les arcs du graphe; (1 point)
b. les ensembles des définitions et utilisations pour chaque nœud; voir l'exemple pour nœud 8 qui définit la variable 'swapped' et dont l'ensemble d'utilisations est vide. (1 point)

Réponse à la question 1.1



Solution:



Please notice it would be legitimate to model the call site considering also the use of the generic element of A as to define A via its address swap uses A's content.

Q2 : Complétez le tableau suivant en donnant les uses pour chaque variable. Voir l'exemple pour la variable 'j' qui a une utilisation à la ligne 3 ; remarquez que i est l'adresse base du tableau et i[] l'élément générique (2 points).

	swap					cocktailShakerSort				
Ligne/var	I	I[]	j	J[]	t	swapped	i	size	A	A[]
1										
2										
3			x	x						
4										
5										
6										
7										
8										
9										
10										
11										
12										
13										
14										
15										
16										
17										
18										
19										
20										

Solution :

	swap					cocktailShakerSort				
Ligne/var	I	I[]	j	J[]	t	swapped	i	size	A	A[]
1	x	x								
2										
3			x	x						
4					x					
5										
6										
7										
8										
9							x	x		
10							x		x	x
11							x		x	o
12										
13						x				
14										
15										
16							x	x		
17							x		x	x
18							x		x	o
19										
20						x				

Q3 : Complétez le tableau suivant en donnant toutes les définitions-utilisations (def-uses) des données. Voir l'exemple pour la variable 'size' dont la définition à la ligne 5 est utilisée à la ligne 9; compléter la cellule 5/9 si nécessaire. (2 points)

	Ligne de définition																		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
l																				
i[]																				
j																				
j[]																				
t																				
swapped																				
i																				
size					9,…															
A																				
A[]																				

Solution :

	Ligne de définition																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
I	2,3																			
i[]	2																			
j	3,4																			
j[]	3																			
t		4																		
swapped								13				13			20				20	
i									9, 10, 11							16, 17, 18				
size					9, 16															
A					10, 11, 17, 18															
A[] (1)			10, 17, 2,3	10, 17, 2,3	10, 17, 2,3 or 11, 18						10, 11, 17, 18							10, 11, 17, 18		

The array A is actually passed to swap, thus the answer may depend of how swap is modeled. There are two options 1) we enter swap and we notice that actually A[] is i[] and j[] 2) we model swap as a black box. In the first case A[] is also uses in 2 and 3, Plus A[] definition of 2 and 3 propagate to 10 and 17 as well as back to 2 and 3. But one may also say swap is a black box and in this case it is the line 11 and 18 thet define the generic content of A[].

Q4 : Pour la procédure `cocktailShakerSort,` complétez le tableau suivant en donnant des valeurs d'entrée pour couvrir le critère all-uses. Précisez les def-uses couverts pour chaque valeur d'entrée. Voir l'exemple du cas de test T1; chaque paire def-use est indiqué <def,use>; donc pour la variable swapped, <8,13> indique que swapped est définie à 8 et utilisée à la ligne 13. (8 points)

T.C.	size	A[0]	A[1]	A[2]	A[3]	swapped	i	size	A	A[]
T1	1	1				<8,13>				

Solution :

We have ti provide : Some definition-clear sub-path from each definition to each use reached by that definition (and each successor node of the use). This actually means every computation and branch directly affected by a definition is exercised. There are may ways to cover the various def uses. A problem here is to realize that the 2 loops sort two by two thus to expose certain def-use pairs we need to traverse multiple times the two loops. For example one may use the defauls 1, then 1 2 3 and 4 3 2 1.

T.C.	size	A[0]	A[1]	A[2]	A[3]	swapped	i	size	A	A[]
T1	1	1				<8,13>	<9,9>	<5,9>		
T2	3	1	2	3			<9,10>		<5,10>	<5,10>
T3	4	4	3	2	1	<12,20> <19,20>	<9,11> <16,16> <16,17> <16,18>	<5,16>	<5,11> <5,17> <5,18>	<5,11> <11,10> <11,11> <11,17> <11,18> <18,10> <18,11> <18,17> <18,18>

Q5 : Est-ce que la fonction `cocktailShakerSort` contient un défaut ou plus ? Si oui, fournir un cas de test suffisant à l' (les)exposer ; il faut justifier la réponse et expliquer 1) ou est l'erreur ; 2) la raison probable et 3) la correction suggérée. (6 points)

Solution :

Yes there are two conceptual problems. Since the algorithm works two by two the loop at line 9 misses the last comparison and thus it does not sort something like 1 2 1. The fi xis easy :

For (i=0 ;i< size -1 ; i++) or better For (i=0 ;i<=size -2 ; i++)

The developer forgot the = in the comparison. Much in the same way the forst two elments risk not to be compares see for example for 3 2 1 4 again the problem is an equal in the stop condition. The fix is :

For (i=size -2 ;i>=0 ; i--)