NAME NOM	COLLEGE NUMBER NUMÉRO DE COLLÈGE

Final Exam
Theoretical Part

EEE243 Applied Computer Programming
7 December 2017, 1300 – 1600 hrs

Examiner: Dr. Tashfeen Karamat, PhD

Instructions:

- **Do not turn this page until instructed to do so.**
- You have 180 minutes to complete both parts of the exam.
- This exam has two parts.
    - The *theoretical* portion of the test is closed-book, out of a total of 35 marks.
    - The *practical* portion of the test is open-book (you may have course notes and textbook but you may **NOT** access the internet), and is worth 65 marks.
- There are 20 multiple choice questions, 20 True/False questions each worth 0.5 marks
- The 5 short answer questions are worth the points indicated in the center column.
- You must hand in the *theoretical* exam before you can begin the *practical* portion of the exam.
- Answer all the questions directly on the questionnaire.
- Immediately fill out your name, college number and the number of your workstation on the information card.
- Write your name on the questionnaire
- If a question seems unclear, make a *reasonable* assumption, document it, and answer the question as though the assumption were correct. *The examiners will not clarify the meaning of questions during the test*.
- Good luck!

Examen final
Partie théorique

GEF243 Programmation informatique appliquée
7 décembre 2017, 13h00–16h00

Examinateur: Capt Adrien Lapointe, CD, MSc

Instructions:

- **Ne tournez pas cette page avant l'instruction de l'examinateur.**
- Vous avez 180 minutes pour compléter les deux parties de l'examen.
- Cet examen a deux parties.
    - La partie *théorique* est à livre fermé et compte pour 35 points.
    - La partie *pratique* est à livre ouvert (vous avez droit à vos notes, à votre livre mais vous ne pouvez **PAS** accéder à l'Internet), et compte pour un total de 65 points.
- Il y a 20 questions à choix de réponse et 20 questions Vrai/Faux valant chacune 0,5 points.
- Les 5 questions à réponse courtes ont les valeurs indiquées dans la colonne centrale.
- Vous devez remettre l'examen *théorique* avant de pouvoir commencer la partie *pratique*.
- Répondez à toutes les questions directement sur le questionnaire d'examen.
- Écrivez immédiatement votre nom, numéro de collège et numéro de station de travail sur la carte d'information.
- Écrivez votre nom sur le questionnaire.
- Si une question ne vous semble pas claire, faites des suppositions raisonnables, documentez-les et répondez à la question en tenant compte des suppositions. *Les examinateurs ne clarifieront pas le sens des questions pendant le test*.
- Bonne chance!

Beginning of THEORETICAL Portion of Exam

Début de la partie THÉORIQUE de l'examen

MULTIPLE CHOICE QUESTIONS / QUESTION À CHOIX DE RÉPONSE

Important Note: Some questions have five choices

Circle the correct answer directly on the exam.

1. The short form of incrementing a variable `x` is:

   a. `x=x+1`
   b. `x++`
   c. `x+x`
   d. Both b and c are correct.

2. Given that variable `num` is an integer type, which of the following is the right way to get input from a user:

   a. `scanf("%d",num);`
   b. `scanf("%c",&num);`
   c. `scanf("num",%d);`
   d. `scanf("%d",&num);`
   e. none of the above

3. Infinite `while` loop can be written as:

   a. `while(1)`
   b. `while(0)`
   c. `while(∞)`
   d. `while(infinite)`

4. Which of the following is a valid declaration of a function which takes a two dimensional array as an argument:

   a. `void hello(int table[][]);`
   b. `void hello(int table[][5]);`
   c. `void hello(int table[]);`
   d. `void hello(int table[5]);`

5. If function is declared as

   ```
   float square(float num);
   ```

   What is the correct way to call it in the `main` function?

   a. `float square(num);`
   b. `square(float num);`
   c. `square (num);`
   d. `float square(float num);`
   e. none of the above

Note importante: Certaines questions ont cinq choix

Encerclez la bonne réponse directement sur l'examen.

1. La forme abrégée pour incrémenter la variable `x` est :
   a. `x=x+1`
   b. `x++`
   c. `x+x`
   d. Les choix b et c sont corrects.

2. Étant donné que la variable `num` est de type entier, lequel des énoncés suivants est le bon moyen d'obtenir une entrée de l'utilisateur:

   a. `scanf("%d",num);`
   b. `scanf("%c",&num);`
   c. `scanf("num",%d);`
   d. `scanf("%d",&num);`
   e. Aucun de ces choix

3. Une boucle sans fin `while` peut être écrite comme :
   a. `while(1)`
   b. `while(0)`
   c. `while(∞)`
   d. `while(infinite)`

4. Lequel des énoncés suivants est une déclaration valide d'une fonction qui prend un tableau à deux dimensions comme argument:

   a. `void hello(int table[][]);`
   b. `void hello(int table[][5]);`
   c. `void hello(int table[]);`
   d. `void hello(int table[5]);`

5. Si une fonction est déclarée comme

   ```
   float square(float num);
   ```

   Quelle est la bonne façon de l'appeler à partir de la fonction `main` ?

   a. `float square(num);`
   b. `square(float num);`
   c. `square (num);`
   d. `float square(float num);`
   e. Aucun de ces choix

6. Which of the following is not a valid declaration of an array in C:

   a. `int scores[]={3, 4, 44, 54};`
   b. `char first_names[25];`
   c. `int grades[3,4,5];`
   d. `float averages[12];`

7. When passing an array to a function:

   a. the values of the arrays are passed
   b. addresses of the values of all array elements are passed
   c. address of first element of the array is passed
   d. none of the choices is correct

8. Which of the following is not a correct way of initializing strings:

   a. `char str[5] = "Good";`
   b. `char str[5] = {'G','0','0','d','\0'};`
   c. `char str[4] = {'G','0','0','d','\0'};`
   d. `char str[]= "Good";`
   e. b and c

9. Which one of the following is a valid way of printing a string contained in an array called `str`:

   a. `printf("Message: %s\n", str);`
   b. `printf("Message: %c\n", str);`
   c. `printf("Message: %s\n", str[]);`
   d. `printf("Message: str\n", %c);`

10. We can pass parameters by reference to function using:

    a. Arrays
    b. Pointers
    c. `Malloc()`
    d. Both a and b

6. Lequel des énoncés suivants n'est pas une déclaration valide d'un tableau en C:

   a. `int scores[]={3, 4, 44, 54};`
   b. `char first_names[25];`
   c. `int grades[3,4,5];`
   d. `float averages[12];`

7. Lors du passage d'un tableau à une fonction:

   a. les valeurs des tableaux sont passées
   b. l'adresse des valeurs de tous les éléments du tableau est passée
   c. l'adresse du premier élément du tableau est passée
   d. Aucun de ces choix n’est correct

8. Lequel des énoncés suivants n'est pas un moyen correct d'initialiser des chaînes de caractères:

   a. `char str[5] = "Good";`
   b. `char str[5] = {'G','0','0','d','\0'};`
   c. `char str[4] = {'G','0','0','d','\0'};`
   d. `char str[]= "Good";`
   e. b et c

9. Lequel des énoncés suivants est une manière valide d’afficher une chaîne de caractères contenue dans un tableau appelé `str` :

   a. `printf("Message: %s\n", str);`
   b. `printf("Message: %c\n", str);`
   c. `printf("Message: %s\n", str[]);`
   d. `printf("Message: str\n", %c);`

10. Nous pouvons passer un paramètre par référence à une fonction en utilisant :

    a. Tableaux
    b. Pointeurs
    c. `Malloc()`
    d. Les réponses a et b

11. After execution of the following code, what value will variable `b` contain

```
----------------------------
int a = 90;
int b = 12;
int *p = NULL;
int *r = NULL;
p = &a;
r = p;
b = *r;
------------------------------
```

a. `90`
b. `12`
c. Address of `r`
d. Address of `p`

12. Given that `float` type takes 4 bytes of memory, the following code moves the pointer `p` ahead by:

```
-------------------------
float x = 52.32;
float *p = &x;
p=p+2;
-------------------------
```

a. Two bytes
b. Two bits
c. Four bytes
d. Eight bytes

13. Which of the following statements is NOT a valid declaration of enumerated types?

a. `enum Months {JAN , FEB, MAR};`
b. `enum Months {JAN=1, FEB=2, MAR=3};`
c. `enum Months {JAN; FEB; MAR};`
d. `enum Months {JAN = 1, FEB, MAR};`

14. Use of structures is especially useful when you want to store related elements of:

a. Same type
b. Different types
c. Type `char`
d. Type `int`

15. Given the following structure definition:

```
struct Student {
   char first_name[15];
   char last_name[25];
   char college_number[6];
   float average;
};
```

11. Après l'exécution du code suivant, quelle valeur la variable `b` contiendra-t-elle?

```
----------------------------
int a = 90;
int b = 12;
int *p = NULL;
int *r = NULL;
p = &a;
r = p;
b = *r;
------------------------------
```

a. `90`
b. `12`
c. Adresse de `r`
d. Adresse of `p`

12. Étant donné que le type `float` prend 4 octets de mémoire, le code suivant avance le pointeur `p` de :

```
-------------------------
float x = 52.32;
float *p = &x;
p=p+2;
-------------------------
```

a. Deux octets
b. Deux bits
c. Quatre octets
d. Huit octets

13. Lequel des énoncés suivants n'est PAS une déclaration valide d'un type énuméré ?

a. `enum Months {JAN , FEB, MAR};`
b. `enum Months {JAN=1, FEB=2, MAR=3};`
c. `enum Months {JAN; FEB; MAR};`
d. `enum Months {JAN = 1, FEB, MAR};`

14. L'utilisation de structures est particulièrement utile lorsque vous souhaitez stocker des éléments apparenté de :
a. Même type
b. Différent type
c. Type `char`
d. Type `int`

15. Étant donné la définition de structure suivante :

```
struct Student {
   char first_name[15];
   char last_name[25];
   char college_number[6];
   float average;
};
```

What is the valid way of declaring a variable `a_student` based on the above structure?

a. `struct Student a_student;`
b. `Student a_student;`
c. `struct a_student;`
d. `struct Student;`

16. Any field of a structure can individually be accessed using

a. Assignment operator (=)
b. Equal operator (==)
c. Dot operator (.)
d. None of the choices is correct

17. The elements in a linked list are referred to as:

a. Fields
b. Bytes
c. Members
d. Nodes

18. Linked lists are stored in:

a. Stack
b. Heap
c. IDE
d. Text

19. The same *LL* can be ordered:

a. In two different ways
b. In ascending order only
c. In descending order only
d. In multiple different ways

20. Which of the following is a valid way of declaring `fp_store` as a pointer to a function which takes two integer parameters and returns a `char`:

a. `char (* fp_ints) (int, int);`
b. `char * fp_ints (int, int);`
c. `char fp_ints (int, int);`
d. `char (* fp_ints) (int*, int*);`

Quelle est une façon correcte de déclarer une variable `a_student` basée sur la structure ci-dessus ?

a. `struct Student a_student;`
b. `Student a_student;`
c. `struct a_student;`
d. `struct Student;`

16. Tout champ d'une structure peut être accédé individuellement en utilisant :

a. Opérateur d'assignation (=)
b. Opérateur d'égalité (==)
c. Opérateur point (.)
d. Aucun de ces choix n'est correct

17. Les éléments d'une liste chaînée sont appelés :

a. Champ
b. Octet
c. Membre
d. Nœud

18. Les listes chaînées sont stockées dans :

a. Pile
b. Tas
c. IDE
d. Texte

19. La même liste chaînée peut être triée :

a. De deux façons différentes
b. En ordre croissant seulement
c. En ordre décroissant seulement
d. De plusieurs façons

20. Laquelle des façons suivantes est valide pour déclarer le pointeur à une fonction `fp_store` qui prend deux paramètres entiers et renvoie un `char` :

a. `char (* fp_ints) (int, int);`
b. `char * fp_ints (int, int);`
c. `char fp_ints (int, int);`
d. `char (* fp_ints) (int*, int*);`

TRUE/FALSE QUESTIONS / QUESTIONS VRAI OU FAUX

21. Scope determines the duration of the existence of a variable.
T **F**

22. In C, `A==B` verifies that `A` is equal to `B`.
T **F**

23. `&&` is a bitwise operator.
T **F**

24. If you want to remember the value of a counter variable after leaving a function, you declare it as `static`.
T **F**

25. In a C program, the division 4/5 will give you 0.8.
T **F**

26. When passing a two-dimensional array to a function, you must specify both dimensions.
T **F**

27. Strings cannot be declared as pointers.
T **F**

28. `\0` is used as a logical marker in C to tell our functions the END of the string.
T **F**

29. The name of an array is a constant pointer.
T **F**

30. It is not possible to have a pointer to a function.
T **F**

31. When using `malloc()` for memory allocation, the compiler will warn you if there is not enough memory.
T **F**

32. When you *declare* a new variable of an enumerated type, it is optional to use the `enum` keyword.
T **F**

33. To re-define `int` type as `INT`, you can use following `typedef int INT;`
T **F**

34. `typedef` may be used to avoid repeating the `struct` keyword with each declaration of a structure type variable.
T **F**

21. La portée détermine la durée de l'existence d'une variable.

22. En C, `A==B` vérifie que `A` est égal à `B`.

23. `&&` est un opérateur bit par bit.

24. Si vous voulez mémoriser la valeur d'une variable compteur après avoir quitté une fonction, vous la déclarez `static`.

25. Dans un programme C, la division 4/5 donnera 0,8.

26. Lorsque vous passez en paramètre à une fonction un tableau à deux dimensions, vous devez spécifier les deux dimensions.

27. Les chaînes de caractères ne peuvent pas être déclarées comme pointeurs.

28. `\0` est utilisé comme marqueur logique en C pour indiquer à nos fonctions la FIN de la chaîne de caractères.

29. Le nom d'un tableau est un pointeur constant.

30. Il n'est pas possible d'avoir un pointeur à une fonction.

31. Lorsque vous utilisez `malloc()` pour l'allocation de mémoire, le compilateur vous avertira s'il n'y a pas assez de mémoire.

32. Lorsque vous *déclarez* une nouvelle variable d'un type énuméré, il est facultatif d'utiliser le mot clé `enum`.

33. Pour redéfinir le type `int` comme `INT`, vous pouvez utiliser `typedef int INT;`

34. `typedef` peut être utilisé pour éviter de répéter le mot-clé `struct` avec chaque déclaration de variable de type structure.

35. A *structure* is a data type whose format is defined by the programmer.

T
F

35. Une *structure* est un type de données dont le format est défini par le programmeur.

36. The following is a valid way of assigning a string to an array of character that is part of the structure `a_student`:

```
a_student.first_name="Bob";
```

T
F

36. Voici une façon valide d'affecter une chaîne de caractère à un tableau de caractères faisant parti de la structure `a_student`:

```
a_student.first_name="Bob";
```

37. Each node in a linked list has a symbolic name (variable name) associated with it.

T
F

37. Chaque nœud d'une liste chaînée est associé à un nom symbolique (nom de variable).

38. Essentially, linked lists functions as an array that can grow and shrink as needed.

T
F

38. Essentiellement, les listes chaînées fonctionnent un peu comme un tableau qui peut grossir et rétrécir au besoin

39. Unlike arrays, linked lists do not have indexes.

T
F

39. Contrairement aux tableaux, les listes chaînées n'ont pas d'indices.

40. It is impossible to have circular linked lists.

T
F

40. Il est impossible d'avoir des listes chaînées circulaires.

SHORT ANSWER QUESTIONS / QUESTIONS À RÉPONSE COURTE

41. Draw the memory model for C programs as seen in class to the right of the code below.

4

41. Dessinez le modèle de mémoire pour les programmes C tel que vu en classe à droite du code ci-dessous.

```
// Code Q42

int i = 0;

void main() {
   int j = 0;
   int k=5;
   j = j + k;
}

int fctn() {
   int l = 0;
   static int k = 2;
   k = k *k;
   int *p_int = (int*)malloc(sizeof(int));
}
```

Your memory model diagram
Votre diagramme du modèle de mémoire.

42. From the code above, identify one example of what would be contained in each memory segment. Draw arrows between your memory diagram and the code above to indicate the place of the element in the memory model.

4

42. À partir du code ci-dessus, identifiez un exemple de ce qui serait contenu dans chaque segment de mémoire. Dessinez des flèches entre votre diagramme de mémoire et le code ci-dessus pour indiquer la place de l'élément dans le modèle de mémoire.

43. What is Encapsulation in regard to programing?

3

43. Qu'est-ce que l'encapsulation en programmation ?

44. Consider the following code, which contains two errors. Indicate the line numbers where the errors are located.

2

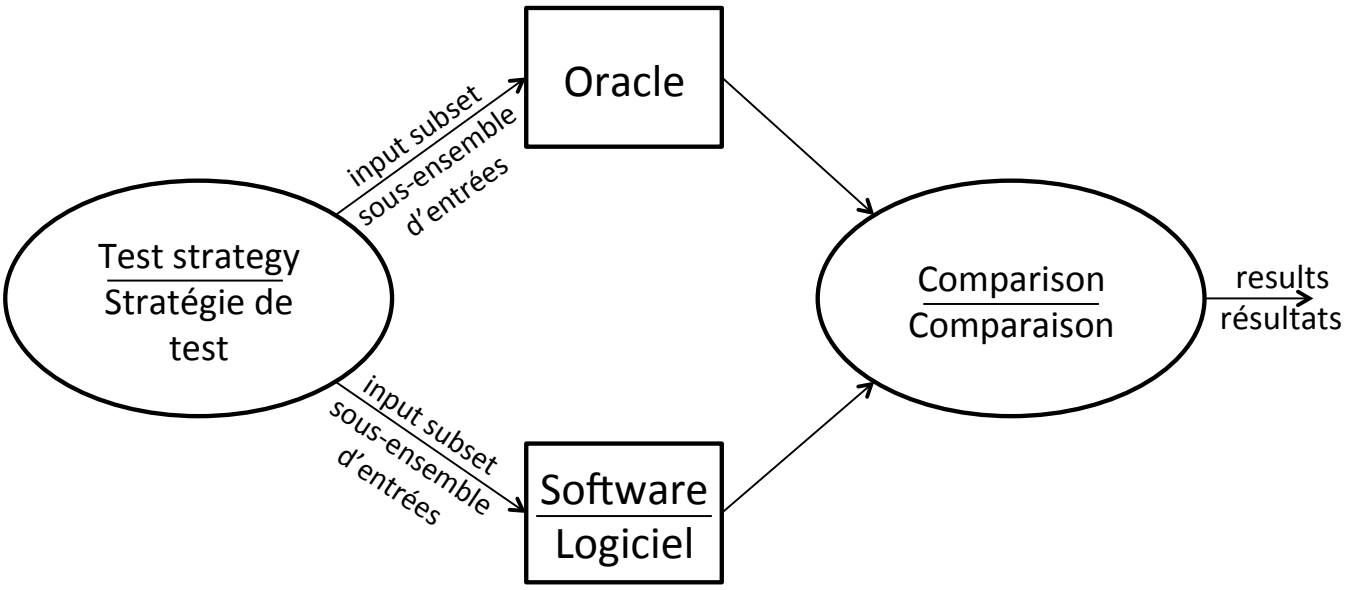
44. Considérez le code suivant contenant deux erreurs. Indiquez les numéros de ligne où se trouvent ces erreurs.

```
1. #include <stdio.h>
2. #include <stdlib.h>
3.
4. int add(int a, int b);
5. void sub(int r, int t, int *res)
6.
7. int main(void) {
8.       int x = 2;
9.       int y = 3;
10.      int z = add(x, y);
11.      int w;
12.
13.      sub(30, 20, w);
14.
15.      printf("Result of addition=%d\n",z);
16.      printf("Result of subtraction=%d\n",w);
17.      return EXIT_SUCCESS;
18. }
19.
20. int add(int a, int b) {
21.      return a + b;
22. }
23.
24. void sub(int r, int t, int *res) {
25.      *res = r - t;
26. }
```

45. The figure below illustrates the way most testing is done. Explain in one sentence what the purpose of the Oracle is.

2

45. La figure ci-dessous illustre la manière dont la plupart des tests sont effectués. Expliquez en une phrase quel est le but de l'Oracle.



End of THEORETICAL Portion of Test

Fin de la partie THÉORIQUE du test