Mid-Term Test
Theoretical Part

EEE243 Applied Computer Programming
27 October 2016, 08:00 – 10:50 hrs

Examiner: Dr. Sidney Givigi, PhD

Instructions:

- **Do not turn this page until instructed to do so.**
- You have 170 minutes to complete the test.
- Questions have the values indicated in the centre column.
- This test has two parts.
    - The *theoretical* portion of the test is closed-book, out of a total of 20 marks.
    - The *practical* portion of the test is open-book (you may have course notes and textbook and you may access the internet), and is worth 30 marks.
- You must hand in the *theoretical* exam booklet before you can begin the *practical* part of the test.
- Answer all questions in the test booklet.
- Immediately fill out your name, college number and the number of your workstation on the information card.
- 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!

Test de mi-session
Partie théorique

GEF243 Programmation informatique appliquée
27 octobre 2016, 08h00–10h50

Examinateur: Capt Adrien Lapointe, CD, MSc

Instructions:

- **Ne tournez pas cette page avant l'instruction de l'examinateur.**
- Vous avez 170 minutes pour compléter le test.
- Les questions ont les valeurs indiquées dans la colonne centrale.
- Ce test a deux parties.
    - La partie *théorique* est à livre fermé et compte pour 20 points.
    - La partie *pratique* est à livre ouvert (vous avez droit à vos notes, à votre livre et vous pouvez accéder à l'Internet), et compte pour un total de 30 points.
- Vous devez remettre le livret d'examen *théorique* avant de pouvoir commencer la partie *pratique*.
- Répondez à toutes les questions dans le livret d'examen.
- Écrivez immédiatement votre nom, numéro de collège et numéro de station de travail sur la carte d'information.
- 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 Test

1. Explain why an array is a derived type in C.

2. Explain why arrays could be treated as pointers and vice versa. Give code examples of how you could use a pointer to iterate over an array.

3. Explain when `continue` and `break` statements are used. Be precise.

4. The Taylor Series Expansion of a sine function is given by:

$$\sin(x) = \sum_{i=0}^{n} \frac{(-1)^i}{(2i+1)!} x^{2i+1}$$

Where x is given in radians.

You are asked to implement the sine function in C for any angle expressed in degrees. You already have a predefined constant for `PI`:

```
#define PI 3.14159
```

Provide C code for the `power` function x^y=`power(x,y)`, the `factorial` function x!=`factorial(x)` and the `sin` function of an angle in degrees as a series with n terms as described above. The functions should have the following prototypes:

a. `double power(double x, double y);`

b. `double factorial(int x);`

c. `double sin(double x, int n);`

Début de la partie THÉORIQUE du test

1 1. Expliquez pourquoi un tableau est un type dérivé en C.

2 2. Expliquez pourquoi les tableaux peuvent être traités comme des pointeurs et vice-versa. Donnez un exemple de code montrant comment un pointeur peut être utilisé pour parcourir un tableau.

2 3. Expliquez quand les énoncés `continue` et `break` sont utilisés. Soyez précis.

8 4. Le développement en série de Taylor d'une fonction sinusoïdale est donné par:

$$\sin(x) = \sum_{i=0}^{n} \frac{(-1)^i}{(2i+1)!} x^{2i+1}$$

Où x est donné en radians.

Vous devez implémenter la fonction sinusoïdale en C pour n'importe quel angle donné en degrés. Vous avez déjà une constante `PI` de définie :

Donnez le code C pour la fonction `power` x^y=`power(x,y)`, pour la fonction `factorial` x!=`factorial(x)` et pour la fonction `sin` d'un angle en degrés comme une série à n termes tel que décrit ci-dessus. Les fonctions devraient avoir les prototypes suivants :

a. `double power(double x, double y);`

b. `double factorial(int x);`

c. `double sin(double x, int n);`

5. Consider the following code. **4** 5. Considérez le code suivant.

```
1  #include <stdio.h>
2  #include <string.h>
3
4  // implement the Caesar Cipher
5  void cipher(char *str, int key);
6
7  int main(void) {
8        char str[] = "hello world";
9        cipher(&str, 4);
10       printf("Your encrypted text is: %s\n", str);
11       return 0;
12 }
13
14 void cipher(char *str, int key) {
15       if (key > 25)
16             key = key%25;
17       for (i=0; i<strlen(str); i++) {
18             if (str[i] == ' ')
19                   continue;
20             else {
21                   str[i] = str[i] + key;
22                   if (str[i] > 'z') {
23                         str[i] = 'a' - 1 + str[i] - 'z';
24                   }
24             }
25       }
26 }
```

The intent of the call to `cypher` on line 9 is that the content of `str` will be encrypted according to the Caesar Cipher. The Caesar Cipher offsets the characters by a key, i.e. if the key is 1, the text "hello world" will become "ifmmp xpsme". There are two errors in the program. Identify the errors and suggest a correction for each. After the problems are solved, what would be printed on line 10?

Le but de l'appel à la fonction `cypher` à la ligne 9 est d'encrypter le contenu de `str` en utilisant le code de César. Le code de César décale un caractère d'une certaine distance (*clé*). Ainsi, si la clé est 1, le texte « hello world » devient « ifmmp xpsme ». Il y a deux erreurs dans le programme. Identifiez les erreurs et suggérez une correction pour chacune. Après avoir corrigé les problèmes, qu'est-ce qui serait affiché par la ligne 10?

6. What is encapsulation? **1** 6. Qu'est-ce que l'encapsulation?

7. In your own words define: function declaration, function definition and function call. **2** 7. Dans vos propres mots, définissez : déclaration de fonction, définition de fonction et appel de fonction.

End of THEORETICAL Portion of Test

Fin de la partie THÉORIQUE du test