ECE Lyon - ING1 Informatique — Devoir Surveillé n°1
Algorithmes, Organigrammes et Langage C

Nom : ..... Prénom : .....

Partie 1 : Algorithmes et Organigrammes

Exercice 1 - Définition d'un algorithme *(2 points)*

Expliquez chacune des 4 grandes propriétés d'un algorithme : **calculabilité**, **complexité**, **correction** et **réutilisabilité**.

Exercice 2 - Conception à partir d'un cahier des charges *(4 points)*

Donnez un algorithme, ainsi que l'organigramme correspondant, permettant de répondre au cahier des charges suivant :

Créez un programme qui demande un nombre entier* positif à l'utilisateur, l'affiche, et indique si celui-ci est pair ou impair. Tant que l'utilisateur rentre un nombre négatif, on lui demande de recommencer, sans afficher les détails sur le nombre négatif. Le programme se termine dès que l'utilisateur a bien entré un nombre positif et que celui-ci a été affiché en précisant s'il est pair ou impair.

* *Pas besoin de vérifier que le nombre soit bien un entier.*

Partie 2 : Langage C

Exercice 3 - Logique *(2 points)*

Donnez pour chaque test logique suivant, la valeur booléenne résultante (vrai (`1`) ou faux (`0`)), ou l'erreur provoquée, **en prenant soin de détailler votre raisonnement**. Dans le cas où le test génèrerait une erreur, indiquez la correction à effectuer pour qu'il fonctionne.

```c
// Variables utilisées dans les tests :
int a = 5, b = 10;
int z = a > 100 ? 1 : 0; // raisonnement / valeur : ..........................

// Tests :
if ( a == 4 || z ) { }

if ( a && z ) { }

if ( b % 2 == 0 || b == 3 * a ) { }

if ( !(a > 100) ) { }

if ( 0 <= b <= 20 ) { }

if ( a = 5 && 1 ) { }
```

Nom : ..... Prénom : .....

Exercice 4 - Trace d'exécution *(2 points)*

Donnez la trace d'exécution du programme C suivant (vous pouvez écrire sur cette feuille ou sur votre copie en utilisant les numéros de lignes) :

```c
#include <stdio.h>

int main(void) {
  int nb = 0;

  for (int i = 1; i <= 3; i++) {

    nb += i;

  }

  printf("%d", nb);

  printf("%d", nb++);

  printf("%d", nb);

  printf("%d", ++nb);

  return 0;
}
```

Exercice 5 - Programme complet *(4 points)*

Écrivez un programme permettant de faire un calcul (addition, soustraction ou multiplication) en demandant de saisir les deux opérandes (nombres entiers) puis l'opérateur (sous la forme d'un caractère : '+', '-' ou '*'). Le programme affichera ensuite le résultat du calcul.

Exercice 6 - Correction d'erreurs *(2 points)*

Faites la liste des erreurs (en utilisant les numéros de lignes et/ou en les entourant directement ici), puis corrigez le programme suivant pour qu'il fonctionne et soit lisible :

```c
#include <stdio.h>

int main(void)
{ int a;
 printf("Veuillez rentrer votre mois de naissance :\n");
 scanf(a);
 while (1 < a > 12)
    printf(a, " n'est pas un mois valide.\n");
    printf("Veuillez recommencer")
    scanf("%d", a);}
printf("Vous êtes né en %d" a);}
```

Nom : ..... Prénom : .....

Exercice 7 - Multi-fichiers et fonctions *(4 points)*

Soit le programme C suivant permettant de calculer l'aire et le périmètre d'un rectangle. Déclarez les deux constantes `LONGUEUR` et `LARGEUR` manquantes en leur donnant les valeurs de votre choix, puis créez les fonctions `aireRectangle`, `perimetreRectangle` qui retournent respectivement l'aire du rectangle et son périmètre, et la procédure `afficherValeursRectangle` qui affiche l'aire et le périmètre du rectangle. Les deux fonctions devront être écrites dans la bibliothèque `mesfonctions`. La procédure, quant à elle, devra être définie juste en dessous de la fonction `main`, dans le fichier `main.c`. Veuillez à ce que le programme soit fonctionnel.

```c
// Fichier main.c

#include <stdio.h>

int main (void) {
  int aire, perimetre;

  aire = aireRectangle(LONGUEUR, LARGEUR);

  perimetre = perimetreRectangle(LONGUEUR, LARGEUR);

  afficherValeursRectangle(aire, perimetre);

  return 0;
}
```

```c
// Fichier mesfonctions.h

#ifndef ECE_DS1_MESFONCTIONS_H
#define ECE_DS1_MESFONCTIONS_H

#endif
```

```c
// Fichier mesfonctions.c

#include "mesfonctions.h"
```

[Bonus] Exercice 8 - Compilation *(1 point)*

Qu'est-ce que la compilation ? Détaillez un maximum vos connaissances.

[Bonus] Exercice 9 - Boucles tant que *(1 point)*

Expliquez la différence entre :

```c
do {
  // Bloc d'instructions
} while (/* condition */);
```

et :

```c
while (/* condition */) {
  // Bloc d'instructions
}
```