Nom : Prénom : N° étudiant :

Licence 1 - 1er semestre **Éléments d'informatique – contrôle continue** 2011–2012

Barème : 1 points par réponse juste (unique à chaque question) ; $-0,5$ points par réponse fausse. Durée : 20 minutes.

1. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#include <studlib.h>`
   - ☐ `#include <studio.h>`
   - ☐ `#include <stdio.h>`
   - ☐ `#appart <stdlib.h>`
2. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#include <studlib.h>`
   - ☐ `#appart <stdlib.h>`
   - ☐ `#include <studio.h>`
   - ☐ `#include <stdio.h>`
3. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
   - ☐ chacun son tour, après que le processus précédent a terminé
   - ☐ en parallèle, chacun dans un registre
   - ☐ tour à tour, un petit peu à chaque fois
   - ☐ tous ensemble
4. Soit un programme contenant les lignes suivantes :

```
int i = 0;
int j = 0;
for (i = 0; i < 0; i = i + 1)
{
    for (j = 0; j < 5; j = j + 1)
    {
      ...
    }
}
printf("j = %d\n", j);
...
}
```

   qu'est ce qui sera affiché ?
   - ☐ `j = 5`
   - ☐ `j = 4`
   - ☐ `j = 0`
   - ☐ `j = %d`
5. Le code suivant :

```
int i;
for (i = 4; i >= 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :
   - ☐ 4 3 2 1 0
   - ☐ 4 3 2 1
   - ☐ 0 1 2 3 4
   - ☐ 1 2 3 4
6. Le langage C est un langage
   - ☐ compilé
   - ☐ composé
   - ☐ lu, écrit, parlé
   - ☐ interprété
7. Quels calculs peut-on programmer en programmation structurée ?
   - ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée
   - ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine
   - ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine
   - ☐ certains programmes sont de vrais plats de spaghetti
8. Si cette erreur apparaît à la compilation : `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
   - ☐ un point-virgule en trop
   - ☐ un point-virgule manquant
   - ☐ une accolade en trop
   - ☐ une accolade manquante
9. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

   affichera :
   - ☐ `Majeur Mineur`
   - ☐ rien
   - ☐ `Majeur`
   - ☐ `Mineur`

10. Le code suivant :

```
int somme = 0;
int i;
for (i = 1; i < 4; i = i + 1)
{
  somme = somme + i;
}
printf("%d", somme);
```

affichera :

- ☐ 1
- ☐ 6
- ☐ 42
- ☐ 0

11. Pour l'extrait de programme suivant :

```
int i = 0;
int j = 0;
for (i = 0; i < 2; i = i + 1)
{
    for (j = 0; j < 3; j = j + 1)
    {
        printf("%d ", j);
    }
}
```

qu'est ce qui sera affiché ?

- ☐ `0 0 1 1 2 2 3`
- ☐ `0 1 2 0 1 2`
- ☐ `0 1 2 0 1 2 3`
- ☐ `0 1 2 3 0 1 2`

12. Sous unix (ou linux), la commande `ls` permet de :

- ☐ voir des clips musicaux
- ☐ compiler un programme
- ☐ afficher le contenu d'un fichier texte
- ☐ afficher la liste de fichiers contenus dans un répertoire

13. Soit un programme contenant les lignes suivantes :

```
int i = 0;
int j = 0;
for (i = 0; i < 3; i = i + 1)
{
    for (j = 0; j < 5; j = j + 1)
    {
          ...
    }
}
printf("j = %d\n", j);
```

qu'est ce qui sera affiché par ce printf ?

- ☐ `j = 4`
- ☐ `j = 5`
- ☐ `j = %d`
- ☐ `j = 0`

14. Une *segmentation fault* est une erreur qui survient lorsque :

- ☐ la division du programme en zones homogènes échoue
- ☐ le programme tente d'accèder à une partie de la mémoire qui ne lui est pas réservée
- ☐ le programme source a été enregistré sur le disque dur en plusieurs morceaux et l'un d'entre eux ne peut pas être chargé par le compilateur
- ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal

15. Un bit est :

- ☐ un battement d'horloge processeur
- ☐ la longueur d'un mot mémoire
- ☐ l'instruction qui met fin à un programme
- ☐ un chiffre binaire (0 ou 1)

16. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction

- ☐ `int toto[taille=5];`
- ☐ `char tableau[5];`
- ☐ `int toto[5];`
- ☐ `int tab[] = 5;`
- ☐ `int[] new tableau(5);`

17. Sur unix (ou linux), la commande `mkdir` permet de :

- ☐ créer un répertoire
- ☐ créer un fichier texte
- ☐ changer de répertoire courant
- ☐ ouvrir un fichier texte

18. Soient deux variables entières `x` et `y` initialisées à 4 et 5 respectivement. L'affichage `x=4 et y=5` est obtenu avec la commande :

- ☐ `printf("x=%d et y=%d\n,x,y");`
- ☐ `printf("x=%x et y=%y\n");`
- ☐ `printf("x=%d et y=%d\n",x,y);`
- ☐ `printf("x=%d et y=%d\n",x y);`

19. Le bus système sert à :

- ☐ Transférer des données et intructions entre processeur et mémoire
- ☐ Écrire des données sur le dique dur
- ☐ transporter les processus du tourniquet au processeur
- ☐ Arriver à l'heure en cours

20. Le code suivant :

```
int i;
for (i = 1; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 4 3 2 1
- ☐ 0 1 2 3 4
- ☐ 4 3 2 1 0
- ☐ 1 2 3 4

21. Pour l'extrait de programme suivant :

```
int produit = 0;
int serie[4] = {2, 2, 2, 2};
for (i = 0; i < 4; i = i + 1)
{
  produit = produit * serie[i];
}
printf("produit = %d", produit);
```

La valeur affichée est :

- ☐ 0
- ☐ 16
- ☐ 8
- ☐ 4

22. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 8 6 4 2 0
- ☐ 0 2 4 6 8
- ☐ 8 6 4 2
- ☐ 8 2

23. Le code suivant :

```
int i;
for (i = 0; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 0 1 2 3
- ☐ 4 3 2 1 0
- ☐ 4 3 2 1
- ☐ 0 1 2 3 4

24. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :

- ☐ retourner un bloc
- ☐ mettre les blocs en séquence les uns à la suite des autres
- ☐ répéter un bloc tant qu'une condition est vérifée
- ☐ sélectionner entre deux blocs à l'aide d'une condition

25. Soit un programme contenant les lignes suivantes :

```
int i = 0;
int j = 0;
for (i = 0; i < 2; i = i + 1)
{
    for (j = 0; j < 3; j = j + 1)
    {
        printf("%d ", i);
    }
}
```

qu'est ce qui sera affiché ?

- ☐ `0 0 0 1 1 1`
- ☐ `1 2 1 2 3`
- ☐ `0 1 0 1 0 1 0 1`
- ☐ `0 1 2 0 1 2`

26. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction

- ☐ `loop i;`
- ☐ `int loop n;`
- ☐ `int %d;`
- ☐ `int k;`

27. Un programme en langage C doit comporter une et une seule définition de la fonction :

- ☐ init
- ☐ begin
- ☐ main
- ☐ include

28. Les lignes

```
int i;
int x=0;
for(i=0,i<5,i=i+1)
{
  x=x+1;
}
```

- ☐ comportent une erreur qui ne sera pas détectée
- ☐ comportent une erreur qui sera détectée au cours de l'analyse syntaxique
- ☐ comportent une erreur qui sera détectée au cours de l'édition de lien
- ☐ ne comportent aucune erreur

29. Si cette erreur apparaît à la compilation :
`Undefined symbols :"_prinft"` ou
`référence indéfinie vers` « `prinft` » que doit-on chercher dans le programme ?

- ☐ une directive préprocesseur `#include` manquante
- ☐ un caractère interdit en C
- ☐ une faute de frappe dans un appel de fonction
- ☐ une variable non déclarée

30. Un registre du processeur est :

- ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs
- ☐ une gamme de fréquence de fonctionnement du processeur
- ☐ une unité de calcul spécialisée de l'ordinateur
- ☐ un composant qui contient la liste des fichiers du système

31. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Mineur`
- ☐ rien
- ☐ `Majeur`
- ☐ `Majeur Mineur`

32. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :

- ☐ `mkdir TP4`
- ☐ `yppasswd`
- ☐ `new TP4`
- ☐ `kwrite TP4`

33. Après exécution jusqu'à la ligne 15 du programme C :

```
10  int main() {
11      int x = 5;
12      int y = 3;
13
14      x = y;
15
16      ...
17  }
```

- ☐ la variable y vaut 5
- ☐ la variable x vaut 5 et la variable y vaut 3
- ☐ la variable x vaut 3
- ☐ le programme affiche "Faux"

34. Pour compiler un programme `prog.c`, on utilise la ligne de commande :

- ☐ `gcc -Wall prog.c -o prog.exe`
- ☐ `gcc -Wall prog.exe -o prog.c`
- ☐ `gcc prog.c -o -Wall prog.exe`
- ☐ `gcc prog.exe -Wall -o prog.c`

35. Après exécution du programme :

```
1  lecture 8 r0
2  valeur 3 r1
3  mult r1 r0
4  valeur 1 r2
5  add r2 r0
6  ecriture r0 8
7  stop
8  5
```

- ☐ le terminal affiche 8
- ☐ la case mémoire 8 contiendra 16
- ☐ la case mémoire 8 contiendra 0
- ☐ le bus explose

36. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :

- ☐ analyse syntaxique
- ☐ analyse sémantique
- ☐ analyse lexicale
- ☐ analyse harmonique

37. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.

- ☐ `#define taille = 3`
- ☐ `#define N 3`
- ☐ `#define taille = N`
- ☐ `#define N =  3`

38. Pour l'extrait de programme suivant :

```
int somme = 0;
for (i = 0; i < 5; i = i + 1)
{
  somme = somme + i;
  i = i + 1; /* attention ! */
}
printf("somme = %d",somme);
```

La valeur de somme affichée est :

- ☐ 0
- ☐ 10
- ☐ 15
- ☐ 6

39. Pour l'extrait de programme suivant :

```
int somme = 0;
int serie[4] = {2, 4, 10, 4};
for (i = 0; i < 4; i = i + 1)
{
  somme = somme + serie[i];
}
printf("somme = %d",somme);
```

La valeur de somme affichée est :

- ☐ 16
- ☐ 3
- ☐ 6
- ☐ 20

40. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
printf("Majeur\n");
```

affichera :

- ☐ `Majeur`
- ☐ rien
- ☐ `Majeur Mineur`
- ☐ `Mineur`

41. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :

- ☐ certaines données de la mémoire de travail
- ☐ en temps d'accès
- ☐ des processus
- ☐ les fichiers du disque

42. L'ordonnancement par tourniquet permet :

- ☐ d'entretenir l'illusion que les processus tournent en parallèle
- ☐ d'afficher des ronds colorés à l'écran
- ☐ de ne pas perdre de temps avec la commutation de contexte
- ☐ de doubler la mémoire disponible

43. Après exécution jusqu'à la ligne 15 du programme C :

```
10  ...
11  int main() {
12      int x = 5;
13
14      x = 3 * x + 1;
15
16      ...
17  }
```

- ☐ le programme affiche `****`
- ☐ la variable x vaut $-\frac{1}{2}$
- ☐ le programme affiche x
- ☐ la variable x vaut 16

44. Après exécution jusqu'à la ligne 14 du programme C :

```
10  int main() {
11      int x = 5;
12
13      printf(" x  = %d\n", 2);
14
15      ...
16  }
```

- ☐ le terminal affiche "Faux"
- ☐ le terminal affiche `x = 2`
- ☐ le terminal affiche `x = 5`
- ☐ le terminal affiche `5`

45. Pour l'extrait de programme suivant :

```
int i;
int j;
for(i=4;i>0;i=i-1)
{
  for(j=i;j<6;j=j+1)
  {
    printf("*");
  }
  printf(" ");
}
```

qu'est ce qui sera affiché ?

- ☐ `***** **** *** ***`
- ☐ `** *** **** *****`
- ☐ `** ** ** ** ** **`
- ☐ `**** **** **** ****`

46. Le code suivant :

```
int i;
for (i = 0; i < 7; i = i + 2)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 0 1 2 3 4 5 6
- ☐ 0 1 2 3 4 5 6 7
- ☐ 0 2 4 6
- ☐ 0 2 4 6 8

47. Quelle étape de la compilation vient d'échouer lorsqu'on a un message comme celui-ci :
`Undefined symbols :"_prinft"` ou
`référence indéfinie vers « prinft »`

- ☐ l'édition de liens
- ☐ l'analyse des entrées clavier
- ☐ l'analyse sémantique
- ☐ l'analyse harmonique

48. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ rien
- ☐ `Mineur`
- ☐ `Majeur Mineur`
- ☐ `Majeur`

49. Quel est l'opérateur de différence en C :

- ☐ `!=`
- ☐ $\neq$
- ☐ `!`
- ☐ `<>`

50. Sous unix (ou linux), la commande `cd` permet de :

- ☐ récupérer un programme arrêté avec la commande `ab`
- ☐ jouer de la musique
- ☐ détruire un fichier
- ☐ ouvir un bureau partagé (common desktop)
- ☐ changer de répertoire courant

51. Un fichier source est :
    - ☐ un document de référence du système
    - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
    - ☐ un document qui doit être protégé
    - ☐ un fichier texte qui sera traduit en instructions processeur
    - ☐ un document illisible pour les humains

52. Le code suivant :

```
int i;
for (i = 4; i > 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

    affichera :
    - ☐ 4 3 2 1 0
    - ☐ 4 3 2 1
    - ☐ 0 1 2 3 4
    - ☐ 0 1 2 3

53. Dans la commande gcc, l'option `-Wall` signifie :
    - ☐ qu'il faut indenter le fichier source
    - ☐ qu'on veut changer alétoirement de fond d'écran
    - ☐ qu'il faut lancer un déboggueur
    - ☐ que l'on veut voir tous les avertissements

54. Pour l'extrait de programme suivant :

```
int produit = 1;
int serie[4] = {2, 2, 2, 2};
for (i = 0; i < 4; i = i + 1)
{
  produit = produit * serie[i];
}
printf("produit = %d", produit);
```

    La valeur affichée est :
    - ☐ 16
    - ☐ 8
    - ☐ 0
    - ☐ 4

55. Pour afficher à l'aide de `printf("%d\n",tab[i]);` le contenu d'un tableau de 5 entiers initialisé au préalable, on utilise plutôt :
    - ☐ `for(i=1;i<=5;i=i+1)`
    - ☐ `for(i=0;i<=5;i=i+1)`
    - ☐ `for(i=0;i<5;i=i+1)`
    - ☐ `for(i=1;i<5;i=i+1)`

56. Pour l'extrait de programme suivant :

```
int i = 0;
int j = 0;
for (i = 0; i < 3; i = i + 1)
{
    for (j = 0; j < 2; j = j + 1)
    {
        printf("%d ", i);
    }
}
printf("\n");
```

    qu'est ce qui sera affiché ?
    - ☐ `0 1 0 1 0 1`
    - ☐ `0 0 1 1 2 2`
    - ☐ `0 1 2 0 1 2`
    - ☐ `1 2 3 1 2`

57. Après exécution jusqu'à la ligne 15 du programme C :

```
10  int main() {
11      int x = 5;
12      int y;
13
14      y = x;
15
16      ...
17  }
```

    - ☐ le programme affiche "Faux"
    - ☐ la variable x vaut 0
    - ☐ la variable y vaut 5
    - ☐ la variable x vaut 5 et la variable y vaut 0

N'oubliez pas de mettre vos prénom, nom, numéro d'étudiant au recto.