Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Quels calculs peut-on programmer en programmation structurée ?
   - ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine
   - ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée
   - ☐ certains programmes sont de vrais plats de spaghetti
   - ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine
2. 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 tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
   - ☐ 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
3. Le code suivant :

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

   affichera :
   - ☐ 0 2 4 6
   - ☐ 0 1 2 3 4 5 6 7
   - ☐ 0 2 4 6 8
   - ☐ 0 1 2 3 4 5 6
4. Le code suivant :

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

   affichera :
   - ☐ `Majeur`
   - ☐ `Mineur`
   - ☐ `Mineur`
`Majeur`
   - ☐ rien
5. 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 :
   - ☐ 4
   - ☐ 16
   - ☐ 8
   - ☐ 0
6. 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 1 2 0 1 2`
   - ☐ `0 0 1 1 2 2 3`
   - ☐ `0 1 2 0 1 2 3`
   - ☐ `0 1 2 3 0 1 2`
7. Le bus système sert à :
   - ☐ Transférer des données et intructions entre processeur et mémoire
   - ☐ transporter les processus du tourniquet au processeur
   - ☐ Arriver à l'heure en cours
   - ☐ Écrire des données sur le dique dur
8. 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 :
   - ☐ 0
   - ☐ 8
   - ☐ 16
   - ☐ 4
9. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse harmonique
   - ☐ analyse lexicale
   - ☐ analyse syntaxique
   - ☐ analyse sémantique
10. Sous unix (ou linux), la commande `cd` permet de :
    - ☐ détruire un fichier
    - ☐ jouer de la musique
    - ☐ récupérer un programme arrêté avec la commande `ab`
    - ☐ ouvir un bureau partagé (common desktop)
    - ☐ changer de répertoire courant

11. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
    - ☐ `loop i;`
    - ☐ `int %d;`
    - ☐ `int k;`
    - ☐ `int loop n;`
12. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :
    - ☐ en temps d'accès
    - ☐ les fichiers du disque
    - ☐ des processus
    - ☐ certaines données de la mémoire de travail
13. Quel est l'opérateur de différence en C :
    - ☐ $\neq$
    - ☐ `<>`
    - ☐ `!`
    - ☐ `!=`
14. Le code suivant :

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

    affichera :
    - ☐ 0 1 2 3 4
    - ☐ 1 2 3 4
    - ☐ 4 3 2 1
    - ☐ 4 3 2 1 0
15. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
    - ☐ `int[] new tableau(5);`
    - ☐ `int tab[] = 5;`
    - ☐ `int toto[5];`
    - ☐ `char tableau[5];`
    - ☐ `int toto[taille=5];`
16. Le langage C est un langage
    - ☐ interprété
    - ☐ composé
    - ☐ compilé
    - ☐ lu, écrit, parlé
17. 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'analyse sémantique
    - ☐ l'analyse harmonique
    - ☐ l'édition de liens
    - ☐ l'analyse des entrées clavier
18. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
    - ☐ mettre les blocs en séquence les uns à la suite des autres
    - ☐ sélectionner entre deux blocs à l'aide d'une condition
    - ☐ répéter un bloc tant qu'une condition est vérifée
    - ☐ retourner un bloc
19. Le code suivant :

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

    affichera :
    - ☐ `Mineur`
    - ☐ `Majeur`
    - ☐ rien
    - ☐ `Mineur`
      `Majeur`
20. 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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. 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 1 0 1 0 1 0 1`
☐ `0 0 0 1 1 1`
☐ `1 2 1 2 3`
☐ `0 1 2 0 1 2`

2. 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 :

☐ 4
☐ 0
☐ 16
☐ 8

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

4. 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 `x = 2`
☐ le terminal affiche "Faux"
☐ le terminal affiche `5`
☐ le terminal affiche `x = 5`

5. Le code suivant :

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

affichera :

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

6. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

☐ `#include <studio.h>`
☐ `#appart <stdlib.h>`
☐ `#include <studlib.h>`
☐ `#include <stdio.h>`

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

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

8. Le code suivant :

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

affichera :

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

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

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

10. Le bus système sert à :

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

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

12. 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 7
- ☐ 0 2 4 6 8
- ☐ 0 1 2 3 4 5 6
- ☐ 0 2 4 6

13. 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
- ☐ 0
- ☐ 4
- ☐ 8

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

15. Quels calculs peut-on programmer en programmation structurée ?

- ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine
- ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée
- ☐ certains programmes sont de vrais plats de spaghetti
- ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine

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

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

17. L'ordonnancement par tourniquet permet :

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

18. 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  }
```

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

19. 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
- ☐ 1 2 3 1 2
- ☐ 0 1 2 0 1 2
- ☐ 0 0 1 1 2 2

20. Le code suivant :

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

affichera :

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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. L'ordonnancement par tourniquet permet :
   - ☐ d'entretenir l'illusion que les processus tournent en parallèle
   - ☐ de doubler la mémoire disponible
   - ☐ de ne pas perdre de temps avec la commutation de contexte
   - ☐ d'afficher des ronds colorés à l'écran
2. Le bus système sert à :
   - ☐ transporter les processus du tourniquet au processeur
   - ☐ Arriver à l'heure en cours
   - ☐ Transférer des données et intructions entre processeur et mémoire
   - ☐ Écrire des données sur le dique dur
3. 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 1 2 0 1 2`
   - ☐ `1 2 1 2 3`
   - ☐ `0 1 0 1 0 1 0 1`
   - ☐ `0 0 0 1 1 1`
4. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `char tableau[5];`
   - ☐ `int toto[5];`
   - ☐ `int toto[taille=5];`
   - ☐ `int[] new tableau(5);`
   - ☐ `int tab[] = 5;`
5. Le code suivant :

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

   affichera :
   - ☐ `Mineur`
   - ☐ rien
   - ☐ `Majeur`
   - ☐ `Mineur`
     `Majeur`
6. Quel est l'opérateur de différence en C :
   - ☐ `!`
   - ☐ `<>`
   - ☐ `!=`
   - ☐ $\neq$
7. Si cette erreur apparaît à la compilation :
   `Undefined symbols :"_prinft"` ou
   `référence indéfinie vers` « `prinft` » que doit-on chercher dans le programme ?
   - ☐ une faute de frappe dans un appel de fonction
   - ☐ un caractère interdit en C
   - ☐ une variable non déclarée
   - ☐ une directive préprocesseur `#include` manquante
8. Le code suivant :

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

   affichera :
   - ☐ `4 3 2 1 0`
   - ☐ `0 1 2 3 4`
   - ☐ `4 3 2 1`
   - ☐ `0 1 2 3`
9. Le code suivant :

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

   affichera :
   - ☐ `1 2 3 4`
   - ☐ `4 3 2 1`
   - ☐ `4 3 2 1 0`
   - ☐ `0 1 2 3 4`
10. 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
    - ☐ le bus explose
    - ☐ la case mémoire 8 contiendra 0
11. 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 :

- ☐ 0
- ☐ 16
- ☐ 8
- ☐ 4

12. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
    - ☐ sélectionner entre deux blocs à l'aide d'une condition
    - ☐ retourner un bloc
    - ☐ répéter un bloc tant qu'une condition est vérifée
    - ☐ mettre les blocs en séquence les uns à la suite des autres
13. Un fichier source est :
    - ☐ un document qui doit être protégé
    - ☐ un document de référence du système
    - ☐ un document illisible pour les humains
    - ☐ un fichier texte qui sera traduit en instructions processeur
    - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
14. 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 = 0`
    - ☐ `j = 4`
    - ☐ `j = 5`
    - ☐ `j = %d`
15. 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);`
16. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ ouvrir un fichier texte
    - ☐ créer un fichier texte
    - ☐ changer de répertoire courant
    - ☐ créer un répertoire
17. Sous unix (ou linux), la commande `ls` permet de :
    - ☐ afficher le contenu d'un fichier texte
    - ☐ afficher la liste de fichiers contenus dans un répertoire
    - ☐ compiler un programme
    - ☐ voir des clips musicaux
18. 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 x vaut 3
    - ☐ la variable x vaut 5 et la variable y vaut 3
    - ☐ le programme affiche "Faux"
    - ☐ la variable y vaut 5
19. 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 sémantique
    - ☐ l'analyse des entrées clavier
    - ☐ l'analyse harmonique
20. Un registre du processeur est :
    - ☐ un composant qui contient la liste des fichiers du système
    - ☐ 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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
   - ☐ `gcc -Wall prog.exe -o prog.c`
   - ☐ `gcc prog.exe -Wall -o prog.c`
   - ☐ `gcc prog.c -o -Wall prog.exe`
   - ☐ `gcc -Wall prog.c -o prog.exe`
2. Une *segmentation fault* est une erreur qui survient lorsque :
   - ☐ 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
   - ☐ la division du programme en zones homogènes échoue
   - ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
3. 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
   - ☐ sélectionner entre deux blocs à l'aide d'une condition
   - ☐ répéter un bloc tant qu'une condition est vérifée
4. Le code suivant :

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

   affichera :
   - ☐ 6
   - ☐ 42
   - ☐ 0
   - ☐ 1
5. 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  }
```

   - ☐ la variable x vaut 16
   - ☐ le programme affiche `****`
   - ☐ la variable x vaut $-\frac{1}{2}$
   - ☐ le programme affiche x
6. 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=0;i<=5;i=i+1)`
   - ☐ `for(i=1;i<5;i=i+1)`
   - ☐ `for(i=0;i<5;i=i+1)`
   - ☐ `for(i=1;i<=5;i=i+1)`
7. Le code suivant :

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

   affichera :
   - ☐ `Majeur`
   - ☐ `Mineur`
   - ☐ rien
   - ☐ `Mineur`
     `Majeur`
8. Dans la commande gcc, l'option `-Wall` signifie :
   - ☐ que l'on veut voir tous les avertissements
   - ☐ qu'on veut changer alétoirement de fond d'écran
   - ☐ qu'il faut lancer un déboggueur
   - ☐ qu'il faut indenter le fichier source
9. 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
   - ☐ un composant qui contient la liste des fichiers du système
   - ☐ une unité de calcul spécialisée de l'ordinateur
10. 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=%x et y=%y\n");`
    - ☐ `printf("x=%d et y=%d\n",x,y);`
    - ☐ `printf("x=%d et y=%d\n,x,y");`
    - ☐ `printf("x=%d et y=%d\n",x y);`
11. Un bit est :
    - ☐ l'instruction qui met fin à un programme
    - ☐ un chiffre binaire (0 ou 1)
    - ☐ un battement d'horloge processeur
    - ☐ la longueur d'un mot mémoire
12. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
    - ☐ chacun son tour, après que le processus précédent a terminé
    - ☐ tous ensemble
    - ☐ tour à tour, un petit peu à chaque fois
    - ☐ en parallèle, chacun dans un registre

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

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

14. 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

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

16. L'ordonnancement par tourniquet permet :

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

17. 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`

18. 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
- ☐ le programme affiche "Faux"
- ☐ la variable x vaut 3

19. 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 :

- ☐ 8
- ☐ 4
- ☐ 16
- ☐ 0

20. Le code suivant :

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

affichera :

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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
   - ☐ mettre les blocs en séquence les uns à la suite des autres
   - ☐ sélectionner entre deux blocs à l'aide d'une condition
   - ☐ répéter un bloc tant qu'une condition est vérifée
   - ☐ retourner un bloc
2. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ init
   - ☐ main
   - ☐ include
   - ☐ begin
3. Le code suivant :

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

   affichera :
   - ☐ 0 1 2 3 4
   - ☐ 0 1 2 3
   - ☐ 4 3 2 1
   - ☐ 4 3 2 1 0
4. 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  }
```

   - ☐ la variable x vaut 5 et la variable y vaut 0
   - ☐ la variable x vaut 0
   - ☐ le programme affiche "Faux"
   - ☐ la variable y vaut 5
5. 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
6. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
   - ☐ `gcc -Wall prog.exe -o prog.c`
   - ☐ `gcc prog.exe -Wall -o prog.c`
   - ☐ `gcc prog.c -o -Wall prog.exe`
   - ☐ `gcc -Wall prog.c -o prog.exe`
7. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
   - ☐ `#define N 3`
   - ☐ `#define taille = N`
   - ☐ `#define N =  3`
   - ☐ `#define taille = 3`
8. 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 `x = 2`
   - ☐ le terminal affiche `5`
   - ☐ le terminal affiche "Faux"
   - ☐ le terminal affiche `x = 5`
9. Le code suivant :

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

   affichera :
   - ☐ `Majeur`
   - ☐ rien
   - ☐ `Mineur`
   - ☐ `Mineur`
     `Majeur`
10. 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 x
    - ☐ le programme affiche `****`
    - ☐ la variable x vaut $-\frac{1}{2}$
    - ☐ la variable x vaut 16
11. 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

12. 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 1 2 0 1 2`

☐ `1 2 1 2 3`

☐ `0 0 0 1 1 1`

☐ `0 1 0 1 0 1 0 1`

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

☐ `int %d;`

☐ `int loop n;`

☐ `loop i;`

☐ `int k;`

14. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

☐ `#include <stdio.h>`

☐ `#appart <stdlib.h>`

☐ `#include <studio.h>`

☐ `#include <studlib.h>`

15. 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

16. Si cette erreur apparaît à la compilation :
`error: expected ';' before '}' token` que doit-on chercher dans le programme ?

☐ un point-virgule manquant

☐ un point-virgule en trop

☐ une accolade en trop

☐ une accolade manquante

17. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

☐ `#appart <stdlib.h>`

☐ `#include <stdio.h>`

☐ `#include <studio.h>`

☐ `#include <studlib.h>`

18. 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 = %d`

☐ `j = 0`

☐ `j = 4`

☐ `j = 5`

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

☐ $\neq$

☐ `!=`

☐ `!`

☐ `<>`

20. 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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Quel est l'opérateur de différence en C :
   - ☐ $\neq$
   - ☐ `!`
   - ☐ `<>`
   - ☐ `!=`

2. Le code suivant :

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

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

3. Une *segmentation fault* est une erreur qui survient lorsque :
   - ☐ 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
   - ☐ le programme tente d'accèder à une partie de la mémoire qui ne lui est pas réservée
   - ☐ la division du programme en zones homogènes échoue

4. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
   - ☐ `gcc -Wall prog.c -o prog.exe`
   - ☐ `gcc prog.c -o -Wall prog.exe`
   - ☐ `gcc prog.exe -Wall -o prog.c`
   - ☐ `gcc -Wall prog.exe -o prog.c`

5. 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 0
   - ☐ la case mémoire 8 contiendra 16
   - ☐ le bus explose

6. Le code suivant :

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

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

7. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
   - ☐ `kwrite TP4`
   - ☐ `yppasswd`
   - ☐ `mkdir TP4`
   - ☐ `new TP4`

8. 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 :
   - ☐ 16
   - ☐ 8
   - ☐ 4
   - ☐ 0

9. 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 :
   - ☐ 20
   - ☐ 16
   - ☐ 6
   - ☐ 3

10. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
    - ☐ `char tableau[5];`
    - ☐ `int tab[] = 5;`
    - ☐ `int toto[5];`
    - ☐ `int toto[taille=5];`
    - ☐ `int[] new tableau(5);`

11. 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

12. 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=%d et y=%d\n",x,y);`

    ☐ `printf("x=%x et y=%y\n");`

    ☐ `printf("x=%d et y=%d\n,x,y");`

13. Dans la commande gcc, l'option `-Wall` signifie :

    ☐ qu'on veut changer alétoirement de fond d'écran

    ☐ que l'on veut voir tous les avertissements

    ☐ qu'il faut lancer un déboggueur

    ☐ qu'il faut indenter le fichier source

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

    ☐ créer un répertoire

    ☐ ouvrir un fichier texte

    ☐ changer de répertoire courant

    ☐ créer un fichier texte

15. 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=0;i<=5;i=i+1)`

    ☐ `for(i=1;i<5;i=i+1)`

    ☐ `for(i=0;i<5;i=i+1)`

    ☐ `for(i=1;i<=5;i=i+1)`

16. Le code suivant :

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

    affichera :

    ☐ 0 1 2 3 4

    ☐ 4 3 2 1

    ☐ 4 3 2 1 0

    ☐ 0 1 2 3

17. 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

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

    ☐ analyse syntaxique

    ☐ analyse sémantique

    ☐ analyse harmonique

    ☐ analyse lexicale

19. 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

    ☐ les fichiers du disque

    ☐ en temps d'accès

    ☐ des processus

20. Le code suivant :

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

    affichera :

    ☐ 42

    ☐ 0

    ☐ 1

    ☐ 6

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Le code suivant :

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

affichera :

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

2. 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 :

- ☐ 4
- ☐ 16
- ☐ 0
- ☐ 8

3. Une *segmentation fault* est une erreur qui survient lorsque :
   - ☐ le programme tente d'accèder à une partie de la mémoire qui ne lui est pas réservée
   - ☐ la division du programme en zones homogènes échoue
   - ☐ 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

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

5. Le code suivant :

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

affichera :

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

6. Quels calculs peut-on programmer en programmation structurée ?
   - ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine
   - ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine
   - ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée
   - ☐ certains programmes sont de vrais plats de spaghetti

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

8. L'ordonnancement par tourniquet permet :
   - ☐ de ne pas perdre de temps avec la commutation de contexte
   - ☐ d'entretenir l'illusion que les processus tournent en parallèle
   - ☐ d'afficher des ronds colorés à l'écran
   - ☐ de doubler la mémoire disponible

9. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `char tableau[5];`
   - ☐ `int toto[5];`
   - ☐ `int[] new tableau(5);`
   - ☐ `int toto[taille=5];`
   - ☐ `int tab[] = 5;`

10. Quel est l'opérateur de différence en C :
    - ☐ `!`
    - ☐ $\neq$
    - ☐ `!=`
    - ☐ `<>`

11. Sous unix (ou linux), la commande `ls` permet de :
    - ☐ afficher la liste de fichiers contenus dans un répertoire
    - ☐ compiler un programme
    - ☐ voir des clips musicaux
    - ☐ afficher le contenu d'un fichier texte

12. Le code suivant :

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

affichera :

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

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

14. 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
- ☐ 1 2 3 1 2
- ☐ 0 1 2 0 1 2

15. Le code suivant :

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

affichera :

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

16. 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 :

- ☐ 4
- ☐ 0
- ☐ 16
- ☐ 8

17. 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 1 0 1 0 1 0 1
- ☐ 0 0 0 1 1 1
- ☐ 1 2 1 2 3
- ☐ 0 1 2 0 1 2

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

19. 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=1;i<5;i=i+1)`
- ☐ `for(i=0;i<=5;i=i+1)`

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

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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. 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  }
```

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

2. Le code suivant :

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

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

3. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
   - ☐ tous ensemble
   - ☐ tour à tour, un petit peu à chaque fois
   - ☐ en parallèle, chacun dans un registre
   - ☐ chacun son tour, après que le processus précédent a terminé

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

5. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
   - ☐ `#define N =  3`
   - ☐ `#define taille = 3`
   - ☐ `#define N 3`
   - ☐ `#define taille = N`

6. 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 1 2 0 1 2 3`
   - ☐ `0 1 2 3 0 1 2`
   - ☐ `0 1 2 0 1 2`
   - ☐ `0 0 1 1 2 2 3`

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

8. 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=0;i<=5;i=i+1)`
   - ☐ `for(i=1;i<=5;i=i+1)`
   - ☐ `for(i=1;i<5;i=i+1)`
   - ☐ `for(i=0;i<5;i=i+1)`

9. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `char tableau[5];`
   - ☐ `int toto[taille=5];`
   - ☐ `int tab[] = 5;`
   - ☐ `int[] new tableau(5);`
   - ☐ `int toto[5];`

10. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
    - ☐ sélectionner entre deux blocs à l'aide d'une condition
    - ☐ répéter un bloc tant qu'une condition est vérifée
    - ☐ retourner un bloc
    - ☐ mettre les blocs en séquence les uns à la suite des autres

11. 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>`

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

13. Quels calculs peut-on programmer en programmation structurée ?

- ☐ certains programmes sont de vrais plats de spaghetti
- ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée
- ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine
- ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine

14. Le code suivant :

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

affichera :

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

15. 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 ?

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

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

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

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

18. 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 :

- ☐ 20
- ☐ 3
- ☐ 16
- ☐ 6

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

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

20. Le code suivant :

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

affichera :

- ☐ 0
- ☐ 6
- ☐ 42
- ☐ 1

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Le code suivant :

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

affichera :

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

2. Une *segmentation fault* est une erreur qui survient lorsque :
   - ☐ 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
   - ☐ le programme tente d'accèder à une partie de la mémoire qui ne lui est pas réservée
   - ☐ la division du programme en zones homogènes échoue

3. Le code suivant :

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

affichera :

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

4. Quel est l'opérateur de différence en C :
   - ☐ `!`
   - ☐ `<>`
   - ☐ `!=`
   - ☐ $\neq$

5. Le code suivant :

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

affichera :

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

6. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ main
   - ☐ include
   - ☐ init
   - ☐ begin

7. Le code suivant :

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

affichera :

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

8. 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  }
```

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

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

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

11. 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
```

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

12. 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 ?

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

13. Un bit est :

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

14. L'ordonnancement par tourniquet permet :

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

15. 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
- ☐ 4
- ☐ 0

16. 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  }
```

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

17. Le code suivant :

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

affichera :

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

18. 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

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

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

20. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#include <stdio.h>`
- ☐ `#include <studlib.h>`
- ☐ `#include <studio.h>`
- ☐ `#appart <stdlib.h>`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Le code suivant :

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

affichera :

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

2. Le code suivant :

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

affichera :

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

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

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

4. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :

- ☐ chacun son tour, après que le processus précédent a terminé
- ☐ tous ensemble
- ☐ en parallèle, chacun dans un registre
- ☐ tour à tour, un petit peu à chaque fois

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

6. Le code suivant :

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

affichera :

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

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

- ☐ `!=`
- ☐ `<>`
- ☐ `!`
- ☐ $\neq$

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

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

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

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

10. 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 1 2 0 1 2`
- ☐ `0 1 2 3 0 1 2`
- ☐ `0 0 1 1 2 2 3`
- ☐ `0 1 2 0 1 2 3`

11. 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
- ☐ 4
- ☐ 0

12. Le code suivant :

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

affichera :

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

13. 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 :

- ☐ 4
- ☐ 8
- ☐ 16
- ☐ 0

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

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

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

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

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

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

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

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

18. Un bit est :

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

19. Le code suivant :

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

affichera :

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

20. 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 = %d`
- ☐ `j = 5`
- ☐ `j = 0`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Le code suivant :

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

affichera :

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

2. 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 = 5`
- ☐ `j = 4`
- ☐ `j = 0`
- ☐ `j = %d`

3. 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
   - ☐ certains programmes sont de vrais plats de spaghetti
   - ☐ 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
4. Sous unix (ou linux), la commande `cd` permet de :
   - ☐ récupérer un programme arrêté avec la commande `ab`
   - ☐ ouvir un bureau partagé (common desktop)
   - ☐ changer de répertoire courant
   - ☐ détruire un fichier
   - ☐ jouer de la musique
5. 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 :
   - ☐ 15
   - ☐ 6
   - ☐ 10
   - ☐ 0
6. 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'analyse des entrées clavier
   - ☐ l'analyse sémantique
   - ☐ l'édition de liens
   - ☐ l'analyse harmonique
7. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
   - ☐ `gcc prog.c -o -Wall prog.exe`
   - ☐ `gcc -Wall prog.exe -o prog.c`
   - ☐ `gcc prog.exe -Wall -o prog.c`
   - ☐ `gcc -Wall prog.c -o prog.exe`
8. 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
9. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse harmonique
   - ☐ analyse lexicale
   - ☐ analyse sémantique
   - ☐ analyse syntaxique
10. 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"
11. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#appart <stdlib.h>`
   - ☐ `#include <studio.h>`
   - ☐ `#include <stdio.h>`
   - ☐ `#include <studlib.h>`

12. 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

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

14. Si cette erreur apparaît à la compilation : `error: expected ';' before '}' token` que doit-on chercher dans le programme ?

- ☐ une accolade en trop
- ☐ un point-virgule manquant
- ☐ un point-virgule en trop
- ☐ une accolade manquante

15. Le code suivant :

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

affichera :

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

16. Un fichier source est :

- ☐ un document de référence du système
- ☐ un document illisible pour les humains
- ☐ 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

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

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

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

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

19. 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 :

- ☐ 4
- ☐ 0
- ☐ 16
- ☐ 8

20. Un bit est :

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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Le code suivant :

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

affichera :

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

2. 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.exe -Wall -o prog.c`
   - ☐ `gcc prog.c -o -Wall prog.exe`
3. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `int toto[5];`
   - ☐ `int toto[taille=5];`
   - ☐ `char tableau[5];`
   - ☐ `int tab[] = 5;`
   - ☐ `int[] new tableau(5);`
4. Un fichier source est :
   - ☐ un document qui doit être protégé
   - ☐ un document illisible pour les humains
   - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
   - ☐ un document de référence du système
   - ☐ un fichier texte qui sera traduit en instructions processeur
5. 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 :

- ☐ 10
- ☐ 15
- ☐ 6
- ☐ 0

6. Le langage C est un langage
   - ☐ composé
   - ☐ compilé
   - ☐ interprété
   - ☐ lu, écrit, parlé
7. 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'analyse des entrées clavier
   - ☐ l'analyse harmonique
   - ☐ l'analyse sémantique
   - ☐ l'édition de liens
8. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ begin
   - ☐ init
   - ☐ include
   - ☐ main
9. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
   - ☐ sélectionner entre deux blocs à l'aide d'une condition
   - ☐ mettre les blocs en séquence les uns à la suite des autres
   - ☐ répéter un bloc tant qu'une condition est vérifée
   - ☐ retourner un bloc
10. Le code suivant :

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

affichera :

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

11. 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
12. Les lignes

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

- ☐ ne comportent aucune erreur
- ☐ 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

☐ comportent une erreur qui ne sera pas détectée

13. 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 5 et la variable y vaut 0

☐ la variable x vaut 0

☐ la variable y vaut 5

14. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

☐ `#appart <stdlib.h>`

☐ `#include <studlib.h>`

☐ `#include <stdio.h>`

☐ `#include <studio.h>`

15. Le code suivant :

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

affichera :

☐ 0 1 2 3 4

☐ 4 3 2 1

☐ 4 3 2 1 0

☐ 0 1 2 3

16. 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=0;i<5;i=i+1)`

☐ `for(i=0;i<=5;i=i+1)`

☐ `for(i=1;i<=5;i=i+1)`

☐ `for(i=1;i<5;i=i+1)`

17. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :

☐ en parallèle, chacun dans un registre

☐ tour à tour, un petit peu à chaque fois

☐ chacun son tour, après que le processus précédent a terminé

☐ tous ensemble

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

☐ changer de répertoire courant

☐ créer un répertoire

☐ créer un fichier texte

☐ ouvrir un fichier texte

19. 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 = 4`

☐ `j = %d`

☐ `j = 5`

☐ `j = 0`

20. 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 :

☐ 4

☐ 0

☐ 16

☐ 8

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Le langage C est un langage
   - ☐ composé
   - ☐ compilé
   - ☐ lu, écrit, parlé
   - ☐ interprété
2. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#include <studlib.h>`
   - ☐ `#include <stdio.h>`
   - ☐ `#include <studio.h>`
   - ☐ `#appart <stdlib.h>`
3. Une *segmentation fault* est une erreur qui survient lorsque :
   - ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
   - ☐ 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
   - ☐ 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
4. Quel est l'opérateur de différence en C :
   - ☐ $\neq$
   - ☐ `<>`
   - ☐ `!=`
   - ☐ `!`
5. Le code suivant :

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

   affichera :
   - ☐ `Mineur`
   - ☐ `Majeur`
   - ☐ rien
   - ☐ `Mineur`
`Majeur`
6. 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 x vaut 3
   - ☐ la variable y vaut 5
   - ☐ le programme affiche "Faux"
   - ☐ la variable x vaut 5 et la variable y vaut 3
7. Les lignes

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

   - ☐ comportent une erreur qui sera détectée au cours de l'analyse syntaxique
   - ☐ comportent une erreur qui ne sera pas détectée
   - ☐ comportent une erreur qui sera détectée au cours de l'édition de lien
   - ☐ ne comportent aucune erreur
8. Le code suivant :

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

   affichera :
   - ☐ 4 3 2 1
   - ☐ 0 1 2 3 4
   - ☐ 0 1 2 3
   - ☐ 4 3 2 1 0
9. Un bit est :
   - ☐ un chiffre binaire (0 ou 1)
   - ☐ un battement d'horloge processeur
   - ☐ l'instruction qui met fin à un programme
   - ☐ la longueur d'un mot mémoire
10. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :
    - ☐ en temps d'accès
    - ☐ les fichiers du disque
    - ☐ des processus
    - ☐ certaines données de la mémoire de travail
11. L'ordonnancement par tourniquet permet :
    - ☐ de doubler la mémoire disponible
    - ☐ de ne pas perdre de temps avec la commutation de contexte
    - ☐ d'afficher des ronds colorés à l'écran
    - ☐ d'entretenir l'illusion que les processus tournent en parallèle
12. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
    - ☐ analyse harmonique
    - ☐ analyse syntaxique
    - ☐ analyse sémantique
    - ☐ analyse lexicale

13. Un programme en langage C doit comporter une et une seule définition de la fonction :
    - ☐ begin
    - ☐ include
    - ☐ main
    - ☐ init

14. 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 `5`
    - ☐ le terminal affiche `x = 5`
    - ☐ le terminal affiche `x = 2`

15. 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 = %d`
    - ☐ `j = 0`

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

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

18. Sous unix (ou linux), la commande `cd` permet de :
    - ☐ détruire un fichier
    - ☐ changer de répertoire courant
    - ☐ ouvir un bureau partagé (common desktop)
    - ☐ récupérer un programme arrêté avec la commande `ab`
    - ☐ jouer de la musique

19. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
    - ☐ `int loop n;`
    - ☐ `int k;`
    - ☐ `int %d;`
    - ☐ `loop i;`

20. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
    - ☐ répéter un bloc tant qu'une condition est vérifée
    - ☐ retourner un bloc
    - ☐ sélectionner entre deux blocs à l'aide d'une condition
    - ☐ mettre les blocs en séquence les uns à la suite des autres

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Le code suivant :

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

affichera :

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

2. 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;`
3. Un bit est :
   - ☐ l'instruction qui met fin à un programme
   - ☐ un chiffre binaire (0 ou 1)
   - ☐ un battement d'horloge processeur
   - ☐ la longueur d'un mot mémoire
4. Le bus système sert à :
   - ☐ Écrire des données sur le dique dur
   - ☐ transporter les processus du tourniquet au processeur
   - ☐ Arriver à l'heure en cours
   - ☐ Transférer des données et intructions entre processeur et mémoire
5. Un registre du processeur est :
   - ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs
   - ☐ un composant qui contient la liste des fichiers du système
   - ☐ une unité de calcul spécialisée de l'ordinateur
   - ☐ une gamme de fréquence de fonctionnement du processeur
6. Quel est l'opérateur de différence en C :
   - ☐ `<>`
   - ☐ $\neq$
   - ☐ `!`
   - ☐ `!=`
7. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
   - ☐ `kwrite TP4`
   - ☐ `mkdir TP4`
   - ☐ `yppasswd`
   - ☐ `new TP4`
8. Le code suivant :

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

affichera :

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

9. Dans la commande gcc, l'option `-Wall` signifie :
   - ☐ qu'il faut lancer un déboggueur
   - ☐ qu'il faut indenter le fichier source
   - ☐ que l'on veut voir tous les avertissements
   - ☐ qu'on veut changer alétoirement de fond d'écran
10. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
    - ☐ `#include <studio.h>`
    - ☐ `#include <stdio.h>`
    - ☐ `#appart <stdlib.h>`
    - ☐ `#include <studlib.h>`
11. Le code suivant :

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

affichera :

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

12. Le code suivant :

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

affichera :

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

13. 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 :

- ☐ 6
- ☐ 15
- ☐ 10
- ☐ 0

14. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
    - ☐ analyse syntaxique
    - ☐ analyse harmonique
    - ☐ analyse sémantique
    - ☐ analyse lexicale
15. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
    - ☐ `gcc prog.exe -Wall -o prog.c`
    - ☐ `gcc prog.c -o -Wall prog.exe`
    - ☐ `gcc -Wall prog.exe -o prog.c`
    - ☐ `gcc -Wall prog.c -o prog.exe`
16. Une *segmentation fault* est une erreur qui survient lorsque :
    - ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
    - ☐ 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'accèder à une partie de la mémoire qui ne lui est pas réservée
    - ☐ la division du programme en zones homogènes échoue
17. Sous unix (ou linux), la commande `ls` permet de :
    - ☐ compiler un programme
    - ☐ afficher la liste de fichiers contenus dans un répertoire
    - ☐ voir des clips musicaux
    - ☐ afficher le contenu d'un fichier texte
18. 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 ?

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

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

20. Le code suivant :

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

affichera :

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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Une *segmentation fault* est une erreur qui survient lorsque :
   - ☐ le programme tente d'accèder à une partie de la mémoire qui ne lui est pas réservée
   - ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
   - ☐ la division du programme en zones homogènes échoue
   - ☐ 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

2. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ main
   - ☐ init
   - ☐ include
   - ☐ begin

3. Le code suivant :

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

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

4. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
   - ☐ `loop i;`
   - ☐ `int %d;`
   - ☐ `int k;`
   - ☐ `int loop n;`

5. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#include <studlib.h>`
   - ☐ `#include <stdio.h>`
   - ☐ `#include <studio.h>`
   - ☐ `#appart <stdlib.h>`

6. 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 :
   - ☐ 16
   - ☐ 0
   - ☐ 4
   - ☐ 8

7. 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

8. Sous unix (ou linux), la commande `ls` permet de :
   - ☐ afficher le contenu d'un fichier texte
   - ☐ voir des clips musicaux
   - ☐ compiler un programme
   - ☐ afficher la liste de fichiers contenus dans un répertoire

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

10. Sous unix (ou linux), la commande `cd` permet de :
    - ☐ jouer de la musique
    - ☐ récupérer un programme arrêté avec la commande `ab`
    - ☐ ouvir un bureau partagé (common desktop)
    - ☐ changer de répertoire courant
    - ☐ détruire un fichier

11. 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
```

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

12. 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
- ☐ 1 2 3 4
- ☐ 4 3 2 1 0

13. Le code suivant :

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

affichera :

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

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

15. Le code suivant :

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

affichera :

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

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

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

17. Le langage C est un langage

- ☐ lu, écrit, parlé
- ☐ interprété
- ☐ composé
- ☐ compilé

18. 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 `5`
- ☐ le terminal affiche `x = 5`

19. 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  }
```

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

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

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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
   - ☐ `gcc -Wall prog.c -o prog.exe`
   - ☐ `gcc prog.exe -Wall -o prog.c`
   - ☐ `gcc prog.c -o -Wall prog.exe`
   - ☐ `gcc -Wall prog.exe -o prog.c`

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

3. 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 :
   - ☐ 20
   - ☐ 16
   - ☐ 6
   - ☐ 3

4. 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 1 0 1 0 1 0 1`
   - ☐ `0 0 0 1 1 1`
   - ☐ `1 2 1 2 3`
   - ☐ `0 1 2 0 1 2`

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

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

7. Sous unix (ou linux), la commande `ls` permet de :
   - ☐ afficher le contenu d'un fichier texte
   - ☐ voir des clips musicaux
   - ☐ afficher la liste de fichiers contenus dans un répertoire
   - ☐ compiler un programme

8. Une *segmentation fault* est une erreur qui survient lorsque :
   - ☐ 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
   - ☐ la division du programme en zones homogènes échoue
   - ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
   - ☐ le programme tente d'accèder à une partie de la mémoire qui ne lui est pas réservée

9. 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;`

10. 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 :
    - ☐ 0
    - ☐ 8
    - ☐ 16
    - ☐ 4

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

12. Le code suivant :

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

affichera :

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

13. Quels calculs peut-on programmer en programmation structurée ?
    - ☐ certains programmes sont de vrais plats de spaghetti
    - ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine
    - ☐ 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

14. Sous unix (ou linux), la commande `cd` permet de :
    - ☐ récupérer un programme arrêté avec la commande `ab`
    - ☐ jouer de la musique
    - ☐ changer de répertoire courant
    - ☐ détruire un fichier
    - ☐ ouvir un bureau partagé (common desktop)

15. 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 `5`
    - ☐ le terminal affiche `x = 2`
    - ☐ le terminal affiche "Faux"
    - ☐ le terminal affiche `x = 5`

16. Quel est l'opérateur de différence en C :
    - ☐ $\neq$
    - ☐ `<>`
    - ☐ `!=`
    - ☐ `!`

17. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
    - ☐ `#define taille = N`
    - ☐ `#define N 3`
    - ☐ `#define N =  3`
    - ☐ `#define taille = 3`

18. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
    - ☐ `new TP4`
    - ☐ `yppasswd`
    - ☐ `mkdir TP4`
    - ☐ `kwrite TP4`

19. 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
    - ☐ des processus
    - ☐ les fichiers du disque
    - ☐ en temps d'accès

20. 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 1 2 3 0 1 2`
- ☐ `0 1 2 0 1 2 3`
- ☐ `0 0 1 1 2 2 3`
- ☐ `0 1 2 0 1 2`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. 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

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

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

3. Le code suivant :

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

affichera :

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

4. 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

5. 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 :

- ☐ 6
- ☐ 3
- ☐ 20
- ☐ 16

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

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

7. Le code suivant :

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

affichera :

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

8. Le code suivant :

```
int i;
for (i = 4; i > 0; 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

9. Un bit est :

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

10. 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
- ☐ un composant qui contient la liste des fichiers du système
- ☐ une unité de calcul spécialisée de l'ordinateur

11. 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 = 0`
- ☐ `j = 5`
- ☐ `j = %d`

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

13. Le langage C est un langage

- ☐ interprété
- ☐ lu, écrit, parlé
- ☐ composé
- ☐ compilé

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

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

15. Le code suivant :

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

affichera :

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

16. 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 :

- ☐ 6
- ☐ 0
- ☐ 15
- ☐ 10

17. 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 3
- ☐ le programme affiche "Faux"
- ☐ la variable x vaut 5 et la variable y vaut 3

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

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

19. 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>`

20. Le code suivant :

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

affichera :

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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Une *segmentation fault* est une erreur qui survient lorsque :
   - ☐ le programme tente d'accèder à une partie de la mémoire qui ne lui est pas réservée
   - ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
   - ☐ 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
   - ☐ la division du programme en zones homogènes échoue

2. Le code suivant :

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

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

3. Sous unix (ou linux), la commande `ls` permet de :
   - ☐ afficher le contenu d'un fichier texte
   - ☐ compiler un programme
   - ☐ afficher la liste de fichiers contenus dans un répertoire
   - ☐ voir des clips musicaux

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

5. Le code suivant :

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

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

6. Le code suivant :

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

   affichera :
   - ☐ 8 6 4 2
   - ☐ 8 2
   - ☐ 0 2 4 6 8
   - ☐ 8 6 4 2 0

7. 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
   - ☐ 4 3 2 1 0
   - ☐ 0 1 2 3 4

8. 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`
   - ☐ `1 2 3 1 2`
   - ☐ `0 0 1 1 2 2`
   - ☐ `0 1 2 0 1 2`

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

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

11. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
    - ☐ analyse syntaxique
    - ☐ analyse sémantique
    - ☐ analyse lexicale
    - ☐ analyse harmonique

12. 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
```

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

13. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ changer de répertoire courant
    - ☐ ouvrir un fichier texte
    - ☐ créer un fichier texte
    - ☐ créer un répertoire

14. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
    - ☐ `yppasswd`
    - ☐ `mkdir TP4`
    - ☐ `kwrite TP4`
    - ☐ `new TP4`

15. Si cette erreur apparaît à la compilation : `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
    - ☐ une accolade manquante
    - ☐ un point-virgule manquant
    - ☐ un point-virgule en trop
    - ☐ une accolade en trop

16. 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 :
    - ☐ 3
    - ☐ 16
    - ☐ 6
    - ☐ 20

17. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
    - ☐ `int[] new tableau(5);`
    - ☐ `int toto[taille=5];`
    - ☐ `char tableau[5];`
    - ☐ `int toto[5];`
    - ☐ `int tab[] = 5;`

18. 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=0;i<5;i=i+1)`
    - ☐ `for(i=1;i<5;i=i+1)`
    - ☐ `for(i=0;i<=5;i=i+1)`
    - ☐ `for(i=1;i<=5;i=i+1)`

19. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
    - ☐ `gcc -Wall prog.c -o prog.exe`
    - ☐ `gcc prog.c -o -Wall prog.exe`
    - ☐ `gcc prog.exe -Wall -o prog.c`
    - ☐ `gcc -Wall prog.exe -o prog.c`

20. Le code suivant :

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

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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
   - ☐ `#define N =  3`
   - ☐ `#define taille = 3`
   - ☐ `#define taille = N`
   - ☐ `#define N 3`
2. Sur unix (ou linux), la commande `mkdir` permet de :
   - ☐ changer de répertoire courant
   - ☐ créer un répertoire
   - ☐ ouvrir un fichier texte
   - ☐ créer un fichier texte
3. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
   - ☐ `kwrite TP4`
   - ☐ `yppasswd`
   - ☐ `new TP4`
   - ☐ `mkdir TP4`
4. 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
   - ☐ 15
   - ☐ 6
   - ☐ 10
5. 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=%d et y=%d\n",x,y);`
   - ☐ `printf("x=%x et y=%y\n");`
   - ☐ `printf("x=%d et y=%d\n",x y);`
6. 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
```

   - ☐ la case mémoire 8 contiendra 0
   - ☐ le bus explose
   - ☐ la case mémoire 8 contiendra 16
   - ☐ le terminal affiche 8
7. 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 x vaut 5 et la variable y vaut 3
   - ☐ le programme affiche "Faux"
   - ☐ la variable y vaut 5
   - ☐ la variable x vaut 3
8. 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 ?
   - ☐ `1 2 3 1 2`
   - ☐ `0 1 0 1 0 1`
   - ☐ `0 0 1 1 2 2`
   - ☐ `0 1 2 0 1 2`
9. 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
10. 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`
11. Le langage C est un langage
    - ☐ interprété
    - ☐ compilé
    - ☐ lu, écrit, parlé
    - ☐ composé

12. Quel est l'opérateur de différence en C :
    - ☐ `!`
    - ☐ `<>`
    - ☐ `!=`
    - ☐ $\neq$

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

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

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

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

17. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
    - ☐ `int k;`
    - ☐ `int %d;`
    - ☐ `loop i;`
    - ☐ `int loop n;`

18. 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

19. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
    - ☐ en parallèle, chacun dans un registre
    - ☐ chacun son tour, après que le processus précédent a terminé
    - ☐ tous ensemble
    - ☐ tour à tour, un petit peu à chaque fois

20. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
    - ☐ `gcc prog.c -o -Wall prog.exe`
    - ☐ `gcc -Wall prog.exe -o prog.c`
    - ☐ `gcc -Wall prog.c -o prog.exe`
    - ☐ `gcc prog.exe -Wall -o prog.c`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. 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");`

2. 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 :
   - ☐ 10
   - ☐ 0
   - ☐ 6
   - ☐ 15

3. Le code suivant :

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

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

4. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
   - ☐ tous ensemble
   - ☐ tour à tour, un petit peu à chaque fois
   - ☐ chacun son tour, après que le processus précédent a terminé
   - ☐ en parallèle, chacun dans un registre

5. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `int tab[] = 5;`
   - ☐ `int[] new tableau(5);`
   - ☐ `char tableau[5];`
   - ☐ `int toto[taille=5];`
   - ☐ `int toto[5];`

6. Le code suivant :

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

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

7. 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`

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

9. L'ordonnancement par tourniquet permet :
   - ☐ de ne pas perdre de temps avec la commutation de contexte
   - ☐ d'entretenir l'illusion que les processus tournent en parallèle
   - ☐ de doubler la mémoire disponible
   - ☐ d'afficher des ronds colorés à l'écran

10. 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=0;i<=5;i=i+1)`
    - ☐ `for(i=1;i<=5;i=i+1)`
    - ☐ `for(i=1;i<5;i=i+1)`
    - ☐ `for(i=0;i<5;i=i+1)`

11. 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 :
    - ☐ 8
    - ☐ 4
    - ☐ 16
    - ☐ 0

12. Le langage C est un langage
    - ☐ compilé
    - ☐ interprété
    - ☐ composé
    - ☐ lu, écrit, parlé

13. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
    - ☐ `gcc -Wall prog.exe -o prog.c`
    - ☐ `gcc prog.exe -Wall -o prog.c`
    - ☐ `gcc prog.c -o -Wall prog.exe`
    - ☐ `gcc -Wall prog.c -o prog.exe`

14. Les lignes

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

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

15. 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 `****`
    - ☐ le programme affiche x
    - ☐ la variable x vaut $-\frac{1}{2}$
    - ☐ la variable x vaut 16

16. Le code suivant :

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

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

17. 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 1 0 1 0 1 0 1`
    - ☐ `0 0 0 1 1 1`
    - ☐ `0 1 2 0 1 2`
    - ☐ `1 2 1 2 3`

18. Quel est l'opérateur de différence en C :
    - ☐ $\neq$
    - ☐ `!=`
    - ☐ `<>`
    - ☐ `!`

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

20. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ changer de répertoire courant
    - ☐ créer un fichier texte
    - ☐ créer un répertoire
    - ☐ ouvrir un fichier texte

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Une *segmentation fault* est une erreur qui survient lorsque :
   - ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
   - ☐ 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'accèder à une partie de la mémoire qui ne lui est pas réservée
   - ☐ la division du programme en zones homogènes échoue
2. Si cette erreur apparaît à la compilation : `Undefined symbols :"_prinft"` ou `référence indéfinie vers « prinft »` que doit-on chercher dans le programme ?
   - ☐ une variable non déclarée
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une faute de frappe dans un appel de fonction
   - ☐ un caractère interdit en C
3. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `char tableau[5];`
   - ☐ `int toto[5];`
   - ☐ `int tab[] = 5;`
   - ☐ `int[] new tableau(5);`
   - ☐ `int toto[taille=5];`
4. Les lignes

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

   - ☐ comportent une erreur qui sera détectée au cours de l'édition de lien
   - ☐ ne comportent aucune erreur
   - ☐ comportent une erreur qui sera détectée au cours de l'analyse syntaxique
   - ☐ comportent une erreur qui ne sera pas détectée
5. 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 :
   - ☐ 8
   - ☐ 0
   - ☐ 4
   - ☐ 16
6. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#include <stdio.h>`
   - ☐ `#include <studlib.h>`
   - ☐ `#appart <stdlib.h>`
   - ☐ `#include <studio.h>`
7. Le code suivant :

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

   affichera :
   - ☐ 0 1 2 3 4
   - ☐ 4 3 2 1 0
   - ☐ 1 2 3 4
   - ☐ 4 3 2 1
8. 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=0;i<=5;i=i+1)`
   - ☐ `for(i=0;i<5;i=i+1)`
   - ☐ `for(i=1;i<5;i=i+1)`
   - ☐ `for(i=1;i<=5;i=i+1)`
9. 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 7
   - ☐ 0 1 2 3 4 5 6
   - ☐ 0 2 4 6
   - ☐ 0 2 4 6 8
10. 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`
11. Un registre du processeur est :
    - ☐ une gamme de fréquence de fonctionnement du processeur
    - ☐ un composant qui contient la liste des fichiers du système
    - ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs
    - ☐ une unité de calcul spécialisée de l'ordinateur

12. Le code suivant :

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

affichera :

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

13. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
    - ☐ répéter un bloc tant qu'une condition est vérifée
    - ☐ sélectionner entre deux blocs à l'aide d'une condition
    - ☐ mettre les blocs en séquence les uns à la suite des autres
    - ☐ retourner un bloc

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

15. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
    - ☐ `#define taille = N`
    - ☐ `#define N =  3`
    - ☐ `#define N 3`
    - ☐ `#define taille = 3`

16. 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 :
- ☐ 6
- ☐ 20
- ☐ 16
- ☐ 3

17. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
    - ☐ analyse syntaxique
    - ☐ analyse harmonique
    - ☐ analyse lexicale
    - ☐ analyse sémantique

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

19. Le code suivant :

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

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

20. Quel est l'opérateur de différence en C :
    - ☐ `<>`
    - ☐ `!=`
    - ☐ `!`
    - ☐ $\neq$

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Le bus système sert à :
   - ☐ Transférer des données et intructions entre processeur et mémoire
   - ☐ Écrire des données sur le dique dur
   - ☐ Arriver à l'heure en cours
   - ☐ transporter les processus du tourniquet au processeur
2. 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 variable non déclarée
   - ☐ une faute de frappe dans un appel de fonction
3. Sur unix (ou linux), la commande `mkdir` permet de :
   - ☐ créer un fichier texte
   - ☐ changer de répertoire courant
   - ☐ ouvrir un fichier texte
   - ☐ créer un répertoire
4. Le langage C est un langage
   - ☐ composé
   - ☐ compilé
   - ☐ interprété
   - ☐ lu, écrit, parlé
5. Le code suivant :

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

   affichera :
   - ☐ rien
   - ☐ `Mineur`
`Majeur`
   - ☐ `Majeur`
   - ☐ `Mineur`
6. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#include <stdio.h>`
   - ☐ `#include <studio.h>`
   - ☐ `#include <studlib.h>`
   - ☐ `#appart <stdlib.h>`
7. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
   - ☐ tous ensemble
   - ☐ en parallèle, chacun dans un registre
   - ☐ tour à tour, un petit peu à chaque fois
   - ☐ chacun son tour, après que le processus précédent a terminé
8. 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 1 2 0 1 2 3`
   - ☐ `0 0 1 1 2 2 3`
   - ☐ `0 1 2 3 0 1 2`
   - ☐ `0 1 2 0 1 2`
9. Le code suivant :

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

   affichera :
   - ☐ 4 3 2 1
   - ☐ 4 3 2 1 0
   - ☐ 0 1 2 3 4
   - ☐ 1 2 3 4
10. Sous unix (ou linux), la commande `ls` permet de :
    - ☐ afficher la liste de fichiers contenus dans un répertoire
    - ☐ compiler un programme
    - ☐ afficher le contenu d'un fichier texte
    - ☐ voir des clips musicaux
11. Quels calculs peut-on programmer en programmation structurée ?
    - ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine
    - ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine
    - ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée
    - ☐ certains programmes sont de vrais plats de spaghetti
12. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
    - ☐ mettre les blocs en séquence les uns à la suite des autres
    - ☐ sélectionner entre deux blocs à l'aide d'une condition
    - ☐ retourner un bloc
    - ☐ répéter un bloc tant qu'une condition est vérifée

13. Un fichier source est :

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

14. Les lignes

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

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

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

16. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#include <studio.h>`
- ☐ `#appart <stdlib.h>`
- ☐ `#include <stdio.h>`
- ☐ `#include <studlib.h>`

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

18. Dans la commande gcc, l'option `-Wall` signifie :

- ☐ qu'il faut lancer un déboggueur
- ☐ qu'on veut changer alétoirement de fond d'écran
- ☐ que l'on veut voir tous les avertissements
- ☐ qu'il faut indenter le fichier source

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

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

20. 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 :

- ☐ 20
- ☐ 16
- ☐ 6
- ☐ 3

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

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

2. Le code suivant :

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

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

3. Le langage C est un langage
   - ☐ composé
   - ☐ lu, écrit, parlé
   - ☐ compilé
   - ☐ interprété

4. Quel est l'opérateur de différence en C :
   - ☐ $\neq$
   - ☐ `<>`
   - ☐ `!=`
   - ☐ `!`

5. Le code suivant :

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

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

6. 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
   - ☐ certains programmes sont de vrais plats de spaghetti
   - ☐ 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

7. Si cette erreur apparaît à la compilation :
   `Undefined symbols :"_prinft"` ou
   `référence indéfinie vers « prinft »` que doit-on chercher dans le programme ?
   - ☐ une variable non déclarée
   - ☐ un caractère interdit en C
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une faute de frappe dans un appel de fonction

8. 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 :
   - ☐ 4
   - ☐ 8
   - ☐ 0
   - ☐ 16

9. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
   - ☐ `gcc prog.exe -Wall -o prog.c`
   - ☐ `gcc -Wall prog.c -o prog.exe`
   - ☐ `gcc -Wall prog.exe -o prog.c`
   - ☐ `gcc prog.c -o -Wall prog.exe`

10. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
    - ☐ `#include <studio.h>`
    - ☐ `#include <stdio.h>`
    - ☐ `#appart <stdlib.h>`
    - ☐ `#include <studlib.h>`

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

12. 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 5 et la variable y vaut 0
- ☐ la variable x vaut 0
- ☐ la variable y vaut 5

13. 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 `5`
- ☐ le terminal affiche `x = 5`
- ☐ le terminal affiche "Faux"
- ☐ le terminal affiche `x = 2`

14. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
    - ☐ 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
    - ☐ retourner un bloc

15. 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 2 4 6 8
- ☐ 0 2 4 6
- ☐ 0 1 2 3 4 5 6 7

16. Le code suivant :

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

affichera :

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

17. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
    - ☐ chacun son tour, après que le processus précédent a terminé
    - ☐ tour à tour, un petit peu à chaque fois
    - ☐ en parallèle, chacun dans un registre
    - ☐ tous ensemble

18. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
    - ☐ `kwrite TP4`
    - ☐ `yppasswd`
    - ☐ `mkdir TP4`
    - ☐ `new TP4`

19. 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 1 2 0 1 2`
- ☐ `1 2 1 2 3`
- ☐ `0 0 0 1 1 1`
- ☐ `0 1 0 1 0 1 0 1`

20. 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)`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
   - ☐ mettre les blocs en séquence les uns à la suite des autres
   - ☐ sélectionner entre deux blocs à l'aide d'une condition
   - ☐ répéter un bloc tant qu'une condition est vérifée
   - ☐ retourner un bloc
2. 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
   - ☐ une faute de frappe dans un appel de fonction
   - ☐ un caractère interdit en C
   - ☐ une variable non déclarée
3. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
   - ☐ `int %d;`
   - ☐ `int k;`
   - ☐ `int loop n;`
   - ☐ `loop i;`
4. Le bus système sert à :
   - ☐ transporter les processus du tourniquet au processeur
   - ☐ Arriver à l'heure en cours
   - ☐ Transférer des données et intructions entre processeur et mémoire
   - ☐ Écrire des données sur le dique dur
5. 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=%d et y=%d\n,x,y");`
   - ☐ `printf("x=%x et y=%y\n");`
   - ☐ `printf("x=%d et y=%d\n",x,y);`
6. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :
   - ☐ les fichiers du disque
   - ☐ certaines données de la mémoire de travail
   - ☐ des processus
   - ☐ en temps d'accès
7. Le code suivant :

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

   affichera :
   - ☐ 0 1 2 3 4
   - ☐ 0 1 2 3
   - ☐ 4 3 2 1 0
   - ☐ 4 3 2 1
8. Le code suivant :

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

   affichera :
   - ☐ `Mineur`
`Majeur`
   - ☐ `Mineur`
   - ☐ `Majeur`
   - ☐ rien
9. 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
10. Un programme en langage C doit comporter une et une seule définition de la fonction :
    - ☐ main
    - ☐ include
    - ☐ begin
    - ☐ init
11. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
    - ☐ `#include <studio.h>`
    - ☐ `#include <studlib.h>`
    - ☐ `#appart <stdlib.h>`
    - ☐ `#include <stdio.h>`
12. 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 ?
    - ☐ `** ** ** ** ** **`
    - ☐ `** *** **** *****`
    - ☐ `**** **** **** ****`
    - ☐ `***** **** *** ***`

13. Le code suivant :

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

affichera :

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

14. Le code suivant :

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

affichera :

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

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

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

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

17. 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=1;i<5;i=i+1)`
- ☐ `for(i=0;i<5;i=i+1)`

18. 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 = 0`
- ☐ `j = %d`
- ☐ `j = 4`
- ☐ `j = 5`

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

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

20. 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 sémantique
- ☐ l'analyse harmonique
- ☐ l'analyse des entrées clavier

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Le code suivant :

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

affichera :

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

2. Le code suivant :

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

affichera :

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

3. Les lignes

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

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

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

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

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

6. 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 = %d`
- ☐ `j = 4`
- ☐ `j = 0`

7. 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 :

- ☐ 10
- ☐ 0
- ☐ 15
- ☐ 6

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

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

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

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

10. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#include <stdio.h>`
- ☐ `#include <studlib.h>`
- ☐ `#appart <stdlib.h>`
- ☐ `#include <studio.h>`

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

12. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
    - ☐ analyse syntaxique
    - ☐ analyse sémantique
    - ☐ analyse harmonique
    - ☐ analyse lexicale
13. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
    - ☐ sélectionner entre deux blocs à l'aide d'une condition
    - ☐ répéter un bloc tant qu'une condition est vérifée
    - ☐ retourner un bloc
    - ☐ mettre les blocs en séquence les uns à la suite des autres
14. 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
    - ☐ le programme affiche "Faux"
    - ☐ la variable x vaut 5 et la variable y vaut 3
    - ☐ la variable x vaut 3
15. 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
16. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
    - ☐ `int loop n;`
    - ☐ `int %d;`
    - ☐ `loop i;`
    - ☐ `int k;`
17. 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
    - ☐ les fichiers du disque
    - ☐ des processus
18. Le code suivant :

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

    affichera :
    - ☐ 1 2 3 4
    - ☐ 4 3 2 1 0
    - ☐ 4 3 2 1
    - ☐ 0 1 2 3 4
19. Le code suivant :

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

    affichera :
    - ☐ `Majeur`
    - ☐ `Mineur`
    - ☐ rien
    - ☐ `Mineur`
`Majeur`
20. Le code suivant :

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

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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Le code suivant :

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

affichera :

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

2. 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
```

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

3. Sur unix (ou linux), la commande `mkdir` permet de :
   - ☐ créer un fichier texte
   - ☐ ouvrir un fichier texte
   - ☐ changer de répertoire courant
   - ☐ créer un répertoire

4. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `char tableau[5];`
   - ☐ `int tab[] = 5;`
   - ☐ `int[] new tableau(5);`
   - ☐ `int toto[5];`
   - ☐ `int toto[taille=5];`

5. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
   - ☐ `kwrite TP4`
   - ☐ `new TP4`
   - ☐ `yppasswd`
   - ☐ `mkdir TP4`

6. Le code suivant :

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

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

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

8. 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  }
```

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

9. Le code suivant :

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

affichera :
- ☐ 6
- ☐ 1
- ☐ 42
- ☐ 0

10. 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 = 5`
- ☐ `j = 0`
- ☐ `j = %d`
- ☐ `j = 4`

11. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :

- ☐ en parallèle, chacun dans un registre
- ☐ tour à tour, un petit peu à chaque fois
- ☐ tous ensemble
- ☐ chacun son tour, après que le processus précédent a terminé

12. 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=0;i<5;i=i+1)`
- ☐ `for(i=1;i<5;i=i+1)`
- ☐ `for(i=1;i<=5;i=i+1)`
- ☐ `for(i=0;i<=5;i=i+1)`

13. Le langage C est un langage

- ☐ composé
- ☐ lu, écrit, parlé
- ☐ interprété
- ☐ compilé

14. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#include <studlib.h>`
- ☐ `#include <stdio.h>`
- ☐ `#appart <stdlib.h>`
- ☐ `#include <studio.h>`

15. 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 :

- ☐ 10
- ☐ 6
- ☐ 0
- ☐ 15

16. Un fichier source est :

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

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

18. 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 variable non déclarée
- ☐ une faute de frappe dans un appel de fonction

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

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

20. 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`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 points par réponse juste (unique) ; −0,5 points par réponse fausse. Durée : 20 minutes.

1. 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)

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

3. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#include <studlib.h>`
   - ☐ `#include <studio.h>`
   - ☐ `#appart <stdlib.h>`
   - ☐ `#include <stdio.h>`

4. 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 = 5`
   - ☐ le terminal affiche `5`
   - ☐ le terminal affiche `x = 2`

5. 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 ?
   - ☐ `1 2 3 1 2`
   - ☐ `0 0 1 1 2 2`
   - ☐ `0 1 0 1 0 1`
   - ☐ `0 1 2 0 1 2`

6. 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
   - ☐ 0
   - ☐ 4
   - ☐ 8

7. Le code suivant :

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

   affichera :
   - ☐ 42
   - ☐ 1
   - ☐ 6
   - ☐ 0

8. Si cette erreur apparaît à la compilation : `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
   - ☐ une accolade manquante
   - ☐ un point-virgule en trop
   - ☐ un point-virgule manquant
   - ☐ une accolade en trop

9. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `char tableau[5];`
   - ☐ `int toto[5];`
   - ☐ `int tab[] = 5;`
   - ☐ `int toto[taille=5];`
   - ☐ `int[] new tableau(5);`

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

11. 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
    - ☐ 15
    - ☐ 10
    - ☐ 6

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

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

13. 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  }
```

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

14. 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
- ☐ le bus explose
- ☐ la case mémoire 8 contiendra 0
- ☐ la case mémoire 8 contiendra 16

15. Un fichier source est :

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

16. Le code suivant :

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

affichera :

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

17. 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
- ☐ 4 3 2 1 0
- ☐ 0 1 2 3 4

18. Le code suivant :

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

affichera :

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

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

- ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
- ☐ la division du programme en zones homogènes échoue
- ☐ 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'accèder à une partie de la mémoire qui ne lui est pas réservée

20. 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=0;i<=5;i=i+1)`
- ☐ `for(i=1;i<=5;i=i+1)`
- ☐ `for(i=0;i<5;i=i+1)`
- ☐ `for(i=1;i<5;i=i+1)`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Un bit est :
   - ☐ un battement d'horloge processeur
   - ☐ l'instruction qui met fin à un programme
   - ☐ la longueur d'un mot mémoire
   - ☐ un chiffre binaire (0 ou 1)
2. Le langage C est un langage
   - ☐ lu, écrit, parlé
   - ☐ interprété
   - ☐ compilé
   - ☐ composé
3. Le code suivant :

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

   affichera :
   - ☐ `Majeur`
   - ☐ rien
   - ☐ `Mineur`
   - ☐ `Mineur`
     `Majeur`
4. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
   - ☐ sélectionner entre deux blocs à l'aide d'une condition
   - ☐ mettre les blocs en séquence les uns à la suite des autres
   - ☐ retourner un bloc
   - ☐ répéter un bloc tant qu'une condition est vérifée
5. 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
6. 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 1 2 0 1 2 3`
   - ☐ `0 1 2 0 1 2`
   - ☐ `0 1 2 3 0 1 2`
   - ☐ `0 0 1 1 2 2 3`
7. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#include <studlib.h>`
   - ☐ `#include <studio.h>`
   - ☐ `#appart <stdlib.h>`
   - ☐ `#include <stdio.h>`
8. Dans la commande gcc, l'option `-Wall` signifie :
   - ☐ qu'on veut changer alétoirement de fond d'écran
   - ☐ qu'il faut indenter le fichier source
   - ☐ qu'il faut lancer un déboggueur
   - ☐ que l'on veut voir tous les avertissements
9. 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;`
10. Sous unix (ou linux), la commande `cd` permet de :
    - ☐ jouer de la musique
    - ☐ récupérer un programme arrêté avec la commande `ab`
    - ☐ changer de répertoire courant
    - ☐ détruire un fichier
    - ☐ ouvir un bureau partagé (common desktop)
11. Le code suivant :

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

    affichera :
    - ☐ 0 1 2 3 4
    - ☐ 4 3 2 1
    - ☐ 1 2 3 4
    - ☐ 4 3 2 1 0
12. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
    - ☐ `#include <studlib.h>`
    - ☐ `#include <studio.h>`
    - ☐ `#appart <stdlib.h>`
    - ☐ `#include <stdio.h>`
13. Sous unix (ou linux), la commande `ls` permet de :
    - ☐ afficher la liste de fichiers contenus dans un répertoire
    - ☐ compiler un programme
    - ☐ voir des clips musicaux
    - ☐ afficher le contenu d'un fichier texte

14. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :

- ☐ en parallèle, chacun dans un registre
- ☐ tour à tour, un petit peu à chaque fois
- ☐ chacun son tour, après que le processus précédent a terminé
- ☐ tous ensemble

15. 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 = %d`
- ☐ `j = 0`

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

- ☐ le programme tente d'accèder à une partie de la mémoire qui ne lui est pas réservée
- ☐ la division du programme en zones homogènes échoue
- ☐ 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

17. 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
```

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

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

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

19. Le code suivant :

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

affichera :

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

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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. 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'analyse sémantique
   - ☐ l'analyse harmonique
   - ☐ l'édition de liens
   - ☐ l'analyse des entrées clavier
2. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ init
   - ☐ include
   - ☐ main
   - ☐ begin
3. Le code suivant :

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

   affichera :
   - ☐ `Mineur`
   - ☐ rien
   - ☐ `Mineur`
     `Majeur`
   - ☐ `Majeur`
4. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
   - ☐ sélectionner entre deux blocs à l'aide d'une condition
   - ☐ retourner un bloc
   - ☐ répéter un bloc tant qu'une condition est vérifée
   - ☐ mettre les blocs en séquence les uns à la suite des autres
5. Le code suivant :

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

   affichera :
   - ☐ 8 6 4 2 0
   - ☐ 8 6 4 2
   - ☐ 0 2 4 6 8
   - ☐ 8 2
6. 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 :
   - ☐ 6
   - ☐ 3
   - ☐ 16
   - ☐ 20
7. Le code suivant :

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

   affichera :
   - ☐ 1 2 3 4
   - ☐ 4 3 2 1
   - ☐ 4 3 2 1 0
   - ☐ 0 1 2 3 4
8. 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 bus explose
   - ☐ la case mémoire 8 contiendra 0
   - ☐ la case mémoire 8 contiendra 16
   - ☐ le terminal affiche 8
9. Quels calculs peut-on programmer en programmation structurée ?
   - ☐ certains programmes sont de vrais plats de spaghetti
   - ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine
   - ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine
   - ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée
10. Le bus système sert à :
    - ☐ Écrire des données sur le dique dur
    - ☐ transporter les processus du tourniquet au processeur
    - ☐ Transférer des données et intructions entre processeur et mémoire
    - ☐ Arriver à l'heure en cours
11. Un registre du processeur est :
    - ☐ une gamme de fréquence de fonctionnement du processeur
    - ☐ une unité de calcul spécialisée de l'ordinateur
    - ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs
    - ☐ un composant qui contient la liste des fichiers du système

12. 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 = 0`
- ☐ `j = 5`
- ☐ `j = 4`
- ☐ `j = %d`

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

- ☐ `!=`
- ☐ $\neq$
- ☐ `<>`
- ☐ `!`

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

15. 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

16. 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 :

- ☐ 4
- ☐ 0
- ☐ 16
- ☐ 8

17. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#include <stdio.h>`
- ☐ `#include <studio.h>`
- ☐ `#appart <stdlib.h>`
- ☐ `#include <studlib.h>`

18. 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 7
- ☐ 0 2 4 6 8
- ☐ 0 1 2 3 4 5 6
- ☐ 0 2 4 6

19. Un fichier source est :

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

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

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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Le code suivant :

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

affichera :

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

2. 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  }
```

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

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

4. 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 :
- ☐ 20
- ☐ 6
- ☐ 3
- ☐ 16

5. Sous unix (ou linux), la commande `cd` permet de :
   - ☐ changer de répertoire courant
   - ☐ détruire un fichier
   - ☐ jouer de la musique
   - ☐ récupérer un programme arrêté avec la commande `ab`
   - ☐ ouvir un bureau partagé (common desktop)

6. Le langage C est un langage
   - ☐ compilé
   - ☐ lu, écrit, parlé
   - ☐ interprété
   - ☐ composé

7. 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 :
- ☐ 0
- ☐ 4
- ☐ 8
- ☐ 16

8. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
   - ☐ répéter un bloc tant qu'une condition est vérifée
   - ☐ retourner un bloc
   - ☐ sélectionner entre deux blocs à l'aide d'une condition
   - ☐ mettre les blocs en séquence les uns à la suite des autres

9. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
   - ☐ `int %d;`
   - ☐ `int loop n;`
   - ☐ `int k;`
   - ☐ `loop i;`

10. 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

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

12. Le code suivant :

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

affichera :

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

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

- ☐ $\neq$
- ☐ `!=`
- ☐ `<>`
- ☐ `!`

14. 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 ?

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

15. 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 = 5`
- ☐ `j = 0`
- ☐ `j = %d`
- ☐ `j = 4`

16. Dans la commande gcc, l'option `-Wall` signifie :

- ☐ qu'il faut lancer un déboggueur
- ☐ qu'on veut changer alétoirement de fond d'écran
- ☐ qu'il faut indenter le fichier source
- ☐ que l'on veut voir tous les avertissements

17. 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  }
```

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

18. Le code suivant :

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

affichera :

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

19. Le code suivant :

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

affichera :

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

20. Quels calculs peut-on programmer en programmation structurée ?

- ☐ certains programmes sont de vrais plats de spaghetti
- ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine
- ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée
- ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Les lignes

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

☐ ne comportent aucune erreur
☐ 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

2. 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 = 0`
☐ `j = 5`
☐ `j = %d`

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

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

☐ la division du programme en zones homogènes échoue
☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
☐ 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

5. Le code suivant :

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

affichera :

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

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

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

7. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :

☐ tous ensemble
☐ tour à tour, un petit peu à chaque fois
☐ en parallèle, chacun dans un registre
☐ chacun son tour, après que le processus précédent a terminé

8. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

☐ `#include <studlib.h>`
☐ `#include <stdio.h>`
☐ `#appart <stdlib.h>`
☐ `#include <studio.h>`

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

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

10. 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 :

☐ 0
☐ 16
☐ 4
☐ 8

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

12. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
    - ☐ `#appart <stdlib.h>`
    - ☐ `#include <stdio.h>`
    - ☐ `#include <studio.h>`
    - ☐ `#include <studlib.h>`

13. Le code suivant :

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

affichera :

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

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

15. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
    - ☐ mettre les blocs en séquence les uns à la suite des autres
    - ☐ sélectionner entre deux blocs à l'aide d'une condition
    - ☐ répéter un bloc tant qu'une condition est vérifée
    - ☐ retourner un bloc

16. Quels calculs peut-on programmer en programmation structurée ?
    - ☐ certains programmes sont de vrais plats de spaghetti
    - ☐ 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

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

18. Le code suivant :

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

affichera :

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

19. Si cette erreur apparaît à la compilation : `Undefined symbols :"_prinft"` ou `référence indéfinie vers` « `prinft` » que doit-on chercher dans le programme ?
    - ☐ un caractère interdit en C
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une variable non déclarée
    - ☐ une faute de frappe dans un appel de fonction

20. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
    - ☐ `loop i;`
    - ☐ `int k;`
    - ☐ `int loop n;`
    - ☐ `int %d;`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Dans la commande gcc, l'option `-Wall` signifie :
   - ☐ qu'il faut indenter le fichier source
   - ☐ que l'on veut voir tous les avertissements
   - ☐ qu'il faut lancer un déboggueur
   - ☐ qu'on veut changer alétoirement de fond d'écran
2. L'ordonnancement par tourniquet permet :
   - ☐ d'afficher des ronds colorés à l'écran
   - ☐ de doubler la mémoire disponible
   - ☐ d'entretenir l'illusion que les processus tournent en parallèle
   - ☐ de ne pas perdre de temps avec la commutation de contexte
3. Un registre du processeur est :
   - ☐ un composant qui contient la liste des fichiers du système
   - ☐ une unité de calcul spécialisée de l'ordinateur
   - ☐ une gamme de fréquence de fonctionnement du processeur
   - ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs
4. Le code suivant :

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

   affichera :
   - ☐ 8 6 4 2
   - ☐ 8 2
   - ☐ 8 6 4 2 0
   - ☐ 0 2 4 6 8
5. 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 0
   - ☐ la case mémoire 8 contiendra 16
   - ☐ le bus explose
6. 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
   - ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine
   - ☐ certains programmes sont de vrais plats de spaghetti
   - ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine
7. Sur unix (ou linux), la commande `mkdir` permet de :
   - ☐ créer un fichier texte
   - ☐ créer un répertoire
   - ☐ ouvrir un fichier texte
   - ☐ changer de répertoire courant
8. 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 :
   - ☐ 6
   - ☐ 16
   - ☐ 3
   - ☐ 20
9. Le bus système sert à :
   - ☐ Transférer des données et intructions entre processeur et mémoire
   - ☐ transporter les processus du tourniquet au processeur
   - ☐ Arriver à l'heure en cours
   - ☐ Écrire des données sur le dique dur
10. Si cette erreur apparaît à la compilation : `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
    - ☐ un point-virgule manquant
    - ☐ un point-virgule en trop
    - ☐ une accolade en trop
    - ☐ une accolade manquante
11. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
    - ☐ `#appart <stdlib.h>`
    - ☐ `#include <studio.h>`
    - ☐ `#include <stdio.h>`
    - ☐ `#include <studlib.h>`
12. 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=%d et y=%d\n,x,y");`
    - ☐ `printf("x=%d et y=%d\n",x,y);`
    - ☐ `printf("x=%x et y=%y\n");`
13. 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
    - ☐ une variable non déclarée
    - ☐ un caractère interdit en C
    - ☐ une faute de frappe dans un appel de fonction

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

15. Le langage C est un langage

- ☐ interprété
- ☐ composé
- ☐ compilé
- ☐ lu, écrit, parlé

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

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

17. Le code suivant :

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

affichera :

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

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

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

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

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

- ☐ $\neq$
- ☐ `<>`
- ☐ `!`
- ☐ `!=`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. 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

2. Sur unix (ou linux), la commande `mkdir` permet de :
   - ☐ créer un fichier texte
   - ☐ créer un répertoire
   - ☐ changer de répertoire courant
   - ☐ ouvrir un fichier texte

3. 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

4. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
   - ☐ `kwrite TP4`
   - ☐ `yppasswd`
   - ☐ `mkdir TP4`
   - ☐ `new TP4`

5. 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

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

7. Une *segmentation fault* est une erreur qui survient lorsque :
   - ☐ la division du programme en zones homogènes échoue
   - ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
   - ☐ 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'accèder à une partie de la mémoire qui ne lui est pas réservée

8. Quel est l'opérateur de différence en C :
   - ☐ `!=`
   - ☐ `!`
   - ☐ `<>`
   - ☐ $\neq$

9. 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 y vaut 5
   - ☐ la variable x vaut 5 et la variable y vaut 0
   - ☐ la variable x vaut 0

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

11. Si cette erreur apparaît à la compilation : `Undefined symbols :"_prinft"` ou `référence indéfinie vers « prinft »` que doit-on chercher dans le programme ?
    - ☐ une variable non déclarée
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une faute de frappe dans un appel de fonction
    - ☐ un caractère interdit en C

12. 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
```

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

13. L'ordonnancement par tourniquet permet :
    - ☐ d'entretenir l'illusion que les processus tournent en parallèle
    - ☐ d'afficher des ronds colorés à l'écran
    - ☐ de doubler la mémoire disponible
    - ☐ de ne pas perdre de temps avec la commutation de contexte

14. 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 fichier texte qui sera traduit en instructions processeur
    - ☐ un document qui doit être protégé
    - ☐ un document illisible pour les humains
15. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
    - ☐ `int k;`
    - ☐ `int %d;`
    - ☐ `int loop n;`
    - ☐ `loop i;`
16. 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`

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

18. Le code suivant :

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

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

19. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
    - ☐ `#define taille = N`
    - ☐ `#define N 3`
    - ☐ `#define taille = 3`
    - ☐ `#define N =  3`
20. Si cette erreur apparaît à la compilation :
    `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
    - ☐ une accolade en trop
    - ☐ une accolade manquante
    - ☐ un point-virgule en trop
    - ☐ un point-virgule manquant

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 points par réponse juste (unique) ; −0,5 points par réponse fausse. Durée : 20 minutes.

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

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

3. Le code suivant :

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

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

4. Le code suivant :

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

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

5. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
   - ☐ mettre les blocs en séquence les uns à la suite des autres
   - ☐ sélectionner entre deux blocs à l'aide d'une condition
   - ☐ répéter un bloc tant qu'une condition est vérifée
   - ☐ retourner un bloc

6. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ include
   - ☐ main
   - ☐ begin
   - ☐ init

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

8. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
   - ☐ en parallèle, chacun dans un registre
   - ☐ chacun son tour, après que le processus précédent a terminé
   - ☐ tour à tour, un petit peu à chaque fois
   - ☐ tous ensemble

9. 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)`

10. 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

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

12. 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 8
- ☐ 0 2 4 6

13. Le code suivant :

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

affichera :

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

14. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
    - ☐ `int toto[taille=5];`
    - ☐ `int tab[] = 5;`
    - ☐ `char tableau[5];`
    - ☐ `int toto[5];`
    - ☐ `int[] new tableau(5);`

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

16. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :
    - ☐ des processus
    - ☐ les fichiers du disque
    - ☐ certaines données de la mémoire de travail
    - ☐ en temps d'accès

17. Si cette erreur apparaît à la compilation : `Undefined symbols :"_prinft"` ou `référence indéfinie vers` « `prinft` » que doit-on chercher dans le programme ?
    - ☐ une variable non déclarée
    - ☐ une faute de frappe dans un appel de fonction
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ un caractère interdit en C

18. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
    - ☐ `#appart <stdlib.h>`
    - ☐ `#include <stdio.h>`
    - ☐ `#include <studlib.h>`
    - ☐ `#include <studio.h>`

19. Sous unix (ou linux), la commande `cd` permet de :
    - ☐ changer de répertoire courant
    - ☐ récupérer un programme arrêté avec la commande `ab`
    - ☐ ouvir un bureau partagé (common desktop)
    - ☐ jouer de la musique
    - ☐ détruire un fichier

20. Les lignes

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

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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. 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  }
```

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

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

3. Le code suivant :

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

   affichera :

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

4. 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  }
```

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

5. 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 ?

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

6. Sous unix (ou linux), la commande `ls` permet de :
   - ☐ afficher le contenu d'un fichier texte
   - ☐ afficher la liste de fichiers contenus dans un répertoire
   - ☐ compiler un programme
   - ☐ voir des clips musicaux

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

8. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#include <studio.h>`
   - ☐ `#include <stdio.h>`
   - ☐ `#appart <stdlib.h>`
   - ☐ `#include <studlib.h>`

9. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
   - ☐ tous ensemble
   - ☐ en parallèle, chacun dans un registre
   - ☐ chacun son tour, après que le processus précédent a terminé
   - ☐ tour à tour, un petit peu à chaque fois

10. Le code suivant :

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

    affichera :

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

11. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
    - ☐ `int %d;`
    - ☐ `loop i;`
    - ☐ `int k;`
    - ☐ `int loop n;`
12. 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 :
    - ☐ 20
    - ☐ 6
    - ☐ 3
    - ☐ 16
13. Sous unix (ou linux), la commande `cd` permet de :
    - ☐ changer de répertoire courant
    - ☐ détruire un fichier
    - ☐ récupérer un programme arrêté avec la commande `ab`
    - ☐ ouvir un bureau partagé (common desktop)
    - ☐ jouer de la musique
14. 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'édition de lien
    - ☐ ne comportent aucune erreur
    - ☐ comportent une erreur qui sera détectée au cours de l'analyse syntaxique
15. Un fichier source est :
    - ☐ un document de référence du système
    - ☐ un document qui doit être protégé
    - ☐ un document illisible pour les humains
    - ☐ un fichier texte qui sera traduit en instructions processeur
    - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
16. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
    - ☐ `yppasswd`
    - ☐ `kwrite TP4`
    - ☐ `new TP4`
    - ☐ `mkdir TP4`
17. 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 `x = 5`
    - ☐ le terminal affiche `x = 2`
    - ☐ le terminal affiche "Faux"
    - ☐ le terminal affiche `5`
18. 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  }
```

    - ☐ la variable x vaut 16
    - ☐ le programme affiche x
    - ☐ le programme affiche `****`
    - ☐ la variable x vaut $-\frac{1}{2}$
19. Un bit est :
    - ☐ l'instruction qui met fin à un programme
    - ☐ la longueur d'un mot mémoire
    - ☐ un battement d'horloge processeur
    - ☐ un chiffre binaire (0 ou 1)
20. 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
```

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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Le langage C est un langage
   - ☐ composé
   - ☐ compilé
   - ☐ lu, écrit, parlé
   - ☐ interprété

2. Le code suivant :

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

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

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

4. Une *segmentation fault* est une erreur qui survient lorsque :
   - ☐ 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
   - ☐ le programme tente d'accèder à une partie de la mémoire qui ne lui est pas réservée
   - ☐ la division du programme en zones homogènes échoue

5. 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`

6. Quels calculs peut-on programmer en programmation structurée ?
   - ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine
   - ☐ certains programmes sont de vrais plats de spaghetti
   - ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée
   - ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine

7. Le code suivant :

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

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

8. Le code suivant :

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

   affichera :
   - ☐ 8 6 4 2
   - ☐ 8 2
   - ☐ 0 2 4 6 8
   - ☐ 8 6 4 2 0

9. Le code suivant :

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

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

10. 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 :

- ☐ 8
- ☐ 4
- ☐ 0
- ☐ 16

11. Le bus système sert à :

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

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

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

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

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

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

15. 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 = 0`
- ☐ `j = %d`
- ☐ `j = 4`

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

17. Le code suivant :

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

affichera :

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

18. Un bit est :

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

19. Le code suivant :

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

affichera :

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

20. 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 :

- ☐ 6
- ☐ 16
- ☐ 3
- ☐ 20

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Si cette erreur apparaît à la compilation : `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
   - ☐ un point-virgule manquant
   - ☐ un point-virgule en trop
   - ☐ une accolade en trop
   - ☐ une accolade manquante

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

3. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
   - ☐ `gcc prog.exe -Wall -o prog.c`
   - ☐ `gcc prog.c -o -Wall prog.exe`
   - ☐ `gcc -Wall prog.c -o prog.exe`
   - ☐ `gcc -Wall prog.exe -o prog.c`

4. 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
   - ☐ le programme affiche "Faux"
   - ☐ la variable x vaut 5 et la variable y vaut 3
   - ☐ la variable x vaut 3

5. L'ordonnancement par tourniquet permet :
   - ☐ d'afficher des ronds colorés à l'écran
   - ☐ d'entretenir l'illusion que les processus tournent en parallèle
   - ☐ de ne pas perdre de temps avec la commutation de contexte
   - ☐ de doubler la mémoire disponible

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

7. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
   - ☐ `kwrite TP4`
   - ☐ `yppasswd`
   - ☐ `new TP4`
   - ☐ `mkdir TP4`

8. Le code suivant :

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

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

9. 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 1 0 1 0 1 0 1`
   - ☐ `0 0 0 1 1 1`
   - ☐ `1 2 1 2 3`
   - ☐ `0 1 2 0 1 2`

10. 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 :
    - ☐ 4
    - ☐ 0
    - ☐ 16
    - ☐ 8

11. 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 ?

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

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

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

13. 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 1 2 0 1 2 3`
- ☐ `0 1 2 0 1 2`
- ☐ `0 1 2 3 0 1 2`
- ☐ `0 0 1 1 2 2 3`

14. 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
- ☐ 0 1 2 3 4
- ☐ 4 3 2 1

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

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

16. Le code suivant :

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

affichera :

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

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

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

18. Le code suivant :

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

affichera :

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

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

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

20. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

    - ☐ `#include <studio.h>`
    - ☐ `#include <stdio.h>`
    - ☐ `#include <studlib.h>`
    - ☐ `#appart <stdlib.h>`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

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

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

3. 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 :
   - ☐ 15
   - ☐ 10
   - ☐ 0
   - ☐ 6

4. Le code suivant :

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

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

5. Le code suivant :

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

   affichera :
   - ☐ 8 2
   - ☐ 8 6 4 2
   - ☐ 0 2 4 6 8
   - ☐ 8 6 4 2 0

6. 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 ****
   - ☐ le programme affiche x
   - ☐ la variable x vaut $-\frac{1}{2}$
   - ☐ la variable x vaut 16

7. Le code suivant :

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

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

8. 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
```

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

9. Une *segmentation fault* est une erreur qui survient lorsque :
   - ☐ 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'accèder à une partie de la mémoire qui ne lui est pas réservée
   - ☐ la division du programme en zones homogènes échoue
   - ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal

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

11. 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 ?

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

12. 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 :

- ☐ 0
- ☐ 8
- ☐ 4
- ☐ 16

13. 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
- ☐ des processus
- ☐ les fichiers du disque
- ☐ en temps d'accès

14. 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=1;i<=5;i=i+1)`
- ☐ `for(i=0;i<=5;i=i+1)`
- ☐ `for(i=0;i<5;i=i+1)`

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

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

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

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

17. 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 8
- ☐ 0 2 4 6

18. Le bus système sert à :

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

19. Le code suivant :

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

affichera :

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

20. Si cette erreur apparaît à la compilation :
`error: expected ';' before '}' token` que doit-on chercher dans le programme ?

- ☐ un point-virgule en trop
- ☐ une accolade manquante
- ☐ une accolade en trop
- ☐ un point-virgule manquant

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#appart <stdlib.h>`
   - ☐ `#include <studio.h>`
   - ☐ `#include <stdio.h>`
   - ☐ `#include <studlib.h>`
2. Le code suivant :

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

   affichera :
   - ☐ 0 2 4 6 8
   - ☐ 8 2
   - ☐ 8 6 4 2
   - ☐ 8 6 4 2 0
3. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `int tab[] = 5;`
   - ☐ `char tableau[5];`
   - ☐ `int toto[5];`
   - ☐ `int[] new tableau(5);`
   - ☐ `int toto[taille=5];`
4. Une *segmentation fault* est une erreur qui survient lorsque :
   - ☐ 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
   - ☐ la division du programme en zones homogènes échoue
   - ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
5. Le bus système sert à :
   - ☐ Transférer des données et intructions entre processeur et mémoire
   - ☐ transporter les processus du tourniquet au processeur
   - ☐ Écrire des données sur le dique dur
   - ☐ Arriver à l'heure en cours
6. Le code suivant :

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

   affichera :
   - ☐ `Mineur`
`Majeur`
   - ☐ `Majeur`
   - ☐ rien
   - ☐ `Mineur`
7. Le code suivant :

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

   affichera :
   - ☐ 0 1 2 3 4
   - ☐ 4 3 2 1
   - ☐ 4 3 2 1 0
   - ☐ 0 1 2 3
8. 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'analyse sémantique
   - ☐ l'analyse harmonique
   - ☐ l'analyse des entrées clavier
   - ☐ l'édition de liens
9. Un bit est :
   - ☐ l'instruction qui met fin à un programme
   - ☐ la longueur d'un mot mémoire
   - ☐ un chiffre binaire (0 ou 1)
   - ☐ un battement d'horloge processeur
10. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
    - ☐ `gcc -Wall prog.exe -o prog.c`
    - ☐ `gcc prog.exe -Wall -o prog.c`
    - ☐ `gcc -Wall prog.c -o prog.exe`
    - ☐ `gcc prog.c -o -Wall prog.exe`
11. 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 variable non déclarée
    - ☐ une faute de frappe dans un appel de fonction
12. 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 = 5`
- ☐ `j = %d`
- ☐ `j = 0`
- ☐ `j = 4`

13. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ créer un répertoire
    - ☐ ouvrir un fichier texte
    - ☐ créer un fichier texte
    - ☐ changer de répertoire courant
14. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
    - ☐ chacun son tour, après que le processus précédent a terminé
    - ☐ tous ensemble
    - ☐ en parallèle, chacun dans un registre
    - ☐ tour à tour, un petit peu à chaque fois
15. Le code suivant :

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

affichera :

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

16. Un registre du processeur est :
    - ☐ une gamme de fréquence de fonctionnement du processeur
    - ☐ une unité de calcul spécialisée de l'ordinateur
    - ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs
    - ☐ un composant qui contient la liste des fichiers du système
17. Le code suivant :

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

affichera :

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

18. Le code suivant :

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

affichera :

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

19. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :
    - ☐ des processus
    - ☐ en temps d'accès
    - ☐ certaines données de la mémoire de travail
    - ☐ les fichiers du disque
20. 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)`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Quel est l'opérateur de différence en C :
   - ☐ `!=`
   - ☐ `!`
   - ☐ `<>`
   - ☐ $\neq$
2. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
   - ☐ `gcc prog.exe -Wall -o prog.c`
   - ☐ `gcc prog.c -o -Wall prog.exe`
   - ☐ `gcc -Wall prog.exe -o prog.c`
   - ☐ `gcc -Wall prog.c -o prog.exe`
3. 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 2 4 6 8
   - ☐ 0 1 2 3 4 5 6 7
   - ☐ 0 2 4 6
4. Quels calculs peut-on programmer en programmation structurée ?
   - ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine
   - ☐ certains programmes sont de vrais plats de spaghetti
   - ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine
   - ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée
5. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
   - ☐ `int %d;`
   - ☐ `loop i;`
   - ☐ `int k;`
   - ☐ `int loop n;`
6. Le langage C est un langage
   - ☐ lu, écrit, parlé
   - ☐ compilé
   - ☐ interprété
   - ☐ composé
7. Sous unix (ou linux), la commande `cd` permet de :
   - ☐ récupérer un programme arrêté avec la commande `ab`
   - ☐ détruire un fichier
   - ☐ jouer de la musique
   - ☐ ouvir un bureau partagé (common desktop)
   - ☐ changer de répertoire courant
8. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
   - ☐ sélectionner entre deux blocs à l'aide d'une condition
   - ☐ mettre les blocs en séquence les uns à la suite des autres
   - ☐ répéter un bloc tant qu'une condition est vérifée
   - ☐ retourner un bloc
9. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `char tableau[5];`
   - ☐ `int toto[5];`
   - ☐ `int toto[taille=5];`
   - ☐ `int tab[] = 5;`
   - ☐ `int[] new tableau(5);`
10. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
    - ☐ `#define N =  3`
    - ☐ `#define N 3`
    - ☐ `#define taille = N`
    - ☐ `#define taille = 3`
11. 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 = 5`
    - ☐ `j = 0`
    - ☐ `j = %d`
    - ☐ `j = 4`
12. 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)
13. Les lignes

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

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

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

14. 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=%x et y=%y\n");`
    - ☐ `printf("x=%d et y=%d\n",x y);`
    - ☐ `printf("x=%d et y=%d\n,x,y");`
    - ☐ `printf("x=%d et y=%d\n",x,y);`
15. 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 0 1 1 2 2
- ☐ 0 1 2 0 1 2
- ☐ 0 1 0 1 0 1
- ☐ 1 2 3 1 2

16. 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
- ☐ 0
- ☐ 8
- ☐ 4

17. Le code suivant :

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

affichera :

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

18. 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 ?

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

19. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :
    - ☐ les fichiers du disque
    - ☐ certaines données de la mémoire de travail
    - ☐ en temps d'accès
    - ☐ des processus
20. 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 y vaut 5
- ☐ la variable x vaut 0
- ☐ la variable x vaut 5 et la variable y vaut 0

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
   - ☐ tous ensemble
   - ☐ tour à tour, un petit peu à chaque fois
   - ☐ en parallèle, chacun dans un registre
   - ☐ chacun son tour, après que le processus précédent a terminé

2. Un fichier source est :
   - ☐ un fichier texte qui sera traduit en instructions processeur
   - ☐ un document illisible pour les humains
   - ☐ 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

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

4. Les lignes

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

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

5. 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 1 2 0 1 2`
   - ☐ `0 1 2 3 0 1 2`
   - ☐ `0 0 1 1 2 2 3`
   - ☐ `0 1 2 0 1 2 3`

6. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
   - ☐ `#define N 3`
   - ☐ `#define N =  3`
   - ☐ `#define taille = N`
   - ☐ `#define taille = 3`

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

8. Le code suivant :

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

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

9. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `int[] new tableau(5);`
   - ☐ `char tableau[5];`
   - ☐ `int toto[5];`
   - ☐ `int toto[taille=5];`
   - ☐ `int tab[] = 5;`

10. Le code suivant :

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

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

11. Le langage C est un langage
    - ☐ lu, écrit, parlé
    - ☐ interprété
    - ☐ composé
    - ☐ compilé

12. 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 ?

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

13. Sous unix (ou linux), la commande `cd` permet de :
    - ☐ détruire un fichier
    - ☐ jouer de la musique
    - ☐ ouvir un bureau partagé (common desktop)
    - ☐ récupérer un programme arrêté avec la commande `ab`
    - ☐ changer de répertoire courant

14. Le code suivant :

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

affichera :

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

15. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
    - ☐ `#include <studlib.h>`
    - ☐ `#appart <stdlib.h>`
    - ☐ `#include <stdio.h>`
    - ☐ `#include <studio.h>`

16. Le code suivant :

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

affichera :

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

17. 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 7
- ☐ 0 2 4 6
- ☐ 0 2 4 6 8
- ☐ 0 1 2 3 4 5 6

18. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ changer de répertoire courant
    - ☐ ouvrir un fichier texte
    - ☐ créer un répertoire
    - ☐ créer un fichier texte

19. 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 harmonique
    - ☐ l'analyse sémantique
    - ☐ l'analyse des entrées clavier

20. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
    - ☐ `kwrite TP4`
    - ☐ `mkdir TP4`
    - ☐ `yppasswd`
    - ☐ `new TP4`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

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

2. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#include <studlib.h>`
   - ☐ `#include <studio.h>`
   - ☐ `#appart <stdlib.h>`
   - ☐ `#include <stdio.h>`
3. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
   - ☐ répéter un bloc tant qu'une condition est vérifée
   - ☐ retourner un bloc
   - ☐ mettre les blocs en séquence les uns à la suite des autres
   - ☐ sélectionner entre deux blocs à l'aide d'une condition
4. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
   - ☐ `new TP4`
   - ☐ `kwrite TP4`
   - ☐ `mkdir TP4`
   - ☐ `yppasswd`
5. 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 5 et la variable y vaut 0
   - ☐ la variable x vaut 0
   - ☐ la variable y vaut 5
6. 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 ?

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

7. Le code suivant :

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

affichera :

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

8. 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 :

- ☐ 16
- ☐ 4
- ☐ 8
- ☐ 0

9. Le code suivant :

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

affichera :

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

10. 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 `5`
- ☐ le terminal affiche `x = 2`
- ☐ le terminal affiche `x = 5`

11. 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 :

- ☐ 4
- ☐ 16
- ☐ 8
- ☐ 0

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

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

13. Le code suivant :

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

affichera :

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

14. L'ordonnancement par tourniquet permet :

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

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

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

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

- ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
- ☐ la division du programme en zones homogènes échoue
- ☐ 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'accèder à une partie de la mémoire qui ne lui est pas réservée

17. Le code suivant :

```
int i;
for (i = 0; i < 5; 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

18. Un fichier source est :

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

19. 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=0;i<5;i=i+1)`
- ☐ `for(i=1;i<=5;i=i+1)`
- ☐ `for(i=0;i<=5;i=i+1)`
- ☐ `for(i=1;i<5;i=i+1)`

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

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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :
   - ☐ en temps d'accès
   - ☐ certaines données de la mémoire de travail
   - ☐ des processus
   - ☐ les fichiers du disque

2. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
   - ☐ `#define N 3`
   - ☐ `#define taille = 3`
   - ☐ `#define taille = N`
   - ☐ `#define N =  3`

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

4. Le code suivant :

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

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

5. Le code suivant :

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

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

6. 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
   - ☐ une faute de frappe dans un appel de fonction
   - ☐ une variable non déclarée
   - ☐ un caractère interdit en C

7. 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 `****`
   - ☐ le programme affiche x
   - ☐ la variable x vaut 16
   - ☐ la variable x vaut $-\frac{1}{2}$

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

9. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
   - ☐ sélectionner entre deux blocs à l'aide d'une condition
   - ☐ répéter un bloc tant qu'une condition est vérifée
   - ☐ retourner un bloc
   - ☐ mettre les blocs en séquence les uns à la suite des autres

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

11. Le code suivant :

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

    affichera :
    - ☐ 8 2
    - ☐ 0 2 4 6 8
    - ☐ 8 6 4 2
    - ☐ 8 6 4 2 0

12. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ créer un fichier texte
    - ☐ ouvrir un fichier texte
    - ☐ changer de répertoire courant
    - ☐ créer un répertoire

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

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

14. 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 ?

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

15. 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 = 5`
- ☐ `j = %d`
- ☐ `j = 4`
- ☐ `j = 0`

16. Le code suivant :

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

affichera :

- ☐ 6
- ☐ 42
- ☐ 0
- ☐ 1

17. Le code suivant :

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

affichera :

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

18. Dans la commande gcc, l'option `-Wall` signifie :

- ☐ qu'il faut lancer un déboggueur
- ☐ qu'on veut changer alétoirement de fond d'écran
- ☐ que l'on veut voir tous les avertissements
- ☐ qu'il faut indenter le fichier source

19. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :

- ☐ chacun son tour, après que le processus précédent a terminé
- ☐ tous ensemble
- ☐ tour à tour, un petit peu à chaque fois
- ☐ en parallèle, chacun dans un registre

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

- ☐ `!`
- ☐ `!=`
- ☐ `<>`
- ☐ $\neq$

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `char tableau[5];`
   - ☐ `int[] new tableau(5);`
   - ☐ `int toto[taille=5];`
   - ☐ `int toto[5];`
   - ☐ `int tab[] = 5;`
2. 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 ?
   - ☐ `***** **** *** ***`
   - ☐ `** *** **** *****`
   - ☐ `**** **** **** ****`
   - ☐ `** ** ** ** ** **`
3. 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 y vaut 5
   - ☐ la variable x vaut 5 et la variable y vaut 0
   - ☐ la variable x vaut 0
4. 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
5. 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=%d et y=%d\n",x,y);`
   - ☐ `printf("x=%x et y=%y\n");`
   - ☐ `printf("x=%d et y=%d\n",x y);`
6. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
   - ☐ `gcc prog.c -o -Wall prog.exe`
   - ☐ `gcc -Wall prog.c -o prog.exe`
   - ☐ `gcc -Wall prog.exe -o prog.c`
   - ☐ `gcc prog.exe -Wall -o prog.c`
7. Dans la commande gcc, l'option `-Wall` signifie :
   - ☐ que l'on veut voir tous les avertissements
   - ☐ qu'il faut indenter le fichier source
   - ☐ qu'il faut lancer un déboggueur
   - ☐ qu'on veut changer alétoirement de fond d'écran
8. Le code suivant :

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

   affichera :
   - ☐ rien
   - ☐ `Mineur`
`Majeur`
   - ☐ `Mineur`
   - ☐ `Majeur`
9. Le code suivant :

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

   affichera :
   - ☐ 0 1 2 3 4
   - ☐ 4 3 2 1
   - ☐ 1 2 3 4
   - ☐ 4 3 2 1 0
10. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
    - ☐ `#include <studio.h>`
    - ☐ `#include <stdio.h>`
    - ☐ `#include <studlib.h>`
    - ☐ `#appart <stdlib.h>`
11. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
    - ☐ tous ensemble
    - ☐ 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

12. Le code suivant :

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

affichera :

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

13. 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  }
```

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

14. 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 1 2 0 1 2`
- ☐ `0 0 0 1 1 1`
- ☐ `0 1 0 1 0 1 0 1`
- ☐ `1 2 1 2 3`

15. Le code suivant :

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

affichera :

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

16. 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
- ☐ 8
- ☐ 4
- ☐ 16

17. 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 `5`
- ☐ le terminal affiche `x = 5`
- ☐ le terminal affiche "Faux"
- ☐ le terminal affiche `x = 2`

18. 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 2 0 1 2`
- ☐ `0 1 0 1 0 1`
- ☐ `0 0 1 1 2 2`
- ☐ `1 2 3 1 2`

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

- ☐ `<>`
- ☐ `!`
- ☐ $\neq$
- ☐ `!=`

20. Si cette erreur apparaît à la compilation :
`error: expected ';' before '}' token` que doit-on chercher dans le programme ?

- ☐ un point-virgule manquant
- ☐ un point-virgule en trop
- ☐ une accolade manquante
- ☐ une accolade en trop

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `int[] new tableau(5);`
   - ☐ `int toto[5];`
   - ☐ `char tableau[5];`
   - ☐ `int toto[taille=5];`
   - ☐ `int tab[] = 5;`
2. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
   - ☐ `yppasswd`
   - ☐ `mkdir TP4`
   - ☐ `new TP4`
   - ☐ `kwrite TP4`
3. Le code suivant :

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

   affichera :
   - ☐ 4 3 2 1
   - ☐ 4 3 2 1 0
   - ☐ 0 1 2 3
   - ☐ 0 1 2 3 4
4. L'ordonnancement par tourniquet permet :
   - ☐ de doubler la mémoire disponible
   - ☐ de ne pas perdre de temps avec la commutation de contexte
   - ☐ d'afficher des ronds colorés à l'écran
   - ☐ d'entretenir l'illusion que les processus tournent en parallèle
5. 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 :
   - ☐ 3
   - ☐ 16
   - ☐ 6
   - ☐ 20
6. 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 :
   - ☐ 4
   - ☐ 0
   - ☐ 8
   - ☐ 16
7. 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
   - ☐ 1 2 3 4
   - ☐ 4 3 2 1 0
8. Le code suivant :

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

   affichera :
   - ☐ 8 6 4 2
   - ☐ 8 6 4 2 0
   - ☐ 0 2 4 6 8
   - ☐ 8 2
9. Le code suivant :

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

   affichera :
   - ☐ rien
   - ☐ `Majeur`
   - ☐ `Mineur`
   - ☐ `Mineur`
     `Majeur`
10. 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 `x = 2`
    - ☐ le terminal affiche "Faux"
    - ☐ le terminal affiche `5`
    - ☐ le terminal affiche `x = 5`

11. Le code suivant :

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

affichera :

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

12. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#include <studio.h>`
- ☐ `#include <studlib.h>`
- ☐ `#include <stdio.h>`
- ☐ `#appart <stdlib.h>`

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

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

14. Un bit est :

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

15. 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 ?

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

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

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

17. 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 1 2 0 1 2`
- ☐ `0 0 1 1 2 2`
- ☐ `1 2 3 1 2`

18. 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  }
```

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

19. 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 harmonique
- ☐ l'analyse sémantique
- ☐ l'analyse des entrées clavier

20. Le code suivant :

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

affichera :

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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. 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=%d et y=%d\n",x,y);`
   - ☐ `printf("x=%d et y=%d\n,x,y");`
   - ☐ `printf("x=%x et y=%y\n");`
2. Le code suivant :

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

   affichera :
   - ☐ 8 6 4 2 0
   - ☐ 8 6 4 2
   - ☐ 0 2 4 6 8
   - ☐ 8 2
3. 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 = 5`
   - ☐ `j = %d`
   - ☐ `j = 4`
   - ☐ `j = 0`
4. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#include <studio.h>`
   - ☐ `#appart <stdlib.h>`
   - ☐ `#include <studlib.h>`
   - ☐ `#include <stdio.h>`
5. Un bit est :
   - ☐ la longueur d'un mot mémoire
   - ☐ un battement d'horloge processeur
   - ☐ un chiffre binaire (0 ou 1)
   - ☐ l'instruction qui met fin à un programme
6. 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 0 1 1 2 2`
   - ☐ `0 1 0 1 0 1`
   - ☐ `0 1 2 0 1 2`
   - ☐ `1 2 3 1 2`
7. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
   - ☐ `int loop n;`
   - ☐ `int %d;`
   - ☐ `loop i;`
   - ☐ `int k;`
8. Le code suivant :

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

   affichera :
   - ☐ 4 3 2 1
   - ☐ 4 3 2 1 0
   - ☐ 1 2 3 4
   - ☐ 0 1 2 3 4
9. 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 1 2 3 0 1 2`
   - ☐ `0 1 2 0 1 2 3`
   - ☐ `0 0 1 1 2 2 3`
   - ☐ `0 1 2 0 1 2`
10. Le code suivant :

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

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

11. Le code suivant :

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

affichera :

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

12. 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  }
```

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

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

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

14. 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 :

- ☐ 4
- ☐ 16
- ☐ 8
- ☐ 0

15. Le code suivant :

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

affichera :

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

16. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#include <studio.h>`
- ☐ `#include <studlib.h>`
- ☐ `#include <stdio.h>`
- ☐ `#appart <stdlib.h>`

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

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

18. Le code suivant :

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

affichera :

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

19. 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 :

- ☐ 6
- ☐ 16
- ☐ 3
- ☐ 20

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

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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. 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
- ☐ 4
- ☐ 8
- ☐ 16

2. Le code suivant :

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

affichera :

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

3. Le code suivant :

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

affichera :

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

4. Le code suivant :

```
int i;
for (i = 4; i > 0; 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

5. 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 :

- ☐ 3
- ☐ 20
- ☐ 6
- ☐ 16

6. 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  }
```

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

7. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :

- ☐ tour à tour, un petit peu à chaque fois
- ☐ chacun son tour, après que le processus précédent a terminé
- ☐ en parallèle, chacun dans un registre
- ☐ tous ensemble

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

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

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

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

10. L'ordonnancement par tourniquet permet :

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

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

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

12. 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 = 5`
- ☐ `j = %d`
- ☐ `j = 0`
- ☐ `j = 4`

13. Les lignes

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

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

14. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#include <studio.h>`
- ☐ `#include <stdio.h>`
- ☐ `#appart <stdlib.h>`
- ☐ `#include <studlib.h>`

15. Quels calculs peut-on programmer 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
- ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée

16. 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`

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

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

18. Le bus système sert à :

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

19. 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
- ☐ 0
- ☐ 8
- ☐ 4

20. Le code suivant :

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

affichera :

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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Le code suivant :

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

affichera :

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

2. Le code suivant :

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

affichera :

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

3. Un bit est :
   - ☐ la longueur d'un mot mémoire
   - ☐ un battement d'horloge processeur
   - ☐ un chiffre binaire (0 ou 1)
   - ☐ l'instruction qui met fin à un programme
4. Sous unix (ou linux), la commande `ls` permet de :
   - ☐ compiler un programme
   - ☐ afficher la liste de fichiers contenus dans un répertoire
   - ☐ voir des clips musicaux
   - ☐ afficher le contenu d'un fichier texte
5. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#appart <stdlib.h>`
   - ☐ `#include <studlib.h>`
   - ☐ `#include <stdio.h>`
   - ☐ `#include <studio.h>`
6. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
   - ☐ mettre les blocs en séquence les uns à la suite des autres
   - ☐ retourner un bloc
   - ☐ sélectionner entre deux blocs à l'aide d'une condition
   - ☐ répéter un bloc tant qu'une condition est vérifée
7. Dans la commande gcc, l'option `-Wall` signifie :
   - ☐ que l'on veut voir tous les avertissements
   - ☐ qu'on veut changer alétoirement de fond d'écran
   - ☐ qu'il faut indenter le fichier source
   - ☐ qu'il faut lancer un déboggueur
8. Le code suivant :

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

affichera :

- ☐ `Mineur`
  `Majeur`
- ☐ `Majeur`
- ☐ rien
- ☐ `Mineur`
9. Si cette erreur apparaît à la compilation :
   `Undefined symbols :"_prinft"` ou
   `référence indéfinie vers « prinft »` que doit-on chercher dans le programme ?
   - ☐ une faute de frappe dans un appel de fonction
   - ☐ un caractère interdit en C
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une variable non déclarée
10. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
    - ☐ `loop i;`
    - ☐ `int %d;`
    - ☐ `int k;`
    - ☐ `int loop n;`
11. 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 `x = 5`
    - ☐ le terminal affiche "Faux"
    - ☐ le terminal affiche `5`
    - ☐ le terminal affiche `x = 2`
12. 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 1 2 0 1 2`

☐ 0 1 2 3 0 1 2

☐ 0 1 2 0 1 2 3

☐ 0 0 1 1 2 2 3

13. 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  }
```

☐ le programme affiche "Faux"

☐ la variable y vaut 5

☐ la variable x vaut 5 et la variable y vaut 3

☐ la variable x vaut 3

14. Le bus système sert à :

☐ Écrire des données sur le dique dur

☐ transporter les processus du tourniquet au processeur

☐ Transférer des données et intructions entre processeur et mémoire

☐ Arriver à l'heure en cours

15. Le code suivant :

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

affichera :

☐ 0 1 2 3 4

☐ 4 3 2 1 0

☐ 0 1 2 3

☐ 4 3 2 1

16. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :

☐ tour à tour, un petit peu à chaque fois

☐ en parallèle, chacun dans un registre

☐ tous ensemble

☐ chacun son tour, après que le processus précédent a terminé

17. 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 1 0 1 0 1 0 1

☐ 0 1 2 0 1 2

☐ 0 0 0 1 1 1

☐ 1 2 1 2 3

18. 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 0 1 1 2 2

☐ 1 2 3 1 2

☐ 0 1 0 1 0 1

☐ 0 1 2 0 1 2

19. Le code suivant :

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

affichera :

☐ 4 3 2 1 0

☐ 4 3 2 1

☐ 1 2 3 4

☐ 0 1 2 3 4

20. Un registre du processeur est :

☐ une unité de calcul spécialisée de l'ordinateur

☐ un composant qui contient la liste des fichiers du système

☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs

☐ une gamme de fréquence de fonctionnement du processeur

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Le code suivant :

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

affichera :

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

2. 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
```

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

3. Le code suivant :

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

affichera :

- ☐ 42
- ☐ 6
- ☐ 0
- ☐ 1

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

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

5. 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 :

- ☐ 15
- ☐ 10
- ☐ 6
- ☐ 0

6. 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
- ☐ un composant qui contient la liste des fichiers du système
- ☐ une unité de calcul spécialisée de l'ordinateur

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

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

8. Le code suivant :

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

affichera :

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

9. Le langage C est un langage

- ☐ compilé
- ☐ interprété
- ☐ lu, écrit, parlé
- ☐ composé

10. Si cette erreur apparaît à la compilation :
`error: expected ';' before '}' token` que doit-on chercher dans le programme ?

- ☐ un point-virgule en trop
- ☐ une accolade manquante
- ☐ un point-virgule manquant
- ☐ une accolade en trop

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

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

12. 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
- ☐ le programme affiche "Faux"
- ☐ la variable x vaut 3
- ☐ la variable x vaut 5 et la variable y vaut 3

13. 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 2
- ☐ 8 6 4 2

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

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

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

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

17. Un fichier source est :

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

18. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#include <studio.h>`
- ☐ `#appart <stdlib.h>`
- ☐ `#include <studlib.h>`
- ☐ `#include <stdio.h>`

19. Un bit est :

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

20. Dans la commande gcc, l'option `-Wall` signifie :

- ☐ qu'il faut lancer un déboggueur
- ☐ qu'on veut changer alétoirement de fond d'écran
- ☐ qu'il faut indenter le fichier source
- ☐ que l'on veut voir tous les avertissements

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `int toto[5];`
   - ☐ `int toto[taille=5];`
   - ☐ `char tableau[5];`
   - ☐ `int tab[] = 5;`
   - ☐ `int[] new tableau(5);`
2. 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  }
```

   - ☐ la variable x vaut 16
   - ☐ le programme affiche `****`
   - ☐ le programme affiche x
   - ☐ la variable x vaut $-\frac{1}{2}$
3. Si cette erreur apparaît à la compilation : `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
   - ☐ une accolade manquante
   - ☐ une accolade en trop
   - ☐ un point-virgule en trop
   - ☐ un point-virgule manquant
4. L'ordonnancement par tourniquet permet :
   - ☐ de doubler la mémoire disponible
   - ☐ d'afficher des ronds colorés à l'écran
   - ☐ de ne pas perdre de temps avec la commutation de contexte
   - ☐ d'entretenir l'illusion que les processus tournent en parallèle
5. Le code suivant :

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

   affichera :
   - ☐ 0 2 4 6 8
   - ☐ 8 6 4 2 0
   - ☐ 8 2
   - ☐ 8 6 4 2
6. Un bit est :
   - ☐ l'instruction qui met fin à un programme
   - ☐ la longueur d'un mot mémoire
   - ☐ un battement d'horloge processeur
   - ☐ un chiffre binaire (0 ou 1)
7. Une *segmentation fault* est une erreur qui survient lorsque :
   - ☐ 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
   - ☐ le programme tente d'accèder à une partie de la mémoire qui ne lui est pas réservée
   - ☐ la division du programme en zones homogènes échoue
8. 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  }
```

   - ☐ le programme affiche "Faux"
   - ☐ la variable x vaut 3
   - ☐ la variable x vaut 5 et la variable y vaut 3
   - ☐ la variable y vaut 5
9. Le code suivant :

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

   affichera :
   - ☐ 0 2 4 6 8
   - ☐ 0 1 2 3 4 5 6
   - ☐ 0 1 2 3 4 5 6 7
   - ☐ 0 2 4 6
10. Le code suivant :

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

   affichera :
   - ☐ 4 3 2 1 0
   - ☐ 0 1 2 3 4
   - ☐ 0 1 2 3
   - ☐ 4 3 2 1
11. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
   - ☐ `int loop n;`
   - ☐ `int k;`
   - ☐ `loop i;`
   - ☐ `int %d;`
12. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
   - ☐ `#define N 3`
   - ☐ `#define N =  3`
   - ☐ `#define taille = N`
   - ☐ `#define taille = 3`

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

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

14. Un registre du processeur est :

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

15. 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 :

- ☐ 6
- ☐ 10
- ☐ 0
- ☐ 15

16. 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);`

17. 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 :

- ☐ 20
- ☐ 6
- ☐ 16
- ☐ 3

18. 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 = 4`
- ☐ `j = %d`
- ☐ `j = 5`
- ☐ `j = 0`

19. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#include <studlib.h>`
- ☐ `#include <studio.h>`
- ☐ `#appart <stdlib.h>`
- ☐ `#include <stdio.h>`

20. 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 :

- ☐ 4
- ☐ 0
- ☐ 8
- ☐ 16

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
   - ☐ `mkdir TP4`
   - ☐ `yppasswd`
   - ☐ `kwrite TP4`
   - ☐ `new TP4`
2. Un registre du processeur est :
   - ☐ une unité de calcul spécialisée de l'ordinateur
   - ☐ une gamme de fréquence de fonctionnement du processeur
   - ☐ un composant qui contient la liste des fichiers du système
   - ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs
3. Un bit est :
   - ☐ un battement d'horloge processeur
   - ☐ l'instruction qui met fin à un programme
   - ☐ la longueur d'un mot mémoire
   - ☐ un chiffre binaire (0 ou 1)
4. 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
   - ☐ une faute de frappe dans un appel de fonction
   - ☐ un caractère interdit en C
   - ☐ une variable non déclarée
5. 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)`
6. 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`
   - ☐ `1 2 3 1 2`
   - ☐ `0 1 2 0 1 2`
7. Le code suivant :

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

   affichera :
   - ☐ 0 2 4 6 8
   - ☐ 8 6 4 2
   - ☐ 8 6 4 2 0
   - ☐ 8 2
8. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `int toto[taille=5];`
   - ☐ `int toto[5];`
   - ☐ `int[] new tableau(5);`
   - ☐ `char tableau[5];`
   - ☐ `int tab[] = 5;`
9. Sous unix (ou linux), la commande `ls` permet de :
   - ☐ compiler un programme
   - ☐ afficher la liste de fichiers contenus dans un répertoire
   - ☐ afficher le contenu d'un fichier texte
   - ☐ voir des clips musicaux
10. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
    - ☐ `#include <studlib.h>`
    - ☐ `#appart <stdlib.h>`
    - ☐ `#include <stdio.h>`
    - ☐ `#include <studio.h>`
11. Le code suivant :

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

    affichera :
    - ☐ 4 3 2 1
    - ☐ 1 2 3 4
    - ☐ 0 1 2 3 4
    - ☐ 4 3 2 1 0
12. Le code suivant :

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

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

13. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :

- ☐ en parallèle, chacun dans un registre
- ☐ tour à tour, un petit peu à chaque fois
- ☐ chacun son tour, après que le processus précédent a terminé
- ☐ tous ensemble

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

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

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

16. 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 ?

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

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

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

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

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

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

20. 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 :

- ☐ 16
- ☐ 4
- ☐ 8
- ☐ 0

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ main
   - ☐ include
   - ☐ init
   - ☐ begin

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

3. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
   - ☐ tour à tour, un petit peu à chaque fois
   - ☐ chacun son tour, après que le processus précédent a terminé
   - ☐ tous ensemble
   - ☐ en parallèle, chacun dans un registre

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

5. Le code suivant :

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

   affichera :
   - ☐ 8 6 4 2 0
   - ☐ 8 2
   - ☐ 8 6 4 2
   - ☐ 0 2 4 6 8

6. Le langage C est un langage
   - ☐ interprété
   - ☐ compilé
   - ☐ composé
   - ☐ lu, écrit, parlé

7. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#include <studio.h>`
   - ☐ `#include <studlib.h>`
   - ☐ `#include <stdio.h>`
   - ☐ `#appart <stdlib.h>`

8. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :
   - ☐ des processus
   - ☐ en temps d'accès
   - ☐ certaines données de la mémoire de travail
   - ☐ les fichiers du disque

9. 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 1 2 0 1 2 3`
   - ☐ `0 1 2 0 1 2`
   - ☐ `0 1 2 3 0 1 2`
   - ☐ `0 0 1 1 2 2 3`

10. Le code suivant :

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

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

11. Si cette erreur apparaît à la compilation : `Undefined symbols :"_prinft"` ou `référence indéfinie vers « prinft »` que doit-on chercher dans le programme ?
    - ☐ une faute de frappe dans un appel de fonction
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une variable non déclarée
    - ☐ un caractère interdit en C

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

13. Si cette erreur apparaît à la compilation :
`error: expected ';' before '}' token` que doit-on chercher dans le programme ?
- ☐ un point-virgule manquant
- ☐ un point-virgule en trop
- ☐ une accolade en trop
- ☐ une accolade manquante

14. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
- ☐ `yppasswd`
- ☐ `mkdir TP4`
- ☐ `new TP4`
- ☐ `kwrite TP4`

15. Le code suivant :

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

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

16. 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 x vaut 5 et la variable y vaut 0
- ☐ la variable y vaut 5

17. 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=0;i<5;i=i+1)`
- ☐ `for(i=1;i<=5;i=i+1)`
- ☐ `for(i=1;i<5;i=i+1)`
- ☐ `for(i=0;i<=5;i=i+1)`

18. 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 :
- ☐ 8
- ☐ 0
- ☐ 16
- ☐ 4

19. Quel est l'opérateur de différence en C :
- ☐ `!`
- ☐ `!=`
- ☐ ≠
- ☐ `<>`

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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **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>`
   - ☐ `#appart <stdlib.h>`
   - ☐ `#include <studio.h>`
   - ☐ `#include <stdio.h>`
2. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse sémantique
   - ☐ analyse lexicale
   - ☐ analyse syntaxique
   - ☐ analyse harmonique
3. 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  }
```

   - ☐ le programme affiche "Faux"
   - ☐ la variable x vaut 5 et la variable y vaut 3
   - ☐ la variable y vaut 5
   - ☐ la variable x vaut 3
4. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `int tab[] = 5;`
   - ☐ `int toto[5];`
   - ☐ `int[] new tableau(5);`
   - ☐ `int toto[taille=5];`
   - ☐ `char tableau[5];`
5. Le code suivant :

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

   affichera :
   - ☐ `Mineur`
`Majeur`
   - ☐ rien
   - ☐ `Majeur`
   - ☐ `Mineur`
6. 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 1 2 3 0 1 2`
   - ☐ `0 0 1 1 2 2 3`
   - ☐ `0 1 2 0 1 2 3`
   - ☐ `0 1 2 0 1 2`
7. Le code suivant :

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

   affichera :
   - ☐ `Mineur`
`Majeur`
   - ☐ `Mineur`
   - ☐ rien
   - ☐ `Majeur`
8. 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 :
   - ☐ 8
   - ☐ 0
   - ☐ 16
   - ☐ 4
9. Le code suivant :

```
int i;
for (i = 4; i >= 0; 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
10. 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 `5`
- ☐ le terminal affiche `x = 5`
- ☐ le terminal affiche `x = 2`
- ☐ le terminal affiche "Faux"

11. 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);`
12. 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 2 0 1 2`
    - ☐ `0 0 1 1 2 2`
    - ☐ `0 1 0 1 0 1`
    - ☐ `1 2 3 1 2`
13. Un programme en langage C doit comporter une et une seule définition de la fonction :
    - ☐ begin
    - ☐ main
    - ☐ include
    - ☐ init
14. Dans la commande gcc, l'option `-Wall` signifie :
    - ☐ que l'on veut voir tous les avertissements
    - ☐ qu'il faut indenter le fichier source
    - ☐ qu'on veut changer alétoirement de fond d'écran
    - ☐ qu'il faut lancer un déboggueur
15. Un bit est :
    - ☐ la longueur d'un mot mémoire
    - ☐ l'instruction qui met fin à un programme
    - ☐ un chiffre binaire (0 ou 1)
    - ☐ un battement d'horloge processeur
16. Le code suivant :

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

    affichera :
    - ☐ 0 1 2 3 4
    - ☐ 4 3 2 1
    - ☐ 4 3 2 1 0
    - ☐ 1 2 3 4
17. Le code suivant :

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

    affichera :
    - ☐ 4 3 2 1 0
    - ☐ 4 3 2 1
    - ☐ 0 1 2 3
    - ☐ 0 1 2 3 4
18. 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 ?
    - ☐ `** *** **** *****`
    - ☐ `**** **** **** ****`
    - ☐ `***** **** *** ***`
    - ☐ `** ** ** ** ** **`
19. Le code suivant :

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

    affichera :
    - ☐ `Mineur`
    - ☐ `Mineur`
`Majeur`
    - ☐ rien
    - ☐ `Majeur`
20. Le bus système sert à :
    - ☐ Transférer des données et intructions entre processeur et mémoire
    - ☐ transporter les processus du tourniquet au processeur
    - ☐ Arriver à l'heure en cours
    - ☐ Écrire des données sur le dique dur

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `int tab[] = 5;`
   - ☐ `char tableau[5];`
   - ☐ `int toto[5];`
   - ☐ `int toto[taille=5];`
   - ☐ `int[] new tableau(5);`
2. 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 `5`
   - ☐ le terminal affiche `x = 2`
   - ☐ le terminal affiche "Faux"
   - ☐ le terminal affiche `x = 5`
3. Le code suivant :

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

   affichera :
   - ☐ rien
   - ☐ `Mineur`
   - ☐ `Majeur`
   - ☐ `Mineur`
     `Majeur`
4. 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 ?
   - ☐ `**** **** **** ****`
   - ☐ `***** **** *** ***`
   - ☐ `** *** **** *****`
   - ☐ `** ** ** ** ** **`
5. 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
   - ☐ 4
   - ☐ 8
6. Les lignes

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

   - ☐ ne comportent aucune erreur
   - ☐ comportent une erreur qui sera détectée au cours de l'analyse syntaxique
   - ☐ comportent une erreur qui ne sera pas détectée
   - ☐ comportent une erreur qui sera détectée au cours de l'édition de lien
7. 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=%x et y=%y\n");`
   - ☐ `printf("x=%d et y=%d\n",x y);`
   - ☐ `printf("x=%d et y=%d\n",x,y);`
   - ☐ `printf("x=%d et y=%d\n,x,y");`
8. Le bus système sert à :
   - ☐ Transférer des données et intructions entre processeur et mémoire
   - ☐ Écrire des données sur le dique dur
   - ☐ Arriver à l'heure en cours
   - ☐ transporter les processus du tourniquet au processeur
9. Sous unix (ou linux), la commande `cd` permet de :
   - ☐ changer de répertoire courant
   - ☐ récupérer un programme arrêté avec la commande `ab`
   - ☐ détruire un fichier
   - ☐ jouer de la musique
   - ☐ ouvir un bureau partagé (common desktop)
10. Dans la commande gcc, l'option `-Wall` signifie :
    - ☐ qu'on veut changer alétoirement de fond d'écran
    - ☐ qu'il faut indenter le fichier source
    - ☐ qu'il faut lancer un déboggueur
    - ☐ que l'on veut voir tous les avertissements

11. Le code suivant :

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

affichera :

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

12. Le code suivant :

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

affichera :

- ☐ 0
- ☐ 1
- ☐ 42
- ☐ 6

13. L'ordonnancement par tourniquet permet :

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

14. 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
- ☐ 4
- ☐ 0
- ☐ 8

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

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

16. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#include <studio.h>`
- ☐ `#include <stdio.h>`
- ☐ `#appart <stdlib.h>`
- ☐ `#include <studlib.h>`

17. 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=0;i<=5;i=i+1)`
- ☐ `for(i=0;i<5;i=i+1)`
- ☐ `for(i=1;i<=5;i=i+1)`
- ☐ `for(i=1;i<5;i=i+1)`

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

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

19. 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  }
```

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

20. Le langage C est un langage

- ☐ compilé
- ☐ composé
- ☐ interprété
- ☐ lu, écrit, parlé

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
   - ☐ `gcc prog.exe -Wall -o prog.c`
   - ☐ `gcc -Wall prog.exe -o prog.c`
   - ☐ `gcc -Wall prog.c -o prog.exe`
   - ☐ `gcc prog.c -o -Wall prog.exe`
2. 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 :
   - ☐ 20
   - ☐ 6
   - ☐ 16
   - ☐ 3
3. Sous unix (ou linux), la commande `ls` permet de :
   - ☐ voir des clips musicaux
   - ☐ afficher la liste de fichiers contenus dans un répertoire
   - ☐ afficher le contenu d'un fichier texte
   - ☐ compiler un programme
4. 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
5. 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 ?
   - ☐ `1 2 1 2 3`
   - ☐ `0 0 0 1 1 1`
   - ☐ `0 1 2 0 1 2`
   - ☐ `0 1 0 1 0 1 0 1`
6. Le langage C est un langage
   - ☐ composé
   - ☐ interprété
   - ☐ compilé
   - ☐ lu, écrit, parlé
7. 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");`
8. 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 :
   - ☐ 15
   - ☐ 0
   - ☐ 6
   - ☐ 10
9. Un bit est :
   - ☐ la longueur d'un mot mémoire
   - ☐ l'instruction qui met fin à un programme
   - ☐ un battement d'horloge processeur
   - ☐ un chiffre binaire (0 ou 1)
10. 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 `x = 5`
    - ☐ le terminal affiche `x = 2`
    - ☐ le terminal affiche `5`
    - ☐ le terminal affiche "Faux"
11. Le code suivant :

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

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

12. Le code suivant :

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

affichera :

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

13. 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  }
```

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

14. Le code suivant :

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

affichera :

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

15. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#appart <stdlib.h>`
- ☐ `#include <studio.h>`
- ☐ `#include <stdio.h>`
- ☐ `#include <studlib.h>`

16. 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 :

- ☐ 0
- ☐ 4
- ☐ 8
- ☐ 16

17. Le code suivant :

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

affichera :

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

18. 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  }
```

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

19. 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 ?

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

20. Le bus système sert à :

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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 points par réponse juste (unique) ; −0,5 points par réponse fausse. Durée : 20 minutes.

1. Le code suivant :

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

affichera :
- ☐ 6
- ☐ 42
- ☐ 1
- ☐ 0

2. Le code suivant :

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

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

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

4. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
- ☐ `loop i;`
- ☐ `int %d;`
- ☐ `int loop n;`
- ☐ `int k;`

5. Le code suivant :

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

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

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

7. 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 = 4`
- ☐ `j = 0`
- ☐ `j = %d`
- ☐ `j = 5`

8. 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=0;i<5;i=i+1)`
- ☐ `for(i=1;i<=5;i=i+1)`
- ☐ `for(i=0;i<=5;i=i+1)`
- ☐ `for(i=1;i<5;i=i+1)`

9. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
- ☐ chacun son tour, après que le processus précédent a terminé
- ☐ tous ensemble
- ☐ tour à tour, un petit peu à chaque fois
- ☐ en parallèle, chacun dans un registre

10. Le code suivant :

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

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

11. 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  }
```

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

12. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
    - ☐ `gcc prog.exe -Wall -o prog.c`
    - ☐ `gcc -Wall prog.c -o prog.exe`
    - ☐ `gcc -Wall prog.exe -o prog.c`
    - ☐ `gcc prog.c -o -Wall prog.exe`
13. Quel est l'opérateur de différence en C :
    - ☐ `<>`
    - ☐ `!=`
    - ☐ $\neq$
    - ☐ `!`
14. Si cette erreur apparaît à la compilation : `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
    - ☐ une accolade en trop
    - ☐ une accolade manquante
    - ☐ un point-virgule manquant
    - ☐ un point-virgule en trop
15. Dans la commande gcc, l'option `-Wall` signifie :
    - ☐ qu'il faut lancer un déboggueur
    - ☐ qu'il faut indenter le fichier source
    - ☐ que l'on veut voir tous les avertissements
    - ☐ qu'on veut changer alétoirement de fond d'écran
16. Sous unix (ou linux), la commande `cd` permet de :
    - ☐ changer de répertoire courant
    - ☐ récupérer un programme arrêté avec la commande `ab`
    - ☐ jouer de la musique
    - ☐ détruire un fichier
    - ☐ ouvir un bureau partagé (common desktop)
17. 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 :

- ☐ 20
- ☐ 3
- ☐ 6
- ☐ 16

18. Sous unix (ou linux), la commande `ls` permet de :
    - ☐ afficher le contenu d'un fichier texte
    - ☐ afficher la liste de fichiers contenus dans un répertoire
    - ☐ compiler un programme
    - ☐ voir des clips musicaux
19. 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 sémantique
    - ☐ l'analyse harmonique
    - ☐ l'analyse des entrées clavier
20. Le langage C est un langage
    - ☐ lu, écrit, parlé
    - ☐ composé
    - ☐ compilé
    - ☐ interprété

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

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

2. Quels calculs peut-on programmer en programmation structurée ?
   - ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine
   - ☐ certains programmes sont de vrais plats de spaghetti
   - ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée
   - ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine
3. Une *segmentation fault* est une erreur qui survient lorsque :
   - ☐ 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
   - ☐ le programme tente d'accèder à une partie de la mémoire qui ne lui est pas réservée
   - ☐ la division du programme en zones homogènes échoue
4. Le code suivant :

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

affichera :

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

5. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#include <stdio.h>`
   - ☐ `#appart <stdlib.h>`
   - ☐ `#include <studio.h>`
   - ☐ `#include <studlib.h>`
6. 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 :

- ☐ 0
- ☐ 8
- ☐ 4
- ☐ 16

7. Un bit est :
   - ☐ un chiffre binaire (0 ou 1)
   - ☐ la longueur d'un mot mémoire
   - ☐ un battement d'horloge processeur
   - ☐ l'instruction qui met fin à un programme
8. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ main
   - ☐ include
   - ☐ init
   - ☐ begin
9. Le code suivant :

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

affichera :

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

10. 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 7
- ☐ 0 2 4 6 8
- ☐ 0 1 2 3 4 5 6
- ☐ 0 2 4 6

11. Sous unix (ou linux), la commande `ls` permet de :
    - ☐ voir des clips musicaux
    - ☐ afficher la liste de fichiers contenus dans un répertoire
    - ☐ afficher le contenu d'un fichier texte
    - ☐ compiler un programme

12. Le code suivant :

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

affichera :

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

13. 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

14. Quel est l'opérateur de différence en C :
    - ☐ `!`
    - ☐ `!=`
    - ☐ `<>`
    - ☐ $\neq$
15. Le langage C est un langage
    - ☐ interprété
    - ☐ compilé
    - ☐ lu, écrit, parlé
    - ☐ composé
16. Sous unix (ou linux), la commande `cd` permet de :
    - ☐ détruire un fichier
    - ☐ jouer de la musique
    - ☐ récupérer un programme arrêté avec la commande `ab`
    - ☐ ouvir un bureau partagé (common desktop)
    - ☐ changer de répertoire courant
17. 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 ?
- ☐ `1 2 3 1 2`
- ☐ `0 0 1 1 2 2`
- ☐ `0 1 2 0 1 2`
- ☐ `0 1 0 1 0 1`

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=%d et y=%d\n",x,y);`
    - ☐ `printf("x=%d et y=%d\n",x y);`
    - ☐ `printf("x=%x et y=%y\n");`
19. Si cette erreur apparaît à la compilation :
    `Undefined symbols :"_prinft"` ou
    `référence indéfinie vers` « `prinft` » que doit-on chercher dans le programme ?
    - ☐ un caractère interdit en C
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une faute de frappe dans un appel de fonction
    - ☐ une variable non déclarée
20. Le code suivant :

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

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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. 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

☐ 15

☐ 10

☐ 6

2. 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 8

☐ 0 2 4 6

3. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

☐ `#include <stdio.h>`

☐ `#include <studio.h>`

☐ `#appart <stdlib.h>`

☐ `#include <studlib.h>`

4. Un registre du processeur est :

☐ une gamme de fréquence de fonctionnement du processeur

☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs

☐ un composant qui contient la liste des fichiers du système

☐ une unité de calcul spécialisée de l'ordinateur

5. 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

6. Un bit est :

☐ l'instruction qui met fin à un programme

☐ la longueur d'un mot mémoire

☐ un battement d'horloge processeur

☐ un chiffre binaire (0 ou 1)

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

☐ `kwrite TP4`

☐ `new TP4`

☐ `yppasswd`

☐ `mkdir TP4`

8. 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 :

☐ 8

☐ 16

☐ 4

☐ 0

9. 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 0 1 1 2 2`

☐ `0 1 0 1 0 1`

☐ `1 2 3 1 2`

☐ `0 1 2 0 1 2`

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

☐ `!=`

☐ `!`

☐ `<>`

☐ $\neq$

11. 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 :

- ☐ 6
- ☐ 16
- ☐ 20
- ☐ 3

12. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ créer un fichier texte
    - ☐ créer un répertoire
    - ☐ ouvrir un fichier texte
    - ☐ changer de répertoire courant
13. Les lignes

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

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

14. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
    - ☐ `gcc -Wall prog.exe -o prog.c`
    - ☐ `gcc prog.c -o -Wall prog.exe`
    - ☐ `gcc -Wall prog.c -o prog.exe`
    - ☐ `gcc prog.exe -Wall -o prog.c`
15. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :
    - ☐ en temps d'accès
    - ☐ les fichiers du disque
    - ☐ certaines données de la mémoire de travail
    - ☐ des processus
16. Le code suivant :

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

affichera :

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

17. Sous unix (ou linux), la commande `ls` permet de :
    - ☐ voir des clips musicaux
    - ☐ afficher le contenu d'un fichier texte
    - ☐ afficher la liste de fichiers contenus dans un répertoire
    - ☐ compiler un programme
18. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
    - ☐ `#define taille = 3`
    - ☐ `#define N 3`
    - ☐ `#define N =  3`
    - ☐ `#define taille = N`
19. L'ordonnancement par tourniquet permet :
    - ☐ d'entretenir l'illusion que les processus tournent en parallèle
    - ☐ de ne pas perdre de temps avec la commutation de contexte
    - ☐ de doubler la mémoire disponible
    - ☐ d'afficher des ronds colorés à l'écran
20. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
    - ☐ `int loop n;`
    - ☐ `int k;`
    - ☐ `loop i;`
    - ☐ `int %d;`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Sous unix (ou linux), la commande `ls` permet de :
   - ☐ compiler un programme
   - ☐ afficher le contenu d'un fichier texte
   - ☐ afficher la liste de fichiers contenus dans un répertoire
   - ☐ voir des clips musicaux
2. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#include <studlib.h>`
   - ☐ `#include <stdio.h>`
   - ☐ `#include <studio.h>`
   - ☐ `#appart <stdlib.h>`
3. Quels calculs peut-on programmer en programmation structurée ?
   - ☐ certains programmes sont de vrais plats de spaghetti
   - ☐ 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
   - ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée
4. 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  }
```

   - ☐ la variable x vaut 0
   - ☐ le programme affiche "Faux"
   - ☐ la variable y vaut 5
   - ☐ la variable x vaut 5 et la variable y vaut 0
5. 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 = 4`
   - ☐ `j = %d`
   - ☐ `j = 5`
   - ☐ `j = 0`
6. Le code suivant :

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

   affichera :
   - ☐ rien
   - ☐ `Mineur`
`Majeur`
   - ☐ `Mineur`
   - ☐ `Majeur`
7. 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
```

   - ☐ la case mémoire 8 contiendra 16
   - ☐ le terminal affiche 8
   - ☐ la case mémoire 8 contiendra 0
   - ☐ le bus explose
8. 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 1 2 0 1 2`
   - ☐ `0 1 2 3 0 1 2`
   - ☐ `0 1 2 0 1 2 3`
   - ☐ `0 0 1 1 2 2 3`
9. 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);`

10. Le code suivant :

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

affichera :

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

11. Le code suivant :

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

affichera :

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

12. Le bus système sert à :

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

13. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#include <studlib.h>`
- ☐ `#include <stdio.h>`
- ☐ `#appart <stdlib.h>`
- ☐ `#include <studio.h>`

14. 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
- ☐ 0 1 0 1 0 1 0 1
- ☐ 1 2 1 2 3
- ☐ 0 1 2 0 1 2

15. Le code suivant :

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

affichera :

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

16. Un registre du processeur est :

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

17. Un bit est :

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

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

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

19. Le code suivant :

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

affichera :

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

20. 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 :

- ☐ 6
- ☐ 16
- ☐ 3
- ☐ 20

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

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

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

3. 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
   - ☐ 4
   - ☐ 8
   - ☐ 16

4. 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
   - ☐ tous ensemble
   - ☐ tour à tour, un petit peu à chaque fois

5. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `char tableau[5];`
   - ☐ `int toto[5];`
   - ☐ `int toto[taille=5];`
   - ☐ `int tab[] = 5;`
   - ☐ `int[] new tableau(5);`

6. 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 ?
   - ☐ `***** **** *** ***`
   - ☐ `**** **** **** ****`
   - ☐ `** *** **** *****`
   - ☐ `** ** ** ** ** **`

7. Le code suivant :

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

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

8. Les lignes

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

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

9. 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 = 4`
   - ☐ `j = 0`
   - ☐ `j = %d`
   - ☐ `j = 5`

10. 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>`

11. Quels calculs peut-on programmer en programmation structurée ?
    - ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine
    - ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée
    - ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine
    - ☐ certains programmes sont de vrais plats de spaghetti
12. Sous unix (ou linux), la commande `ls` permet de :
    - ☐ compiler un programme
    - ☐ voir des clips musicaux
    - ☐ afficher la liste de fichiers contenus dans un répertoire
    - ☐ afficher le contenu d'un fichier texte
13. Le code suivant :

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

    affichera :
    - ☐ `Mineur`
    - ☐ `Majeur`
    - ☐ rien
    - ☐ `Mineur`
      `Majeur`
14. L'ordonnancement par tourniquet permet :
    - ☐ d'entretenir l'illusion que les processus tournent en parallèle
    - ☐ de ne pas perdre de temps avec la commutation de contexte
    - ☐ d'afficher des ronds colorés à l'écran
    - ☐ de doubler la mémoire disponible
15. Le code suivant :

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

    affichera :
    - ☐ rien
    - ☐ `Majeur`
    - ☐ `Mineur`
      `Majeur`
    - ☐ `Mineur`
16. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ créer un répertoire
    - ☐ ouvrir un fichier texte
    - ☐ changer de répertoire courant
    - ☐ créer un fichier texte
17. 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
    - ☐ des processus
    - ☐ en temps d'accès
    - ☐ les fichiers du disque
18. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
    - ☐ analyse harmonique
    - ☐ analyse syntaxique
    - ☐ analyse lexicale
    - ☐ analyse sémantique
19. 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 `5`
    - ☐ le terminal affiche `x = 5`
    - ☐ le terminal affiche `x = 2`
20. Le code suivant :

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

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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. 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 = 0`
- ☐ `j = 5`
- ☐ `j = %d`

2. 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
- ☐ 6
- ☐ 15
- ☐ 10

3. Le code suivant :

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

affichera :

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

4. Le code suivant :

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

affichera :

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

5. 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
   - ☐ certains programmes sont de vrais plats de spaghetti
   - ☐ 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

6. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
   - ☐ `#define taille = N`
   - ☐ `#define taille = 3`
   - ☐ `#define N =  3`
   - ☐ `#define N 3`

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

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

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

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

11. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
    - ☐ `gcc prog.exe -Wall -o prog.c`
    - ☐ `gcc prog.c -o -Wall prog.exe`
    - ☐ `gcc -Wall prog.exe -o prog.c`
    - ☐ `gcc -Wall prog.c -o prog.exe`
12. Le code suivant :

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

    affichera :
    - ☐ 1 2 3 4
    - ☐ 4 3 2 1 0
    - ☐ 4 3 2 1
    - ☐ 0 1 2 3 4
13. Sous unix (ou linux), la commande `cd` permet de :
    - ☐ ouvir un bureau partagé (common desktop)
    - ☐ détruire un fichier
    - ☐ jouer de la musique
    - ☐ récupérer un programme arrêté avec la commande `ab`
    - ☐ changer de répertoire courant
14. L'ordonnancement par tourniquet permet :
    - ☐ d'entretenir l'illusion que les processus tournent en parallèle
    - ☐ de doubler la mémoire disponible
    - ☐ de ne pas perdre de temps avec la commutation de contexte
    - ☐ d'afficher des ronds colorés à l'écran
15. 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 :
    - ☐ 8
    - ☐ 16
    - ☐ 0
    - ☐ 4
16. Si cette erreur apparaît à la compilation :
    `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
    - ☐ un point-virgule en trop
    - ☐ une accolade manquante
    - ☐ un point-virgule manquant
    - ☐ une accolade en trop
17. 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 1 2 3 0 1 2`
    - ☐ `0 1 2 0 1 2`
    - ☐ `0 0 1 1 2 2 3`
    - ☐ `0 1 2 0 1 2 3`
18. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :
    - ☐ des processus
    - ☐ certaines données de la mémoire de travail
    - ☐ en temps d'accès
    - ☐ les fichiers du disque
19. 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  }
```

    - ☐ la variable x vaut 16
    - ☐ la variable x vaut $-\frac{1}{2}$
    - ☐ le programme affiche x
    - ☐ le programme affiche `****`
20. 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 harmonique
    - ☐ l'analyse sémantique
    - ☐ l'analyse des entrées clavier

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse syntaxique
   - ☐ analyse lexicale
   - ☐ analyse sémantique
   - ☐ analyse harmonique

2. L'ordonnancement par tourniquet permet :
   - ☐ d'afficher des ronds colorés à l'écran
   - ☐ de ne pas perdre de temps avec la commutation de contexte
   - ☐ d'entretenir l'illusion que les processus tournent en parallèle
   - ☐ de doubler la mémoire disponible

3. Le code suivant :

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

   affichera :
   - ☐ 8 2
   - ☐ 8 6 4 2 0
   - ☐ 0 2 4 6 8
   - ☐ 8 6 4 2

4. 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 1 2 0 1 2`
   - ☐ `0 1 2 0 1 2 3`
   - ☐ `0 1 2 3 0 1 2`
   - ☐ `0 0 1 1 2 2 3`

5. Le code suivant :

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

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

6. Si cette erreur apparaît à la compilation :
   `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
   - ☐ un point-virgule en trop
   - ☐ une accolade en trop
   - ☐ une accolade manquante
   - ☐ un point-virgule manquant

7. 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 16
   - ☐ le programme affiche x
   - ☐ la variable x vaut $-\frac{1}{2}$

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

9. Quel est l'opérateur de différence en C :
   - ☐ `<>`
   - ☐ `!=`
   - ☐ `!`
   - ☐ $\neq$

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

11. Le code suivant :

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

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

12. 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 :

- ☐ 6
- ☐ 16
- ☐ 20
- ☐ 3

13. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
    - ☐ `#include <studlib.h>`
    - ☐ `#include <stdio.h>`
    - ☐ `#include <studio.h>`
    - ☐ `#appart <stdlib.h>`
14. 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`
- ☐ `0 1 2 0 1 2`
- ☐ `0 1 0 1 0 1 0 1`
- ☐ `1 2 1 2 3`

15. 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=%d et y=%d\n,x,y");`
    - ☐ `printf("x=%x et y=%y\n");`
    - ☐ `printf("x=%d et y=%d\n",x y);`
16. 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 ?

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

17. Un registre du processeur est :
    - ☐ un composant qui contient la liste des fichiers du système
    - ☐ une gamme de fréquence de fonctionnement du processeur
    - ☐ une unité de calcul spécialisée de l'ordinateur
    - ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs
18. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
    - ☐ `#define taille = 3`
    - ☐ `#define N =  3`
    - ☐ `#define N 3`
    - ☐ `#define taille = N`
19. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
    - ☐ `int k;`
    - ☐ `int %d;`
    - ☐ `loop i;`
    - ☐ `int loop n;`
20. Un fichier source est :
    - ☐ un document qui doit être protégé
    - ☐ un document de référence du système
    - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
    - ☐ un document illisible pour les humains
    - ☐ un fichier texte qui sera traduit en instructions processeur

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. 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 :

- ☐ 6
- ☐ 20
- ☐ 3
- ☐ 16

2. Le code suivant :

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

affichera :

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

3. Le langage C est un langage
   - ☐ lu, écrit, parlé
   - ☐ interprété
   - ☐ composé
   - ☐ compilé

4. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
   - ☐ `gcc prog.exe -Wall -o prog.c`
   - ☐ `gcc -Wall prog.exe -o prog.c`
   - ☐ `gcc prog.c -o -Wall prog.exe`
   - ☐ `gcc -Wall prog.c -o prog.exe`

5. Le code suivant :

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

affichera :

- ☐ 42
- ☐ 6
- ☐ 0
- ☐ 1

6. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse harmonique
   - ☐ analyse lexicale
   - ☐ analyse sémantique
   - ☐ analyse syntaxique

7. Sous unix (ou linux), la commande `ls` permet de :
   - ☐ afficher le contenu d'un fichier texte
   - ☐ afficher la liste de fichiers contenus dans un répertoire
   - ☐ voir des clips musicaux
   - ☐ compiler un programme

8. 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 ?

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

9. Quels calculs peut-on programmer 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
   - ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée

10. Le code suivant :

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

affichera :

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

11. 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 2 4 6
- ☐ 0 1 2 3 4 5 6 7
- ☐ 0 2 4 6 8

12. Quel est l'opérateur de différence en C :
    - ☐ `<>`
    - ☐ `!`
    - ☐ `!=`
    - ☐ $\neq$
13. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :
    - ☐ en temps d'accès
    - ☐ les fichiers du disque
    - ☐ des processus
    - ☐ certaines données de la mémoire de travail
14. 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=1;i<=5;i=i+1)`
    - ☐ `for(i=0;i<5;i=i+1)`
    - ☐ `for(i=0;i<=5;i=i+1)`
15. 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 :

- ☐ 16
- ☐ 8
- ☐ 0
- ☐ 4

16. Le code suivant :

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

affichera :

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

17. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
    - ☐ `#define N 3`
    - ☐ `#define taille = 3`
    - ☐ `#define taille = N`
    - ☐ `#define N =  3`
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=%d et y=%d\n,x,y");`
    - ☐ `printf("x=%d et y=%d\n",x y);`
    - ☐ `printf("x=%x et y=%y\n");`
19. Un bit est :
    - ☐ un battement d'horloge processeur
    - ☐ la longueur d'un mot mémoire
    - ☐ un chiffre binaire (0 ou 1)
    - ☐ l'instruction qui met fin à un programme
20. 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 `****`
- ☐ le programme affiche x
- ☐ la variable x vaut $-\frac{1}{2}$
- ☐ la variable x vaut 16

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Le langage C est un langage
   - ☐ interprété
   - ☐ lu, écrit, parlé
   - ☐ compilé
   - ☐ composé

2. Le code suivant :

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

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

3. 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  }
```

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

4. Le code suivant :

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

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

5. Le code suivant :

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

   affichera :
   - ☐ 42
   - ☐ 0
   - ☐ 1
   - ☐ 6

6. 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=1;i<5;i=i+1)`
   - ☐ `for(i=0;i<5;i=i+1)`

7. 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 = 4`
   - ☐ `j = %d`
   - ☐ `j = 0`
   - ☐ `j = 5`

8. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
   - ☐ `#define taille = N`
   - ☐ `#define N =  3`
   - ☐ `#define N 3`
   - ☐ `#define taille = 3`

9. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#appart <stdlib.h>`
   - ☐ `#include <stdio.h>`
   - ☐ `#include <studlib.h>`
   - ☐ `#include <studio.h>`

10. L'ordonnancement par tourniquet permet :
    - ☐ d'entretenir l'illusion que les processus tournent en parallèle
    - ☐ de ne pas perdre de temps avec la commutation de contexte
    - ☐ de doubler la mémoire disponible
    - ☐ d'afficher des ronds colorés à l'écran

11. 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 :

- ☐ 4
- ☐ 16
- ☐ 0
- ☐ 8

12. 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

13. 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 `5`
- ☐ le terminal affiche "Faux"
- ☐ le terminal affiche `x = 2`
- ☐ le terminal affiche `x = 5`

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

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

15. Le code suivant :

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

affichera :

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

16. 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 ?

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

17. Le code suivant :

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

affichera :

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

18. Dans la commande gcc, l'option `-Wall` signifie :

- ☐ qu'on veut changer alétoirement de fond d'écran
- ☐ que l'on veut voir tous les avertissements
- ☐ qu'il faut indenter le fichier source
- ☐ qu'il faut lancer un déboggueur

19. Le code suivant :

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

affichera :

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

20. Le code suivant :

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

affichera :

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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

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

1. Sur unix (ou linux), la commande `mkdir` permet de :
   - ☐ créer un fichier texte
   - ☐ changer de répertoire courant
   - ☐ créer un répertoire
   - ☐ ouvrir un fichier texte

2. Le code suivant :

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

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

3. Le code suivant :

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

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

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

5. Le code suivant :

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

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

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

7. 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 :
   - ☐ 8
   - ☐ 4
   - ☐ 16
   - ☐ 0

8. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ main
   - ☐ begin
   - ☐ include
   - ☐ init

9. 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 1 2 0 1 2`
   - ☐ `1 2 1 2 3`
   - ☐ `0 0 0 1 1 1`
   - ☐ `0 1 0 1 0 1 0 1`

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

11. 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  }
```

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

12. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
    - ☐ `gcc prog.c -o -Wall prog.exe`
    - ☐ `gcc -Wall prog.exe -o prog.c`
    - ☐ `gcc -Wall prog.c -o prog.exe`
    - ☐ `gcc prog.exe -Wall -o prog.c`
13. Un registre du processeur est :
    - ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs
    - ☐ une unité de calcul spécialisée de l'ordinateur
    - ☐ un composant qui contient la liste des fichiers du système
    - ☐ une gamme de fréquence de fonctionnement du processeur
14. Un bit est :
    - ☐ la longueur d'un mot mémoire
    - ☐ un chiffre binaire (0 ou 1)
    - ☐ l'instruction qui met fin à un programme
    - ☐ un battement d'horloge processeur
15. 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 = %d`
- ☐ `j = 5`
- ☐ `j = 4`
- ☐ `j = 0`

16. Le code suivant :

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

affichera :

- ☐ 0
- ☐ 1
- ☐ 42
- ☐ 6

17. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
    - ☐ `#include <studlib.h>`
    - ☐ `#appart <stdlib.h>`
    - ☐ `#include <stdio.h>`
    - ☐ `#include <studio.h>`
18. Le code suivant :

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

affichera :

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

19. Les lignes

```
int i;
int x=0;
for(i=0,i<5,i=i+1)
{
  x=x+1;
}
```

- ☐ comportent une erreur qui sera détectée au cours de l'analyse syntaxique
- ☐ ne comportent aucune erreur
- ☐ comportent une erreur qui ne sera pas détectée
- ☐ comportent une erreur qui sera détectée au cours de l'édition de lien

20. Le langage C est un langage
    - ☐ lu, écrit, parlé
    - ☐ composé
    - ☐ interprété
    - ☐ compilé

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Quels calculs peut-on programmer en programmation structurée ?
   - ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine
   - ☐ certains programmes sont de vrais plats de spaghetti
   - ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine
   - ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée
2. 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
   - ☐ sélectionner entre deux blocs à l'aide d'une condition
   - ☐ répéter un bloc tant qu'une condition est vérifée
3. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
   - ☐ chacun son tour, après que le processus précédent a terminé
   - ☐ tour à tour, un petit peu à chaque fois
   - ☐ tous ensemble
   - ☐ en parallèle, chacun dans un registre
4. 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 `x = 5`
   - ☐ le terminal affiche `5`
   - ☐ le terminal affiche "Faux"
   - ☐ le terminal affiche `x = 2`
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
   - ☐ 0 1 2 3
   - ☐ 0 1 2 3 4
   - ☐ 4 3 2 1
6. 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 3 0 1 2`
   - ☐ `0 1 2 0 1 2 3`
   - ☐ `0 1 2 0 1 2`
7. Le bus système sert à :
   - ☐ Arriver à l'heure en cours
   - ☐ Écrire des données sur le dique dur
   - ☐ transporter les processus du tourniquet au processeur
   - ☐ Transférer des données et intructions entre processeur et mémoire
8. 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 1 2 0 1 2`
   - ☐ `0 0 0 1 1 1`
   - ☐ `0 1 0 1 0 1 0 1`
   - ☐ `1 2 1 2 3`
9. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
printf("Majeur\n");
```

   affichera :
   - ☐ `Majeur`
   - ☐ rien
   - ☐ `Mineur`
   - ☐ `Mineur`
     `Majeur`
10. Le code suivant :

```
int i;
for (i = 0; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :
   - ☐ 4 3 2 1
   - ☐ 0 1 2 3 4
   - ☐ 0 1 2 3
   - ☐ 4 3 2 1 0

11. L'ordonnancement par tourniquet permet :
    - ☐ de doubler la mémoire disponible
    - ☐ d'afficher des ronds colorés à l'écran
    - ☐ de ne pas perdre de temps avec la commutation de contexte
    - ☐ d'entretenir l'illusion que les processus tournent en parallèle

12. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :
    - ☐ `Mineur` `Majeur`
    - ☐ `Majeur`
    - ☐ rien
    - ☐ `Mineur`

13. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
    - ☐ analyse harmonique
    - ☐ analyse sémantique
    - ☐ analyse lexicale
    - ☐ analyse syntaxique

14. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
    - ☐ `kwrite TP4`
    - ☐ `mkdir TP4`
    - ☐ `yppasswd`
    - ☐ `new TP4`

15. 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"

16. Sous unix (ou linux), la commande `ls` permet de :
    - ☐ afficher le contenu d'un fichier texte
    - ☐ afficher la liste de fichiers contenus dans un répertoire
    - ☐ compiler un programme
    - ☐ voir des clips musicaux

17. Quel est l'opérateur de différence en C :
    - ☐ `!=`
    - ☐ $\neq$
    - ☐ `<>`
    - ☐ `!`

18. 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

19. Sous unix (ou linux), la commande `cd` permet de :
    - ☐ jouer de la musique
    - ☐ changer de répertoire courant
    - ☐ ouvir un bureau partagé (common desktop)
    - ☐ détruire un fichier
    - ☐ récupérer un programme arrêté avec la commande `ab`

20. 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 = 0`
    - ☐ `j = 4`
    - ☐ `j = %d`
    - ☐ `j = 5`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :
   - ☐ les fichiers du disque
   - ☐ des processus
   - ☐ en temps d'accès
   - ☐ certaines données de la mémoire de travail

2. 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 `x = 2`
   - ☐ le terminal affiche `x = 5`
   - ☐ le terminal affiche `5`
   - ☐ le terminal affiche "Faux"

3. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
   - ☐ `gcc -Wall prog.exe -o prog.c`
   - ☐ `gcc prog.c -o -Wall prog.exe`
   - ☐ `gcc prog.exe -Wall -o prog.c`
   - ☐ `gcc -Wall prog.c -o prog.exe`

4. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse sémantique
   - ☐ analyse lexicale
   - ☐ analyse syntaxique
   - ☐ analyse harmonique

5. Une *segmentation fault* est une erreur qui survient lorsque :
   - ☐ la division du programme en zones homogènes échoue
   - ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
   - ☐ 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'accèder à une partie de la mémoire qui ne lui est pas réservée

6. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

   affichera :
   - ☐ `Majeur`
   - ☐ `Mineur`
   - ☐ rien
   - ☐ `Mineur`
     `Majeur`

7. Un registre du processeur est :
   - ☐ une unité de calcul spécialisée de l'ordinateur
   - ☐ un composant qui contient la liste des fichiers du système
   - ☐ une gamme de fréquence de fonctionnement du processeur
   - ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs

8. 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 bus explose
   - ☐ la case mémoire 8 contiendra 16
   - ☐ le terminal affiche 8
   - ☐ la case mémoire 8 contiendra 0

9. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
   - ☐ en parallèle, chacun dans un registre
   - ☐ tous ensemble
   - ☐ chacun son tour, après que le processus précédent a terminé
   - ☐ tour à tour, un petit peu à chaque fois

10. Le bus système sert à :
    - ☐ Arriver à l'heure en cours
    - ☐ transporter les processus du tourniquet au processeur
    - ☐ Écrire des données sur le dique dur
    - ☐ Transférer des données et intructions entre processeur et mémoire

11. 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=1;i<=5;i=i+1)`
    - ☐ `for(i=0;i<=5;i=i+1)`
    - ☐ `for(i=0;i<5;i=i+1)`

12. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ changer de répertoire courant
    - ☐ créer un répertoire
    - ☐ créer un fichier texte
    - ☐ ouvrir un fichier texte

13. 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 ?

- ☐ `**** **** **** ****`
- ☐ `***** **** *** ***`
- ☐ `** ** ** ** ** **`
- ☐ `** *** **** *****`

14. 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 2 4 6 8
- ☐ 0 2 4 6
- ☐ 0 1 2 3 4 5 6 7

15. Le code suivant :

```
int i;
for (i = 4; i > 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 0 1 2 3 4
- ☐ 4 3 2 1 0
- ☐ 0 1 2 3
- ☐ 4 3 2 1

16. Si cette erreur apparaît à la compilation : `Undefined symbols :"_prinft"` ou `référence indéfinie vers` « `prinft` » que doit-on chercher dans le programme ?

- ☐ une variable non déclarée
- ☐ une directive préprocesseur `#include` manquante
- ☐ un caractère interdit en C
- ☐ une faute de frappe dans un appel de fonction

17. Le code suivant :

```
int somme = 0;
int i;
for (i = 1; i < 4; i = i + 1)
{
  somme = somme + i;
}
printf("%d", somme);
```

affichera :

- ☐ 6
- ☐ 1
- ☐ 0
- ☐ 42

18. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 8 6 4 2 0
- ☐ 8 2
- ☐ 8 6 4 2
- ☐ 0 2 4 6 8

19. Un bit est :

- ☐ la longueur d'un mot mémoire
- ☐ un battement d'horloge processeur
- ☐ un chiffre binaire (0 ou 1)
- ☐ l'instruction qui met fin à un programme

20. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Mineur`
- ☐ `Majeur`
- ☐ `Mineur`
  `Majeur`
- ☐ rien

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Si cette erreur apparaît à la compilation : `Undefined symbols :"_prinft"` ou `référence indéfinie vers « prinft »` que doit-on chercher dans le programme ?
   - ☐ une faute de frappe dans un appel de fonction
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une variable non déclarée
   - ☐ un caractère interdit en C

2. 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 ?
   - ☐ `** ** ** ** ** **`
   - ☐ `***** **** *** ***`
   - ☐ `**** **** **** ****`
   - ☐ `** *** **** *****`

3. 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

4. Sous unix (ou linux), la commande `cd` permet de :
   - ☐ ouvir un bureau partagé (common desktop)
   - ☐ détruire un fichier
   - ☐ jouer de la musique
   - ☐ changer de répertoire courant
   - ☐ récupérer un programme arrêté avec la commande `ab`

5. Le code suivant :

```
int i;
for (i = 1; i < 5; 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. 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 `x = 2`
   - ☐ le terminal affiche "Faux"
   - ☐ le terminal affiche `x = 5`
   - ☐ le terminal affiche `5`

7. 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 :
   - ☐ 8
   - ☐ 16
   - ☐ 0
   - ☐ 4

8. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ include
   - ☐ init
   - ☐ main
   - ☐ begin

9. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
   - ☐ sélectionner entre deux blocs à l'aide d'une condition
   - ☐ retourner un bloc
   - ☐ mettre les blocs en séquence les uns à la suite des autres
   - ☐ répéter un bloc tant qu'une condition est vérifée

10. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :
    - ☐ des processus
    - ☐ certaines données de la mémoire de travail
    - ☐ les fichiers du disque
    - ☐ en temps d'accès

11. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
    - ☐ `kwrite TP4`
    - ☐ `mkdir TP4`
    - ☐ `yppasswd`
    - ☐ `new TP4`

12. Le langage C est un langage
    - ☐ composé
    - ☐ interprété
    - ☐ lu, écrit, parlé
    - ☐ compilé

13. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :

- ☐ analyse harmonique
- ☐ analyse syntaxique
- ☐ analyse lexicale
- ☐ analyse sémantique

14. Quel est l'opérateur de différence en C :

- ☐ $\neq$
- ☐ `!`
- ☐ `<>`
- ☐ `!=`

15. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#include <studio.h>`
- ☐ `#appart <stdlib.h>`
- ☐ `#include <stdio.h>`
- ☐ `#include <studlib.h>`

16. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction

- ☐ `int %d;`
- ☐ `int k;`
- ☐ `loop i;`
- ☐ `int loop n;`

17. 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 :

- ☐ 4
- ☐ 8
- ☐ 0
- ☐ 16

18. 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 = 4`
- ☐ `j = %d`
- ☐ `j = 0`
- ☐ `j = 5`

19. 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
- ☐ tous ensemble
- ☐ tour à tour, un petit peu à chaque fois

20. 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  }
```

- ☐ la variable x vaut $-\frac{1}{2}$
- ☐ la variable x vaut 16
- ☐ le programme affiche x
- ☐ le programme affiche `****`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#include <stdio.h>`
   - ☐ `#appart <stdlib.h>`
   - ☐ `#include <studio.h>`
   - ☐ `#include <studlib.h>`
2. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
   - ☐ `int %d;`
   - ☐ `int loop n;`
   - ☐ `int k;`
   - ☐ `loop i;`
3. Sous unix (ou linux), la commande `ls` permet de :
   - ☐ afficher la liste de fichiers contenus dans un répertoire
   - ☐ afficher le contenu d'un fichier texte
   - ☐ voir des clips musicaux
   - ☐ compiler un programme
4. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ begin
   - ☐ init
   - ☐ include
   - ☐ main
5. 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'analyse harmonique
   - ☐ l'analyse des entrées clavier
   - ☐ l'édition de liens
   - ☐ l'analyse sémantique
6. 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 ?
   - ☐ `1 2 3 1 2`
   - ☐ `0 1 2 0 1 2`
   - ☐ `0 0 1 1 2 2`
   - ☐ `0 1 0 1 0 1`
7. Le code suivant :

```
int i;
for (i = 4; i > 0; 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
8. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
   - ☐ tour à tour, un petit peu à chaque fois
   - ☐ chacun son tour, après que le processus précédent a terminé
   - ☐ tous ensemble
   - ☐ en parallèle, chacun dans un registre
9. Si cette erreur apparaît à la compilation :
   `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
   - ☐ une accolade en trop
   - ☐ une accolade manquante
   - ☐ un point-virgule en trop
   - ☐ un point-virgule manquant
10. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
    - ☐ sélectionner entre deux blocs à l'aide d'une condition
    - ☐ retourner un bloc
    - ☐ répéter un bloc tant qu'une condition est vérifiée
    - ☐ mettre les blocs en séquence les uns à la suite des autres
11. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
    - ☐ `int[] new tableau(5);`
    - ☐ `int toto[5];`
    - ☐ `int toto[taille=5];`
    - ☐ `int tab[] = 5;`
    - ☐ `char tableau[5];`
12. 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 7
    - ☐ 0 2 4 6 8
    - ☐ 0 2 4 6
    - ☐ 0 1 2 3 4 5 6

13. 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 1 0 1 0 1 0 1
- ☐ 0 1 2 0 1 2
- ☐ 0 0 0 1 1 1
- ☐ 1 2 1 2 3

14. 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 = %d`
- ☐ `j = 0`
- ☐ `j = 5`
- ☐ `j = 4`

15. Le code suivant :

```
int i;
for (i = 1; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 0 1 2 3 4
- ☐ 1 2 3 4
- ☐ 4 3 2 1
- ☐ 4 3 2 1 0

16. Un fichier source est :

- ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
- ☐ un fichier texte qui sera traduit en instructions processeur
- ☐ un document qui doit être protégé
- ☐ un document illisible pour les humains
- ☐ un document de référence du système

17. 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  }
```

- ☐ le programme affiche "Faux"
- ☐ la variable x vaut 5 et la variable y vaut 3
- ☐ la variable x vaut 3
- ☐ la variable y vaut 5

18. Le bus système sert à :

- ☐ Transférer des données et intructions entre processeur et mémoire
- ☐ transporter les processus du tourniquet au processeur
- ☐ Écrire des données sur le dique dur
- ☐ Arriver à l'heure en cours

19. 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)`

20. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 8 2
- ☐ 8 6 4 2
- ☐ 0 2 4 6 8
- ☐ 8 6 4 2 0

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. 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)`
2. Un fichier source est :
   - ☐ un document de référence du système
   - ☐ un fichier texte qui sera traduit en instructions processeur
   - ☐ un document qui doit être protégé
   - ☐ un document illisible pour les humains
   - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
3. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse sémantique
   - ☐ analyse lexicale
   - ☐ analyse syntaxique
   - ☐ analyse harmonique
4. 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[] new tableau(5);`
   - ☐ `int tab[] = 5;`
5. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#appart <stdlib.h>`
   - ☐ `#include <studio.h>`
   - ☐ `#include <stdio.h>`
   - ☐ `#include <studlib.h>`
6. 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
```

   - ☐ la case mémoire 8 contiendra 16
   - ☐ le bus explose
   - ☐ la case mémoire 8 contiendra 0
   - ☐ le terminal affiche 8
7. 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 3 0 1 2`
   - ☐ `0 1 2 0 1 2 3`
8. 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 7
   - ☐ 0 2 4 6 8
   - ☐ 0 2 4 6
   - ☐ 0 1 2 3 4 5 6
9. Un bit est :
   - ☐ l'instruction qui met fin à un programme
   - ☐ la longueur d'un mot mémoire
   - ☐ un chiffre binaire (0 ou 1)
   - ☐ un battement d'horloge processeur
10. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ ouvrir un fichier texte
    - ☐ changer de répertoire courant
    - ☐ créer un fichier texte
    - ☐ créer un répertoire
11. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

    affichera :
    - ☐ `Mineur`
`Majeur`
    - ☐ `Mineur`
    - ☐ rien
    - ☐ `Majeur`
12. Le code suivant :

```
int i;
for (i = 1; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

    affichera :
    - ☐ 0 1 2 3 4
    - ☐ 1 2 3 4
    - ☐ 4 3 2 1 0
    - ☐ 4 3 2 1

13. Quels calculs peut-on programmer en programmation structurée ?

- ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine
- ☐ certains programmes sont de vrais plats de spaghetti
- ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine
- ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée

14. 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 = 4`
- ☐ `j = 5`
- ☐ `j = %d`
- ☐ `j = 0`

15. Le code suivant :

```
int i;
for (i = 4; i > 0; 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

16. 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 ?

- ☐ `** *** **** *****`
- ☐ `** ** ** ** ** **`
- ☐ `**** **** **** ****`
- ☐ `***** **** *** ***`

17. 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 tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
- ☐ 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

18. Un programme en langage C doit comporter une et une seule définition de la fonction :

- ☐ init
- ☐ begin
- ☐ main
- ☐ include

19. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction

- ☐ `loop i;`
- ☐ `int %d;`
- ☐ `int loop n;`
- ☐ `int k;`

20. Quel est l'opérateur de différence en C :

- ☐ `!`
- ☐ $\neq$
- ☐ `<>`
- ☐ `!=`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Si cette erreur apparaît à la compilation :
   `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
   - ☐ un point-virgule manquant
   - ☐ un point-virgule en trop
   - ☐ une accolade manquante
   - ☐ une accolade en trop
2. Quel est l'opérateur de différence en C :
   - ☐ `!`
   - ☐ `!=`
   - ☐ $\neq$
   - ☐ `<>`
3. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

   affichera :
   - ☐ `Mineur`
     `Majeur`
   - ☐ `Majeur`
   - ☐ rien
   - ☐ `Mineur`
4. 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
   - ☐ sélectionner entre deux blocs à l'aide d'une condition
   - ☐ répéter un bloc tant qu'une condition est vérifée
5. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :
   - ☐ 8 6 4 2
   - ☐ 8 6 4 2 0
   - ☐ 0 2 4 6 8
   - ☐ 8 2
6. Dans la commande gcc, l'option `-Wall` signifie :
   - ☐ qu'on veut changer alétoirement de fond d'écran
   - ☐ que l'on veut voir tous les avertissements
   - ☐ qu'il faut lancer un déboggueur
   - ☐ qu'il faut indenter le fichier source
7. Les lignes

```
int i;
int x=0;
for(i=0,i<5,i=i+1)
{
  x=x+1;
}
```

   - ☐ ne comportent aucune erreur
   - ☐ 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
8. L'ordonnancement par tourniquet permet :
   - ☐ d'afficher des ronds colorés à l'écran
   - ☐ de doubler la mémoire disponible
   - ☐ de ne pas perdre de temps avec la commutation de contexte
   - ☐ d'entretenir l'illusion que les processus tournent en parallèle
9. 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'analyse harmonique
   - ☐ l'analyse sémantique
   - ☐ l'analyse des entrées clavier
   - ☐ l'édition de liens
10. 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>`
11. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
    - ☐ `#appart <stdlib.h>`
    - ☐ `#include <studlib.h>`
    - ☐ `#include <studio.h>`
    - ☐ `#include <stdio.h>`
12. 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 `5`
    - ☐ le terminal affiche `x = 5`

13. 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 ?

- ☐ `1 2 3 1 2`
- ☐ `0 1 2 0 1 2`
- ☐ `0 1 0 1 0 1`
- ☐ `0 0 1 1 2 2`

14. Le code suivant :

```
int i;
for (i = 4; i > 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 4 3 2 1
- ☐ 0 1 2 3 4
- ☐ 4 3 2 1 0
- ☐ 0 1 2 3

15. Un registre du processeur est :

- ☐ une unité de calcul spécialisée de l'ordinateur
- ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs
- ☐ un composant qui contient la liste des fichiers du système
- ☐ une gamme de fréquence de fonctionnement du processeur

16. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction

- ☐ `int %d;`
- ☐ `int k;`
- ☐ `int loop n;`
- ☐ `loop i;`

17. Le bus système sert à :

- ☐ Transférer des données et intructions entre processeur et mémoire
- ☐ Arriver à l'heure en cours
- ☐ transporter les processus du tourniquet au processeur
- ☐ Écrire des données sur le dique dur

18. 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
- ☐ 0
- ☐ 8
- ☐ 4

19. Un programme en langage C doit comporter une et une seule définition de la fonction :

- ☐ include
- ☐ begin
- ☐ main
- ☐ init

20. 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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Le code suivant :

```
int i;
for (i = 0; i < 7; i = i + 2)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 0 2 4 6
- ☐ 0 2 4 6 8
- ☐ 0 1 2 3 4 5 6
- ☐ 0 1 2 3 4 5 6 7

2. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
printf("Majeur\n");
```

affichera :

- ☐ rien
- ☐ `Mineur`
  `Majeur`
- ☐ `Majeur`
- ☐ `Mineur`

3. 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
- ☐ une faute de frappe dans un appel de fonction
- ☐ un caractère interdit en C
- ☐ une variable non déclarée

4. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :

- ☐ chacun son tour, après que le processus précédent a terminé
- ☐ tour à tour, un petit peu à chaque fois
- ☐ en parallèle, chacun dans un registre
- ☐ tous ensemble

5. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Mineur`
- ☐ `Mineur`
  `Majeur`
- ☐ `Majeur`
- ☐ rien

6. 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
```

- ☐ la case mémoire 8 contiendra 0
- ☐ la case mémoire 8 contiendra 16
- ☐ le terminal affiche 8
- ☐ le bus explose

7. Sous unix (ou linux), la commande `ls` permet de :

- ☐ afficher la liste de fichiers contenus dans un répertoire
- ☐ voir des clips musicaux
- ☐ compiler un programme
- ☐ afficher le contenu d'un fichier texte

8. Le code suivant :

```
int i;
for (i = 0; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 0 1 2 3 4
- ☐ 4 3 2 1
- ☐ 4 3 2 1 0
- ☐ 0 1 2 3

9. Le bus système sert à :

- ☐ Arriver à l'heure en cours
- ☐ Transférer des données et intructions entre processeur et mémoire
- ☐ Écrire des données sur le dique dur
- ☐ transporter les processus du tourniquet au processeur

10. 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 = 0`
- ☐ `j = 5`
- ☐ `j = %d`
- ☐ `j = 4`

11. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
    - ☐ `new TP4`
    - ☐ `kwrite TP4`
    - ☐ `yppasswd`
    - ☐ `mkdir TP4`
12. Quels calculs peut-on programmer en programmation structurée ?
    - ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine
    - ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine
    - ☐ certains programmes sont de vrais plats de spaghetti
    - ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée
13. Dans la commande gcc, l'option `-Wall` signifie :
    - ☐ qu'on veut changer alétoirement de fond d'écran
    - ☐ qu'il faut indenter le fichier source
    - ☐ qu'il faut lancer un déboggueur
    - ☐ que l'on veut voir tous les avertissements
14. 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'analyse harmonique
    - ☐ l'édition de liens
    - ☐ l'analyse des entrées clavier
    - ☐ l'analyse sémantique
15. 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  }
```

    - ☐ la variable x vaut 16
    - ☐ la variable x vaut $-\frac{1}{2}$
    - ☐ le programme affiche `****`
    - ☐ le programme affiche x
16. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

    affichera :
    - ☐ 0 2 4 6 8
    - ☐ 8 6 4 2
    - ☐ 8 2
    - ☐ 8 6 4 2 0
17. Un programme en langage C doit comporter une et une seule définition de la fonction :
    - ☐ begin
    - ☐ init
    - ☐ main
    - ☐ include
18. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ créer un répertoire
    - ☐ changer de répertoire courant
    - ☐ créer un fichier texte
    - ☐ ouvrir un fichier texte
19. Sous unix (ou linux), la commande `cd` permet de :
    - ☐ ouvir un bureau partagé (common desktop)
    - ☐ récupérer un programme arrêté avec la commande `ab`
    - ☐ jouer de la musique
    - ☐ changer de répertoire courant
    - ☐ détruire un fichier
20. Une *segmentation fault* est une erreur qui survient lorsque :
    - ☐ 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'accèder à une partie de la mémoire qui ne lui est pas réservée
    - ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
    - ☐ la division du programme en zones homogènes échoue

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. L'ordonnancement par tourniquet permet :
   - ☐ d'entretenir l'illusion que les processus tournent en parallèle
   - ☐ de ne pas perdre de temps avec la commutation de contexte
   - ☐ de doubler la mémoire disponible
   - ☐ d'afficher des ronds colorés à l'écran
2. Sous unix (ou linux), la commande `ls` permet de :
   - ☐ afficher le contenu d'un fichier texte
   - ☐ voir des clips musicaux
   - ☐ afficher la liste de fichiers contenus dans un répertoire
   - ☐ compiler un programme
3. Si cette erreur apparaît à la compilation :
   `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
   - ☐ une accolade en trop
   - ☐ un point-virgule en trop
   - ☐ un point-virgule manquant
   - ☐ une accolade manquante
4. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#include <stdio.h>`
   - ☐ `#include <studio.h>`
   - ☐ `#include <studlib.h>`
   - ☐ `#appart <stdlib.h>`
5. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :
   - ☐ les fichiers du disque
   - ☐ en temps d'accès
   - ☐ des processus
   - ☐ certaines données de la mémoire de travail
6. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :
   - ☐ 8 2
   - ☐ 0 2 4 6 8
   - ☐ 8 6 4 2
   - ☐ 8 6 4 2 0
7. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
   - ☐ `loop i;`
   - ☐ `int k;`
   - ☐ `int loop n;`
   - ☐ `int %d;`
8. 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 :
   - ☐ 10
   - ☐ 6
   - ☐ 0
   - ☐ 15
9. 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 0
   - ☐ la case mémoire 8 contiendra 16
   - ☐ le bus explose
10. Une *segmentation fault* est une erreur qui survient lorsque :
    - ☐ 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'accèder à une partie de la mémoire qui ne lui est pas réservée
    - ☐ la division du programme en zones homogènes échoue
    - ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
11. 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 ?
    - ☐ `** ** ** ** ** **`
    - ☐ `***** **** *** ***`
    - ☐ `**** **** **** ****`
    - ☐ `** *** **** *****`
12. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
    - ☐ `#define N 3`
    - ☐ `#define N =  3`
    - ☐ `#define taille = 3`
    - ☐ `#define taille = N`

13. Le code suivant :

```
int i;
for (i = 1; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 0 1 2 3 4
- ☐ 1 2 3 4
- ☐ 4 3 2 1
- ☐ 4 3 2 1 0

14. 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'édition de lien
- ☐ ne comportent aucune erreur
- ☐ comportent une erreur qui sera détectée au cours de l'analyse syntaxique

15. Un fichier source est :

- ☐ un fichier texte qui sera traduit en instructions processeur
- ☐ un document illisible pour les humains
- ☐ 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

16. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction

- ☐ `char tableau[5];`
- ☐ `int toto[taille=5];`
- ☐ `int[] new tableau(5);`
- ☐ `int toto[5];`
- ☐ `int tab[] = 5;`

17. Le code suivant :

```
int i;
for (i = 4; i > 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 4 3 2 1 0
- ☐ 0 1 2 3
- ☐ 4 3 2 1
- ☐ 0 1 2 3 4

18. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Mineur`
`Majeur`
- ☐ `Mineur`
- ☐ `Majeur`
- ☐ rien

19. Sous unix (ou linux), la commande `cd` permet de :

- ☐ récupérer un programme arrêté avec la commande `ab`
- ☐ jouer de la musique
- ☐ ouvir un bureau partagé (common desktop)
- ☐ détruire un fichier
- ☐ changer de répertoire courant

20. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
printf("Majeur\n");
```

affichera :

- ☐ `Majeur`
- ☐ rien
- ☐ `Mineur`
`Majeur`
- ☐ `Mineur`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
   - ☐ répéter un bloc tant qu'une condition est vérifée
   - ☐ mettre les blocs en séquence les uns à la suite des autres
   - ☐ retourner un bloc
   - ☐ sélectionner entre deux blocs à l'aide d'une condition
2. Dans la commande gcc, l'option `-Wall` signifie :
   - ☐ qu'il faut lancer un déboggueur
   - ☐ que l'on veut voir tous les avertissements
   - ☐ qu'on veut changer alétoirement de fond d'écran
   - ☐ qu'il faut indenter le fichier source
3. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `int[] new tableau(5);`
   - ☐ `int toto[taille=5];`
   - ☐ `char tableau[5];`
   - ☐ `int toto[5];`
   - ☐ `int tab[] = 5;`
4. 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 1 2 3 0 1 2`
   - ☐ `0 1 2 0 1 2 3`
   - ☐ `0 0 1 1 2 2 3`
   - ☐ `0 1 2 0 1 2`
5. Le code suivant :

```
int i;
for (i = 4; i >= 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :
   - ☐ 4 3 2 1
   - ☐ 4 3 2 1 0
   - ☐ 0 1 2 3 4
   - ☐ 1 2 3 4
6. 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 2 0 1 2`
   - ☐ `0 1 0 1 0 1 0 1`
7. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
   - ☐ chacun son tour, après que le processus précédent a terminé
   - ☐ tour à tour, un petit peu à chaque fois
   - ☐ tous ensemble
   - ☐ en parallèle, chacun dans un registre
8. 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
   - ☐ 0 1 2 3 4
   - ☐ 4 3 2 1
9. Le code suivant :

```
int i;
for (i = 4; i > 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :
   - ☐ 4 3 2 1 0
   - ☐ 0 1 2 3
   - ☐ 0 1 2 3 4
   - ☐ 4 3 2 1
10. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
printf("Majeur\n");
```

    affichera :
    - ☐ rien
    - ☐ `Mineur`
    - ☐ `Majeur`
    - ☐ `Mineur`
      `Majeur`

11. 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 tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
- ☐ 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

12. 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 2 4 6
- ☐ 0 2 4 6 8
- ☐ 0 1 2 3 4 5 6 7

13. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Majeur`
- ☐ rien
- ☐ `Mineur`
- ☐ `Mineur`
  `Majeur`

14. Les lignes

```
int i;
int x=0;
for(i=0,i<5,i=i+1)
{
  x=x+1;
}
```

- ☐ comportent une erreur qui sera détectée au cours de l'édition de lien
- ☐ ne comportent aucune erreur
- ☐ comportent une erreur qui ne sera pas détectée
- ☐ comportent une erreur qui sera détectée au cours de l'analyse syntaxique

15. 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
- ☐ le programme affiche "Faux"
- ☐ la variable x vaut 5 et la variable y vaut 3
- ☐ la variable x vaut 3

16. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.

- ☐ `#define taille = N`
- ☐ `#define taille = 3`
- ☐ `#define N =  3`
- ☐ `#define N 3`

17. 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=%x et y=%y\n");`
- ☐ `printf("x=%d et y=%d\n",x y);`
- ☐ `printf("x=%d et y=%d\n",x,y);`
- ☐ `printf("x=%d et y=%d\n,x,y");`

18. Un bit est :

- ☐ la longueur d'un mot mémoire
- ☐ un chiffre binaire (0 ou 1)
- ☐ un battement d'horloge processeur
- ☐ l'instruction qui met fin à un programme

19. Quel est l'opérateur de différence en C :

- ☐ `<>`
- ☐ `!`
- ☐ $\neq$
- ☐ `!=`

20. 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

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. 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 `****`
   - ☐ le programme affiche x
   - ☐ la variable x vaut 16
   - ☐ la variable x vaut $-\frac{1}{2}$

2. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `int toto[taille=5];`
   - ☐ `char tableau[5];`
   - ☐ `int tab[] = 5;`
   - ☐ `int toto[5];`
   - ☐ `int[] new tableau(5);`

3. Un bit est :
   - ☐ l'instruction qui met fin à un programme
   - ☐ la longueur d'un mot mémoire
   - ☐ un chiffre binaire (0 ou 1)
   - ☐ un battement d'horloge processeur

4. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
   - ☐ tous ensemble
   - ☐ tour à tour, un petit peu à chaque fois
   - ☐ chacun son tour, après que le processus précédent a terminé
   - ☐ en parallèle, chacun dans un registre

5. Sous unix (ou linux), la commande `cd` permet de :
   - ☐ jouer de la musique
   - ☐ détruire un fichier
   - ☐ ouvir un bureau partagé (common desktop)
   - ☐ récupérer un programme arrêté avec la commande `ab`
   - ☐ changer de répertoire courant

6. 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
   - ☐ le bus explose
   - ☐ la case mémoire 8 contiendra 16
   - ☐ la case mémoire 8 contiendra 0

7. 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 0 1 1 2 2`
   - ☐ `0 1 2 0 1 2`
   - ☐ `0 1 0 1 0 1`
   - ☐ `1 2 3 1 2`

8. Dans la commande gcc, l'option `-Wall` signifie :
   - ☐ qu'on veut changer alétoirement de fond d'écran
   - ☐ qu'il faut indenter le fichier source
   - ☐ qu'il faut lancer un déboggueur
   - ☐ que l'on veut voir tous les avertissements

9. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

   affichera :
   - ☐ `Mineur`
   - ☐ `Mineur`
`Majeur`
   - ☐ `Majeur`
   - ☐ rien

10. Une *segmentation fault* est une erreur qui survient lorsque :
    - ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
    - ☐ 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
    - ☐ la division du programme en zones homogènes échoue

11. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
    - ☐ sélectionner entre deux blocs à l'aide d'une condition

- ☐ répéter un bloc tant qu'une condition est vérifée
- ☐ retourner un bloc
- ☐ mettre les blocs en séquence les uns à la suite des autres

12. 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 ?

- ☐ `1 2 1 2 3`
- ☐ `0 0 0 1 1 1`
- ☐ `0 1 0 1 0 1 0 1`
- ☐ `0 1 2 0 1 2`

13. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#include <studlib.h>`
- ☐ `#include <stdio.h>`
- ☐ `#include <studio.h>`
- ☐ `#appart <stdlib.h>`

14. 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=%d et y=%d\n,x,y");`
- ☐ `printf("x=%x et y=%y\n");`
- ☐ `printf("x=%d et y=%d\n",x y);`

15. Sous unix (ou linux), la commande `ls` permet de :

- ☐ compiler un programme
- ☐ afficher la liste de fichiers contenus dans un répertoire
- ☐ voir des clips musicaux
- ☐ afficher le contenu d'un fichier texte

16. 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 1 2 3 4
- ☐ 4 3 2 1 0

17. Pour compiler un programme `prog.c`, on utilise la ligne de commande :

- ☐ `gcc -Wall prog.exe -o prog.c`
- ☐ `gcc prog.c -o -Wall prog.exe`
- ☐ `gcc -Wall prog.c -o prog.exe`
- ☐ `gcc prog.exe -Wall -o prog.c`

18. 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 3`
- ☐ `0 1 2 0 1 2`
- ☐ `0 1 2 3 0 1 2`

19. Un programme en langage C doit comporter une et une seule définition de la fonction :

- ☐ main
- ☐ include
- ☐ init
- ☐ begin

20. Sur unix (ou linux), la commande `mkdir` permet de :

- ☐ ouvrir un fichier texte
- ☐ créer un fichier texte
- ☐ changer de répertoire courant
- ☐ créer un répertoire

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.

   ☐ `#define N 3`

   ☐ `#define taille = 3`

   ☐ `#define N =  3`

   ☐ `#define taille = N`

2. Le code suivant :

```
int i;
for (i = 4; i > 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :

   ☐ 4 3 2 1

   ☐ 4 3 2 1 0

   ☐ 0 1 2 3 4

   ☐ 0 1 2 3

3. 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 0 1 1 2 2

   ☐ 0 1 0 1 0 1

   ☐ 0 1 2 0 1 2

   ☐ 1 2 3 1 2

4. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

   ☐ `#include <studio.h>`

   ☐ `#include <stdio.h>`

   ☐ `#appart <stdlib.h>`

   ☐ `#include <studlib.h>`

5. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

   affichera :

   ☐ `Majeur`

   ☐ `Mineur`

   ☐ `Mineur`
   `Majeur`

   ☐ rien

6. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :

   ☐ analyse lexicale

   ☐ analyse syntaxique

   ☐ analyse harmonique

   ☐ analyse sémantique

7. Quel est l'opérateur de différence en C :

   ☐ $\neq$

   ☐ `!=`

   ☐ `!`

   ☐ `<>`

8. Si cette erreur apparaît à la compilation :
   `Undefined symbols :"_prinft"` ou
   `référence indéfinie vers « prinft »` que doit-on chercher dans le programme ?

   ☐ un caractère interdit en C

   ☐ une faute de frappe dans un appel de fonction

   ☐ une directive préprocesseur `#include` manquante

   ☐ une variable non déclarée

9. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

   ☐ `#include <stdio.h>`

   ☐ `#include <studio.h>`

   ☐ `#appart <stdlib.h>`

   ☐ `#include <studlib.h>`

10. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

    affichera :

    ☐ `Majeur`

    ☐ `Mineur`

    ☐ `Mineur`
    `Majeur`

    ☐ rien

11. 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=%d et y=%d\n,x,y");`

    ☐ `printf("x=%d et y=%d\n",x y);`

    ☐ `printf("x=%x et y=%y\n");`

12. Le code suivant :

```
int i;
for (i = 1; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 4 3 2 1 0
- ☐ 4 3 2 1
- ☐ 1 2 3 4
- ☐ 0 1 2 3 4

13. 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 :

- ☐ 8
- ☐ 0
- ☐ 4
- ☐ 16

14. Le code suivant :

```
int i;
for (i = 4; i >= 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 4 3 2 1 0
- ☐ 1 2 3 4
- ☐ 0 1 2 3 4
- ☐ 4 3 2 1

15. 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 1 2 0 1 2
- ☐ 0 0 1 1 2 2 3
- ☐ 0 1 2 0 1 2 3
- ☐ 0 1 2 3 0 1 2

16. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction

- ☐ `int toto[5];`
- ☐ `int[] new tableau(5);`
- ☐ `int tab[] = 5;`
- ☐ `char tableau[5];`
- ☐ `int toto[taille=5];`

17. Sous unix (ou linux), la commande `cd` permet de :

- ☐ jouer de la musique
- ☐ ouvir un bureau partagé (common desktop)
- ☐ détruire un fichier
- ☐ récupérer un programme arrêté avec la commande `ab`
- ☐ changer de répertoire courant

18. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :

- ☐ sélectionner entre deux blocs à l'aide d'une condition
- ☐ répéter un bloc tant qu'une condition est vérifée
- ☐ mettre les blocs en séquence les uns à la suite des autres
- ☐ retourner un bloc

19. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :

- ☐ en temps d'accès
- ☐ des processus
- ☐ les fichiers du disque
- ☐ certaines données de la mémoire de travail

20. 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=1;i<=5;i=i+1)`
- ☐ `for(i=0;i<5;i=i+1)`
- ☐ `for(i=0;i<=5;i=i+1)`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
   - ☐ retourner un bloc
   - ☐ sélectionner entre deux blocs à l'aide d'une condition
   - ☐ mettre les blocs en séquence les uns à la suite des autres
   - ☐ répéter un bloc tant qu'une condition est vérifée

2. 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  }
```

   - ☐ la variable y vaut 5
   - ☐ la variable x vaut 5 et la variable y vaut 0
   - ☐ le programme affiche "Faux"
   - ☐ la variable x vaut 0

3. 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 :
   - ☐ 15
   - ☐ 10
   - ☐ 6
   - ☐ 0

4. Sous unix (ou linux), la commande `cd` permet de :
   - ☐ récupérer un programme arrêté avec la commande `ab`
   - ☐ jouer de la musique
   - ☐ ouvir un bureau partagé (common desktop)
   - ☐ changer de répertoire courant
   - ☐ détruire un fichier

5. 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 fichier texte qui sera traduit en instructions processeur
   - ☐ un document qui doit être protégé
   - ☐ un document illisible pour les humains

6. 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 1 0 1 0 1 0 1`
   - ☐ `0 0 0 1 1 1`
   - ☐ `0 1 2 0 1 2`
   - ☐ `1 2 1 2 3`

7. 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);`

8. Si cette erreur apparaît à la compilation : `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
   - ☐ un point-virgule en trop
   - ☐ une accolade manquante
   - ☐ une accolade en trop
   - ☐ un point-virgule manquant

9. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

   affichera :
   - ☐ rien
   - ☐ `Majeur`
   - ☐ `Mineur`
   - ☐ `Mineur`
     `Majeur`

10. 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 1 2 0 1 2`
    - ☐ `0 0 1 1 2 2 3`
    - ☐ `0 1 2 3 0 1 2`
    - ☐ `0 1 2 0 1 2 3`

11. 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
```

- ☐ la case mémoire 8 contiendra 16
- ☐ la case mémoire 8 contiendra 0
- ☐ le terminal affiche 8
- ☐ le bus explose

12. 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)`
13. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
    - ☐ `gcc prog.c -o -Wall prog.exe`
    - ☐ `gcc -Wall prog.exe -o prog.c`
    - ☐ `gcc prog.exe -Wall -o prog.c`
    - ☐ `gcc -Wall prog.c -o prog.exe`
14. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Mineur`
- ☐ `Majeur`
- ☐ rien
- ☐ `Mineur`
  `Majeur`

15. Dans la commande gcc, l'option `-Wall` signifie :
    - ☐ qu'il faut indenter le fichier source
    - ☐ qu'on veut changer alétoirement de fond d'écran
    - ☐ que l'on veut voir tous les avertissements
    - ☐ qu'il faut lancer un déboggueur
16. Le code suivant :

```
int i;
for (i = 1; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 0 1 2 3 4
- ☐ 1 2 3 4
- ☐ 4 3 2 1
- ☐ 4 3 2 1 0

17. Le code suivant :

```
int i;
for (i = 0; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 0 1 2 3 4
- ☐ 0 1 2 3
- ☐ 4 3 2 1
- ☐ 4 3 2 1 0

18. Une *segmentation fault* est une erreur qui survient lorsque :
    - ☐ le programme tente d'accèder à une partie de la mémoire qui ne lui est pas réservée
    - ☐ la division du programme en zones homogènes échoue
    - ☐ 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
19. Le bus système sert à :
    - ☐ transporter les processus du tourniquet au processeur
    - ☐ Écrire des données sur le dique dur
    - ☐ Transférer des données et intructions entre processeur et mémoire
    - ☐ Arriver à l'heure en cours
20. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
    - ☐ analyse sémantique
    - ☐ analyse harmonique
    - ☐ analyse lexicale
    - ☐ analyse syntaxique

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Un fichier source est :
   - ☐ un fichier texte qui sera traduit en instructions processeur
   - ☐ un document illisible pour les humains
   - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
   - ☐ un document qui doit être protégé
   - ☐ un document de référence du système
2. 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
   - ☐ ne comportent aucune erreur
   - ☐ comportent une erreur qui sera détectée au cours de l'édition de lien
3. Si cette erreur apparaît à la compilation : `Undefined symbols :"_prinft"` ou `référence indéfinie vers` « `prinft` » que doit-on chercher dans le programme ?
   - ☐ une variable non déclarée
   - ☐ un caractère interdit en C
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une faute de frappe dans un appel de fonction
4. 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
```
   - ☐ la case mémoire 8 contiendra 16
   - ☐ le terminal affiche 8
   - ☐ le bus explose
   - ☐ la case mémoire 8 contiendra 0
5. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
   - ☐ `int %d;`
   - ☐ `int loop n;`
   - ☐ `loop i;`
   - ☐ `int k;`
6. Sur unix (ou linux), la commande `mkdir` permet de :
   - ☐ créer un fichier texte
   - ☐ changer de répertoire courant
   - ☐ créer un répertoire
   - ☐ ouvrir un fichier texte
7. 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 1 2 3 0 1 2`
   - ☐ `0 1 2 0 1 2 3`
   - ☐ `0 1 2 0 1 2`
   - ☐ `0 0 1 1 2 2 3`
8. Le langage C est un langage
   - ☐ interprété
   - ☐ compilé
   - ☐ composé
   - ☐ lu, écrit, parlé
9. 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 `5`
   - ☐ le terminal affiche "Faux"
   - ☐ le terminal affiche `x = 2`
   - ☐ le terminal affiche `x = 5`
10. Le code suivant :
```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```
    affichera :
    - ☐ `Mineur`
    - ☐ `Mineur`
`Majeur`
    - ☐ rien
    - ☐ `Majeur`
11. 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=%d et y=%d\n",x,y);`
    - ☐ `printf("x=%d et y=%d\n",x y);`
    - ☐ `printf("x=%x et y=%y\n");`

12. Un registre du processeur est :
    - ☐ une gamme de fréquence de fonctionnement du processeur
    - ☐ un composant qui contient la liste des fichiers du système
    - ☐ une unité de calcul spécialisée de l'ordinateur
    - ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs
13. Sous unix (ou linux), la commande `cd` permet de :
    - ☐ détruire un fichier
    - ☐ jouer de la musique
    - ☐ récupérer un programme arrêté avec la commande `ab`
    - ☐ changer de répertoire courant
    - ☐ ouvir un bureau partagé (common desktop)
14. 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 ?

- ☐ ***** **** *** ***
- ☐ **** **** **** ****
- ☐ ** ** ** ** ** **
- ☐ ** *** **** *****

15. 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  }
```

- ☐ le programme affiche "Faux"
- ☐ la variable x vaut 3
- ☐ la variable x vaut 5 et la variable y vaut 3
- ☐ la variable y vaut 5

16. 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
17. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
    - ☐ 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
    - ☐ retourner un bloc
18. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :
    - ☐ en temps d'accès
    - ☐ certaines données de la mémoire de travail
    - ☐ les fichiers du disque
    - ☐ des processus
19. 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
- ☐ 0 1 2 3 4
- ☐ 4 3 2 1

20. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
    - ☐ `#include <studio.h>`
    - ☐ `#include <studlib.h>`
    - ☐ `#appart <stdlib.h>`
    - ☐ `#include <stdio.h>`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. 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 :

☐ 6

☐ 10

☐ 15

☐ 0

2. Si cette erreur apparaît à la compilation : `error: expected ';' before '}' token` que doit-on chercher dans le programme ?

☐ une accolade en trop

☐ une accolade manquante

☐ un point-virgule en trop

☐ un point-virgule manquant

3. Le code suivant :

```
int i;
for (i = 1; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

☐ 4 3 2 1

☐ 1 2 3 4

☐ 4 3 2 1 0

☐ 0 1 2 3 4

4. 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");`

5. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

☐ `Majeur`

☐ rien

☐ `Mineur`
`Majeur`

☐ `Mineur`

6. 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  }
```

☐ la variable x vaut 5 et la variable y vaut 0

☐ le programme affiche "Faux"

☐ la variable y vaut 5

☐ la variable x vaut 0

7. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

☐ `Majeur`

☐ rien

☐ `Mineur`

☐ `Mineur`
`Majeur`

8. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

☐ `#include <stdio.h>`

☐ `#include <studio.h>`

☐ `#include <studlib.h>`

☐ `#appart <stdlib.h>`

9. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :

☐ analyse syntaxique

☐ analyse lexicale

☐ analyse sémantique

☐ analyse harmonique

10. 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 `5`

- ☐ le terminal affiche `x = 5`
- ☐ le terminal affiche "Faux"
- ☐ le terminal affiche `x = 2`

11. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ rien
- ☐ `Mineur`
- ☐ `Mineur`
  `Majeur`
- ☐ `Majeur`

12. 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 harmonique
- ☐ l'analyse sémantique

13. Le code suivant :

```
int i;
for (i = 4; i >= 0; 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

14. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#include <studio.h>`
- ☐ `#include <studlib.h>`
- ☐ `#include <stdio.h>`
- ☐ `#appart <stdlib.h>`

15. 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
```

- ☐ la case mémoire 8 contiendra 0
- ☐ le terminal affiche 8
- ☐ le bus explose
- ☐ la case mémoire 8 contiendra 16

16. 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 0 1 1 2 2`
- ☐ `0 1 0 1 0 1`
- ☐ `1 2 3 1 2`
- ☐ `0 1 2 0 1 2`

17. 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 1 2 0 1 2`
- ☐ `0 1 2 0 1 2 3`
- ☐ `0 1 2 3 0 1 2`
- ☐ `0 0 1 1 2 2 3`

18. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :

- ☐ en parallèle, chacun dans un registre
- ☐ tour à tour, un petit peu à chaque fois
- ☐ chacun son tour, après que le processus précédent a terminé
- ☐ tous ensemble

19. Le code suivant :

```
int i;
for (i = 0; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 4 3 2 1
- ☐ 0 1 2 3
- ☐ 4 3 2 1 0
- ☐ 0 1 2 3 4

20. Un programme en langage C doit comporter une et une seule définition de la fonction :

- ☐ begin
- ☐ init
- ☐ main
- ☐ include

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Le code suivant :

```
int somme = 0;
int i;
for (i = 1; i < 4; i = i + 1)
{
  somme = somme + i;
}
printf("%d", somme);
```

affichera :

- ☐ 6
- ☐ 0
- ☐ 42
- ☐ 1

2. 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 3
- ☐ la variable x vaut 5 et la variable y vaut 3
- ☐ le programme affiche "Faux"

3. 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
```

- ☐ la case mémoire 8 contiendra 16
- ☐ le bus explose
- ☐ le terminal affiche 8
- ☐ la case mémoire 8 contiendra 0

4. Une *segmentation fault* est une erreur qui survient lorsque :

- ☐ la division du programme en zones homogènes échoue
- ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
- ☐ 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'accèder à une partie de la mémoire qui ne lui est pas réservée

5. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#appart <stdlib.h>`
- ☐ `#include <stdio.h>`
- ☐ `#include <studlib.h>`
- ☐ `#include <studio.h>`

6. Le code suivant :

```
int i;
for (i = 0; i < 7; i = i + 2)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 0 2 4 6
- ☐ 0 1 2 3 4 5 6
- ☐ 0 2 4 6 8
- ☐ 0 1 2 3 4 5 6 7

7. Le code suivant :

```
int i;
for (i = 4; i > 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 4 3 2 1
- ☐ 0 1 2 3 4
- ☐ 4 3 2 1 0
- ☐ 0 1 2 3

8. Sur unix (ou linux), la commande `mkdir` permet de :

- ☐ ouvrir un fichier texte
- ☐ changer de répertoire courant
- ☐ créer un répertoire
- ☐ créer un fichier texte

9. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ rien
- ☐ `Mineur`
- ☐ `Mineur`
  `Majeur`
- ☐ `Majeur`

10. L'ordonnancement par tourniquet permet :
    - ☐ d'entretenir l'illusion que les processus tournent en parallèle
    - ☐ d'afficher des ronds colorés à l'écran
    - ☐ de doubler la mémoire disponible
    - ☐ de ne pas perdre de temps avec la commutation de contexte

11. Un fichier source est :
    - ☐ un document illisible pour les humains
    - ☐ un document qui doit être protégé
    - ☐ un fichier texte qui sera traduit en instructions processeur
    - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
    - ☐ un document de référence du système

12. Quels calculs peut-on programmer en programmation structurée ?
    - ☐ certains programmes sont de vrais plats de spaghetti
    - ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine
    - ☐ 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

13. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

    affichera :
    - ☐ `Majeur`
    - ☐ rien
    - ☐ `Mineur`
    - ☐ `Mineur`
      `Majeur`

14. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
    - ☐ `#define taille = N`
    - ☐ `#define N 3`
    - ☐ `#define N =  3`
    - ☐ `#define taille = 3`

15. 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 = 0`
    - ☐ `j = 4`
    - ☐ `j = %d`
    - ☐ `j = 5`

16. Un bit est :
    - ☐ un chiffre binaire (0 ou 1)
    - ☐ la longueur d'un mot mémoire
    - ☐ un battement d'horloge processeur
    - ☐ l'instruction qui met fin à un programme

17. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
    - ☐ `int k;`
    - ☐ `int loop n;`
    - ☐ `int %d;`
    - ☐ `loop i;`

18. 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 :
    - ☐ 10
    - ☐ 6
    - ☐ 0
    - ☐ 15

19. Le code suivant :

```
int age = 20;
if (age < 18)
{
   printf("Mineur\n");
}
printf("Majeur\n");
```

    affichera :
    - ☐ `Majeur`
    - ☐ `Mineur`
      `Majeur`
    - ☐ `Mineur`
    - ☐ rien

20. 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 :
    - ☐ 4
    - ☐ 0
    - ☐ 16
    - ☐ 8

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Le code suivant :

```
int i;
for (i = 0; i < 7; i = i + 2)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

☐ 0 2 4 6

☐ 0 2 4 6 8

☐ 0 1 2 3 4 5 6 7

☐ 0 1 2 3 4 5 6

2. 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

☐ une variable non déclarée

☐ un caractère interdit en C

☐ une faute de frappe dans un appel de fonction

3. 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 1 2 0 1 2 3`

☐ `0 0 1 1 2 2 3`

☐ `0 1 2 3 0 1 2`

☐ `0 1 2 0 1 2`

4. Un programme en langage C doit comporter une et une seule définition de la fonction :

☐ init

☐ main

☐ begin

☐ include

5. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

☐ `Mineur`
`Majeur`

☐ `Mineur`

☐ `Majeur`

☐ rien

6. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

☐ `#include <studio.h>`

☐ `#include <studlib.h>`

☐ `#include <stdio.h>`

☐ `#appart <stdlib.h>`

7. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
printf("Majeur\n");
```

affichera :

☐ `Mineur`
`Majeur`

☐ `Mineur`

☐ `Majeur`

☐ rien

8. 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 harmonique

☐ l'analyse des entrées clavier

☐ l'analyse sémantique

9. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :

☐ `kwrite TP4`

☐ `new TP4`

☐ `mkdir TP4`

☐ `yppasswd`

10. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

☐ `Mineur`

☐ `Mineur`
`Majeur`

☐ `Majeur`

☐ rien

11. Quel est l'opérateur de différence en C :
    - ☐ `!=`
    - ☐ `!`
    - ☐ $\neq$
    - ☐ `<>`

12. L'ordonnancement par tourniquet permet :
    - ☐ de doubler la mémoire disponible
    - ☐ 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

13. Le code suivant :

```
int i;
for (i = 0; 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
- ☐ 0 1 2 3

14. 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 :
- ☐ 4
- ☐ 0
- ☐ 16
- ☐ 8

15. 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  }
```

- ☐ la variable x vaut 0
- ☐ le programme affiche "Faux"
- ☐ la variable x vaut 5 et la variable y vaut 0
- ☐ la variable y vaut 5

16. 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 = 5`
- ☐ `j = 0`
- ☐ `j = %d`
- ☐ `j = 4`

17. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
    - ☐ `gcc prog.c -o -Wall prog.exe`
    - ☐ `gcc -Wall prog.c -o prog.exe`
    - ☐ `gcc prog.exe -Wall -o prog.c`
    - ☐ `gcc -Wall prog.exe -o prog.c`

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=%d et y=%d\n",x y);`
    - ☐ `printf("x=%x et y=%y\n");`
    - ☐ `printf("x=%d et y=%d\n",x,y);`

19. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
    - ☐ `#include <studlib.h>`
    - ☐ `#include <stdio.h>`
    - ☐ `#appart <stdlib.h>`
    - ☐ `#include <studio.h>`

20. Dans la commande gcc, l'option `-Wall` signifie :
    - ☐ que l'on veut voir tous les avertissements
    - ☐ qu'on veut changer alétoirement de fond d'écran
    - ☐ qu'il faut indenter le fichier source
    - ☐ qu'il faut lancer un déboggueur

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
   - ☐ `#define N =  3`
   - ☐ `#define N 3`
   - ☐ `#define taille = N`
   - ☐ `#define taille = 3`
2. Un bit est :
   - ☐ un battement d'horloge processeur
   - ☐ un chiffre binaire (0 ou 1)
   - ☐ l'instruction qui met fin à un programme
   - ☐ la longueur d'un mot mémoire
3. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :
   - ☐ 0 2 4 6 8
   - ☐ 8 6 4 2 0
   - ☐ 8 2
   - ☐ 8 6 4 2
4. Un registre du processeur est :
   - ☐ une gamme de fréquence de fonctionnement du processeur
   - ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs
   - ☐ un composant qui contient la liste des fichiers du système
   - ☐ une unité de calcul spécialisée de l'ordinateur
5. Le langage C est un langage
   - ☐ lu, écrit, parlé
   - ☐ interprété
   - ☐ composé
   - ☐ compilé
6. Dans la commande gcc, l'option `-Wall` signifie :
   - ☐ qu'on veut changer alétoirement de fond d'écran
   - ☐ qu'il faut indenter le fichier source
   - ☐ que l'on veut voir tous les avertissements
   - ☐ qu'il faut lancer un déboggueur
7. 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
   - ☐ 6
   - ☐ 15
8. 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
```

   - ☐ la case mémoire 8 contiendra 16
   - ☐ la case mémoire 8 contiendra 0
   - ☐ le terminal affiche 8
   - ☐ le bus explose
9. Le code suivant :

```
int i;
for (i = 0; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :
   - ☐ 4 3 2 1 0
   - ☐ 4 3 2 1
   - ☐ 0 1 2 3
   - ☐ 0 1 2 3 4
10. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
    - ☐ `yppasswd`
    - ☐ `new TP4`
    - ☐ `kwrite TP4`
    - ☐ `mkdir TP4`
11. 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
    - ☐ 1 2 3 4
    - ☐ 0 1 2 3 4
12. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
    - ☐ retourner un bloc
    - ☐ répéter un bloc tant qu'une condition est vérifée
    - ☐ mettre les blocs en séquence les uns à la suite des autres
    - ☐ sélectionner entre deux blocs à l'aide d'une condition
13. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
    - ☐ `#appart <stdlib.h>`
    - ☐ `#include <stdio.h>`
    - ☐ `#include <studlib.h>`
    - ☐ `#include <studio.h>`

14. 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 1 0 1 0 1 0 1`
- ☐ `1 2 1 2 3`
- ☐ `0 0 0 1 1 1`
- ☐ `0 1 2 0 1 2`

15. 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  }
```

- ☐ la variable x vaut 0
- ☐ la variable y vaut 5
- ☐ la variable x vaut 5 et la variable y vaut 0
- ☐ le programme affiche "Faux"

16. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
printf("Majeur\n");
```

affichera :

- ☐ `Majeur`
- ☐ `Mineur`
- ☐ `Mineur`
  `Majeur`
- ☐ rien

17. 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
- ☐ 4
- ☐ 8
- ☐ 16

18. 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  }
```

- ☐ la variable x vaut 16
- ☐ la variable x vaut $-\frac{1}{2}$
- ☐ le programme affiche `****`
- ☐ le programme affiche x

19. Le code suivant :

```
int somme = 0;
int i;
for (i = 1; i < 4; i = i + 1)
{
  somme = somme + i;
}
printf("%d", somme);
```

affichera :

- ☐ 6
- ☐ 42
- ☐ 0
- ☐ 1

20. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ rien
- ☐ `Mineur`
- ☐ `Majeur`
- ☐ `Mineur`
  `Majeur`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse lexicale
   - ☐ analyse syntaxique
   - ☐ analyse harmonique
   - ☐ analyse sémantique
2. Un bit est :
   - ☐ un battement d'horloge processeur
   - ☐ un chiffre binaire (0 ou 1)
   - ☐ l'instruction qui met fin à un programme
   - ☐ la longueur d'un mot mémoire
3. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
   - ☐ `int loop n;`
   - ☐ `int %d;`
   - ☐ `loop i;`
   - ☐ `int k;`
4. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :
   - ☐ 0 2 4 6 8
   - ☐ 8 6 4 2
   - ☐ 8 6 4 2 0
   - ☐ 8 2
5. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
   - ☐ `mkdir TP4`
   - ☐ `new TP4`
   - ☐ `kwrite TP4`
   - ☐ `yppasswd`
6. 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 :
   - ☐ 6
   - ☐ 10
   - ☐ 0
   - ☐ 15
7. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
   - ☐ `gcc prog.c -o -Wall prog.exe`
   - ☐ `gcc prog.exe -Wall -o prog.c`
   - ☐ `gcc -Wall prog.exe -o prog.c`
   - ☐ `gcc -Wall prog.c -o prog.exe`
8. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

   affichera :
   - ☐ `Mineur`
`Majeur`
   - ☐ `Mineur`
   - ☐ `Majeur`
   - ☐ rien
9. Dans la commande gcc, l'option `-Wall` signifie :
   - ☐ qu'il faut lancer un déboggueur
   - ☐ qu'on veut changer alétoirement de fond d'écran
   - ☐ que l'on veut voir tous les avertissements
   - ☐ qu'il faut indenter le fichier source
10. 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 x vaut 3
    - ☐ la variable y vaut 5
    - ☐ la variable x vaut 5 et la variable y vaut 3
    - ☐ le programme affiche "Faux"
11. 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=0;i<5;i=i+1)`
    - ☐ `for(i=0;i<=5;i=i+1)`
    - ☐ `for(i=1;i<5;i=i+1)`
    - ☐ `for(i=1;i<=5;i=i+1)`
12. 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 illisible pour les humains
    - ☐ un document qui doit être protégé
    - ☐ un fichier texte qui sera traduit en instructions processeur

13. 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 `****`
- ☐ le programme affiche x
- ☐ la variable x vaut 16
- ☐ la variable x vaut $-\frac{1}{2}$

14. Si cette erreur apparaît à la compilation :
`error: expected ';' before '}' token` que doit-on chercher dans le programme ?
- ☐ une accolade en trop
- ☐ un point-virgule en trop
- ☐ un point-virgule manquant
- ☐ une accolade manquante

15. 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 1 0 1 0 1 0 1`
- ☐ `0 0 0 1 1 1`
- ☐ `0 1 2 0 1 2`
- ☐ `1 2 1 2 3`

16. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Majeur`
- ☐ `Mineur`
  `Majeur`
- ☐ `Mineur`
- ☐ rien

17. 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>`

18. 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'analyse des entrées clavier
- ☐ l'analyse sémantique
- ☐ l'édition de liens
- ☐ l'analyse harmonique

19. Un programme en langage C doit comporter une et une seule définition de la fonction :
- ☐ main
- ☐ begin
- ☐ include
- ☐ init

20. 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 = %d`
- ☐ `j = 0`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. 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 x
   - ☐ la variable x vaut $-\frac{1}{2}$
   - ☐ la variable x vaut 16
   - ☐ le programme affiche `****`

2. Les lignes

```
int i;
int x=0;
for(i=0,i<5,i=i+1)
{
  x=x+1;
}
```

   - ☐ ne comportent aucune erreur
   - ☐ 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
   - ☐ comportent une erreur qui ne sera pas détectée

3. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse harmonique
   - ☐ analyse lexicale
   - ☐ analyse syntaxique
   - ☐ analyse sémantique

4. Si cette erreur apparaît à la compilation : `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
   - ☐ une accolade manquante
   - ☐ une accolade en trop
   - ☐ un point-virgule manquant
   - ☐ un point-virgule en trop

5. 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`

6. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
   - ☐ `new TP4`
   - ☐ `mkdir TP4`
   - ☐ `kwrite TP4`
   - ☐ `yppasswd`

7. Quel est l'opérateur de différence en C :
   - ☐ `!`
   - ☐ $\neq$
   - ☐ `<>`
   - ☐ `!=`

8. Le code suivant :

```
int somme = 0;
int i;
for (i = 1; i < 4; i = i + 1)
{
  somme = somme + i;
}
printf("%d", somme);
```

   affichera :
   - ☐ 42
   - ☐ 0
   - ☐ 6
   - ☐ 1

9. Un fichier source est :
   - ☐ un document illisible pour les humains
   - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
   - ☐ un fichier texte qui sera traduit en instructions processeur
   - ☐ un document qui doit être protégé
   - ☐ un document de référence du système

10. 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 :
   - ☐ 0
   - ☐ 8
   - ☐ 16
   - ☐ 4

11. Une *segmentation fault* est une erreur qui survient lorsque :

- ☐ 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
- ☐ la division du programme en zones homogènes échoue
- ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal

12. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :

- ☐ les fichiers du disque
- ☐ des processus
- ☐ en temps d'accès
- ☐ certaines données de la mémoire de travail

13. Le langage C est un langage

- ☐ composé
- ☐ compilé
- ☐ interprété
- ☐ lu, écrit, parlé

14. 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  }
```

- ☐ la variable x vaut 5 et la variable y vaut 0
- ☐ le programme affiche "Faux"
- ☐ la variable y vaut 5
- ☐ la variable x vaut 0

15. L'ordonnancement par tourniquet permet :

- ☐ de ne pas perdre de temps avec la commutation de contexte
- ☐ d'afficher des ronds colorés à l'écran
- ☐ de doubler la mémoire disponible
- ☐ d'entretenir l'illusion que les processus tournent en parallèle

16. 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 `5`
- ☐ le terminal affiche `x = 5`
- ☐ le terminal affiche "Faux"
- ☐ le terminal affiche `x = 2`

17. 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=%d et y=%d\n",x,y);`
- ☐ `printf("x=%d et y=%d\n",x y);`
- ☐ `printf("x=%x et y=%y\n");`

18. Sur unix (ou linux), la commande `mkdir` permet de :

- ☐ ouvrir un fichier texte
- ☐ créer un fichier texte
- ☐ changer de répertoire courant
- ☐ créer un répertoire

19. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.

- ☐ `#define N =  3`
- ☐ `#define taille = N`
- ☐ `#define N 3`
- ☐ `#define taille = 3`

20. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#include <stdio.h>`
- ☐ `#appart <stdlib.h>`
- ☐ `#include <studlib.h>`
- ☐ `#include <studio.h>`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

☐ 8 6 4 2 0

☐ 8 2

☐ 8 6 4 2

☐ 0 2 4 6 8

2. Un programme en langage C doit comporter une et une seule définition de la fonction :

☐ include

☐ main

☐ begin

☐ init

3. Dans la commande gcc, l'option `-Wall` signifie :

☐ que l'on veut voir tous les avertissements

☐ qu'il faut lancer un déboggueur

☐ qu'on veut changer alétoirement de fond d'écran

☐ qu'il faut indenter le fichier source

4. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :

☐ analyse lexicale

☐ analyse harmonique

☐ analyse syntaxique

☐ analyse sémantique

5. 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 :

☐ 4

☐ 8

☐ 0

☐ 16

6. Le code suivant :

```
int somme = 0;
int i;
for (i = 1; i < 4; i = i + 1)
{
  somme = somme + i;
}
printf("%d", somme);
```

affichera :

☐ 0

☐ 42

☐ 1

☐ 6

7. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

☐ `#appart <stdlib.h>`

☐ `#include <stdio.h>`

☐ `#include <studio.h>`

☐ `#include <studlib.h>`

8. Le bus système sert à :

☐ Écrire des données sur le dique dur

☐ transporter les processus du tourniquet au processeur

☐ Arriver à l'heure en cours

☐ Transférer des données et intructions entre processeur et mémoire

9. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

☐ `#appart <stdlib.h>`

☐ `#include <stdio.h>`

☐ `#include <studio.h>`

☐ `#include <studlib.h>`

10. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :

☐ tour à tour, un petit peu à chaque fois

☐ tous ensemble

☐ chacun son tour, après que le processus précédent a terminé

☐ en parallèle, chacun dans un registre

11. 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
```

☐ la case mémoire 8 contiendra 0

☐ le bus explose

☐ la case mémoire 8 contiendra 16

☐ le terminal affiche 8

12. Sous unix (ou linux), la commande `cd` permet de :

☐ ouvir un bureau partagé (common desktop)

☐ changer de répertoire courant

☐ récupérer un programme arrêté avec la commande `ab`

☐ jouer de la musique

☐ détruire un fichier

13. 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 ?

- ☐ `** *** **** *****`
- ☐ `***** **** *** ***`
- ☐ `** ** ** ** ** **`
- ☐ `**** **** **** ****`

14. 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`

15. Le code suivant :

```
int i;
for (i = 1; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 0 1 2 3 4
- ☐ 1 2 3 4
- ☐ 4 3 2 1 0
- ☐ 4 3 2 1

16. 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  }
```

- ☐ la variable x vaut 16
- ☐ le programme affiche x
- ☐ la variable x vaut $-\frac{1}{2}$
- ☐ le programme affiche `****`

17. Le code suivant :

```
int i;
for (i = 0; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 4 3 2 1 0
- ☐ 4 3 2 1
- ☐ 0 1 2 3
- ☐ 0 1 2 3 4

18. Le langage C est un langage

- ☐ lu, écrit, parlé
- ☐ composé
- ☐ compilé
- ☐ interprété

19. 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 7
- ☐ 0 2 4 6
- ☐ 0 1 2 3 4 5 6
- ☐ 0 2 4 6 8

20. 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'analyse sémantique
- ☐ l'analyse des entrées clavier
- ☐ l'analyse harmonique
- ☐ l'édition de liens

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Quel est l'opérateur de différence en C :
   - ☐ `<>`
   - ☐ $\neq$
   - ☐ `!`
   - ☐ `!=`

2. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

   affichera :
   - ☐ `Mineur`
   - ☐ `Mineur`
     `Majeur`
   - ☐ `Majeur`
   - ☐ rien

3. Dans la commande gcc, l'option `-Wall` signifie :
   - ☐ que l'on veut voir tous les avertissements
   - ☐ qu'il faut indenter le fichier source
   - ☐ qu'il faut lancer un déboggueur
   - ☐ qu'on veut changer alétoirement de fond d'écran

4. 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
```

   - ☐ la case mémoire 8 contiendra 0
   - ☐ la case mémoire 8 contiendra 16
   - ☐ le terminal affiche 8
   - ☐ le bus explose

5. 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 :
   - ☐ 16
   - ☐ 4
   - ☐ 8
   - ☐ 0

6. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

   affichera :
   - ☐ `Mineur`
   - ☐ `Majeur`
   - ☐ `Mineur`
     `Majeur`
   - ☐ rien

7. 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 1 0 1 0 1 0 1`
   - ☐ `0 0 0 1 1 1`
   - ☐ `1 2 1 2 3`
   - ☐ `0 1 2 0 1 2`

8. 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 x vaut 5 et la variable y vaut 3
   - ☐ la variable x vaut 3
   - ☐ la variable y vaut 5
   - ☐ le programme affiche "Faux"

9. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
   - ☐ tour à tour, un petit peu à chaque fois
   - ☐ chacun son tour, après que le processus précédent a terminé
   - ☐ en parallèle, chacun dans un registre
   - ☐ tous ensemble

10. Un bit est :
    - ☐ un battement d'horloge processeur
    - ☐ l'instruction qui met fin à un programme
    - ☐ un chiffre binaire (0 ou 1)
    - ☐ la longueur d'un mot mémoire

11. 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'analyse sémantique
    - ☐ l'édition de liens
    - ☐ l'analyse harmonique
    - ☐ l'analyse des entrées clavier

12. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
    - ☐ `int loop n;`
    - ☐ `loop i;`
    - ☐ `int %d;`
    - ☐ `int k;`

13. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
    - ☐ analyse harmonique
    - ☐ analyse sémantique
    - ☐ analyse syntaxique
    - ☐ analyse lexicale

14. Un registre du processeur est :
    - ☐ 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
    - ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs

15. 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 3`
    - ☐ `0 1 2 0 1 2`
    - ☐ `0 1 2 3 0 1 2`

16. Le code suivant :

```
int i;
for (i = 0; i < 7; i = i + 2)
{
    printf("%d ", i);
}
printf("\n");
```

    affichera :
    - ☐ 0 2 4 6 8
    - ☐ 0 2 4 6
    - ☐ 0 1 2 3 4 5 6
    - ☐ 0 1 2 3 4 5 6 7

17. 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=0;i<=5;i=i+1)`
    - ☐ `for(i=1;i<=5;i=i+1)`
    - ☐ `for(i=1;i<5;i=i+1)`
    - ☐ `for(i=0;i<5;i=i+1)`

18. 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

19. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

    affichera :
    - ☐ 8 6 4 2
    - ☐ 8 2
    - ☐ 8 6 4 2 0
    - ☐ 0 2 4 6 8

20. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ changer de répertoire courant
    - ☐ ouvrir un fichier texte
    - ☐ créer un fichier texte
    - ☐ créer un répertoire

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :

   ☐ `kwrite TP4`

   ☐ `mkdir TP4`

   ☐ `new TP4`

   ☐ `yppasswd`

2. Le code suivant :

```
int i;
for (i = 4; i >= 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :

   ☐ 0 1 2 3 4

   ☐ 1 2 3 4

   ☐ 4 3 2 1

   ☐ 4 3 2 1 0

3. 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 :

   ☐ 0

   ☐ 16

   ☐ 4

   ☐ 8

4. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

   affichera :

   ☐ rien

   ☐ `Mineur`
   `Majeur`

   ☐ `Mineur`

   ☐ `Majeur`

5. Le code suivant :

```
int i;
for (i = 0; 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

   ☐ 0 1 2 3

6. 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 ?

   ☐ `***** **** *** ***`

   ☐ `** ** ** ** ** **`

   ☐ `**** **** **** ****`

   ☐ `** *** **** *****`

7. 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  }
```

   ☐ la variable x vaut $-\frac{1}{2}$

   ☐ le programme affiche x

   ☐ la variable x vaut 16

   ☐ le programme affiche `****`

8. 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 1 2 0 1 2`

   ☐ `1 2 1 2 3`

   ☐ `0 1 0 1 0 1 0 1`

   ☐ `0 0 0 1 1 1`

9. Sous unix (ou linux), la commande `ls` permet de :

   ☐ afficher la liste de fichiers contenus dans un répertoire

   ☐ voir des clips musicaux

   ☐ afficher le contenu d'un fichier texte

   ☐ compiler un programme

10. 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 :

- ☐ 10
- ☐ 15
- ☐ 0
- ☐ 6

11. 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 bus explose
- ☐ le terminal affiche 8
- ☐ la case mémoire 8 contiendra 0
- ☐ la case mémoire 8 contiendra 16

12. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#appart <stdlib.h>`
- ☐ `#include <studlib.h>`
- ☐ `#include <stdio.h>`
- ☐ `#include <studio.h>`

13. Un bit est :

- ☐ un battement d'horloge processeur
- ☐ l'instruction qui met fin à un programme
- ☐ un chiffre binaire (0 ou 1)
- ☐ la longueur d'un mot mémoire

14. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Majeur`
- ☐ `Mineur`
- ☐ rien
- ☐ `Mineur`
  `Majeur`

15. 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
- ☐ ne comportent aucune erreur
- ☐ comportent une erreur qui sera détectée au cours de l'édition de lien

16. Sur unix (ou linux), la commande `mkdir` permet de :

- ☐ créer un fichier texte
- ☐ créer un répertoire
- ☐ ouvrir un fichier texte
- ☐ changer de répertoire courant

17. L'ordonnancement par tourniquet permet :

- ☐ de ne pas perdre de temps avec la commutation de contexte
- ☐ d'afficher des ronds colorés à l'écran
- ☐ d'entretenir l'illusion que les processus tournent en parallèle
- ☐ de doubler la mémoire disponible

18. 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 `x = 5`
- ☐ le terminal affiche `x = 2`
- ☐ le terminal affiche `5`
- ☐ le terminal affiche "Faux"

19. Un programme en langage C doit comporter une et une seule définition de la fonction :

- ☐ include
- ☐ init
- ☐ begin
- ☐ main

20. Sous unix (ou linux), la commande `cd` permet de :

- ☐ jouer de la musique
- ☐ détruire un fichier
- ☐ changer de répertoire courant
- ☐ récupérer un programme arrêté avec la commande `ab`
- ☐ ouvir un bureau partagé (common desktop)

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. 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 :

☐ 16

☐ 0

☐ 4

☐ 8

2. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

☐ `#appart <stdlib.h>`

☐ `#include <studio.h>`

☐ `#include <stdio.h>`

☐ `#include <studlib.h>`

3. Un fichier source est :

☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur

☐ un fichier texte qui sera traduit en instructions processeur

☐ un document de référence du système

☐ un document illisible pour les humains

☐ un document qui doit être protégé

4. 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

5. Le code suivant :

```
int i;
for (i = 0; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

☐ 0 1 2 3 4

☐ 0 1 2 3

☐ 4 3 2 1 0

☐ 4 3 2 1

6. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :

☐ `new TP4`

☐ `kwrite TP4`

☐ `yppasswd`

☐ `mkdir TP4`

7. Le langage C est un langage

☐ lu, écrit, parlé

☐ composé

☐ interprété

☐ compilé

8. 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=%x et y=%y\n");`

☐ `printf("x=%d et y=%d\n",x y);`

☐ `printf("x=%d et y=%d\n",x,y);`

☐ `printf("x=%d et y=%d\n,x,y");`

9. 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

10. Le code suivant :

```
int i;
for (i = 4; i > 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

☐ 4 3 2 1 0

☐ 0 1 2 3

☐ 4 3 2 1

☐ 0 1 2 3 4

11. 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 sémantique

☐ l'analyse des entrées clavier

☐ l'analyse harmonique

12. 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

☐ 1 2 3 4

☐ 4 3 2 1 0

13. Le code suivant :

```
int i;
for (i = 4; i >= 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

☐ 4 3 2 1 0

☐ 1 2 3 4

☐ 0 1 2 3 4

☐ 4 3 2 1

14. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
printf("Majeur\n");
```

affichera :

☐ `Mineur`

☐ `Majeur`

☐ rien

☐ `Mineur`
`Majeur`

15. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.

☐ `#define N =  3`

☐ `#define taille = 3`

☐ `#define N 3`

☐ `#define taille = N`

16. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

☐ `Mineur`

☐ rien

☐ `Mineur`
`Majeur`

☐ `Majeur`

17. L'ordonnancement par tourniquet permet :

☐ d'afficher des ronds colorés à l'écran

☐ de ne pas perdre de temps avec la commutation de contexte

☐ de doubler la mémoire disponible

☐ d'entretenir l'illusion que les processus tournent en parallèle

18. 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 :

☐ 6

☐ 20

☐ 3

☐ 16

19. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction

☐ `loop i;`

☐ `int k;`

☐ `int %d;`

☐ `int loop n;`

20. 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 7

☐ 0 2 4 6 8

☐ 0 1 2 3 4 5 6

☐ 0 2 4 6

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
   - ☐ `mkdir TP4`
   - ☐ `new TP4`
   - ☐ `kwrite TP4`
   - ☐ `yppasswd`
2. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
   - ☐ `#define taille = 3`
   - ☐ `#define taille = N`
   - ☐ `#define N 3`
   - ☐ `#define N =  3`
3. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :
   - ☐ en temps d'accès
   - ☐ des processus
   - ☐ les fichiers du disque
   - ☐ certaines données de la mémoire de travail
4. Si cette erreur apparaît à la compilation :
   `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
   - ☐ un point-virgule en trop
   - ☐ une accolade manquante
   - ☐ une accolade en trop
   - ☐ un point-virgule manquant
5. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :
   - ☐ 8 2
   - ☐ 8 6 4 2
   - ☐ 0 2 4 6 8
   - ☐ 8 6 4 2 0
6. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ main
   - ☐ include
   - ☐ begin
   - ☐ init
7. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

   affichera :
   - ☐ rien
   - ☐ `Mineur`
`Majeur`
   - ☐ `Majeur`
   - ☐ `Mineur`
8. Sur unix (ou linux), la commande `mkdir` permet de :
   - ☐ ouvrir un fichier texte
   - ☐ créer un répertoire
   - ☐ changer de répertoire courant
   - ☐ créer un fichier texte
9. Un fichier source est :
   - ☐ un document de référence du système
   - ☐ un fichier texte qui sera traduit en instructions processeur
   - ☐ un document illisible pour les humains
   - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
   - ☐ un document qui doit être protégé
10. 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 ?
    - ☐ `1 2 1 2 3`
    - ☐ `0 1 0 1 0 1 0 1`
    - ☐ `0 1 2 0 1 2`
    - ☐ `0 0 0 1 1 1`
11. 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 sémantique
    - ☐ l'analyse harmonique
    - ☐ l'analyse des entrées clavier
12. 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 `5`
    - ☐ le terminal affiche `x = 2`
    - ☐ le terminal affiche `x = 5`

13. Quels calculs peut-on programmer en programmation structurée ?

- ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine
- ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine
- ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée
- ☐ certains programmes sont de vrais plats de spaghetti

14. 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 = %d`
- ☐ `j = 5`
- ☐ `j = 0`

15. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
printf("Majeur\n");
```

affichera :

- ☐ `Majeur`
- ☐ `Mineur`
- ☐ `Mineur`
  `Majeur`
- ☐ rien

16. Si cette erreur apparaît à la compilation : `Undefined symbols :"_prinft"` ou `référence indéfinie vers` ≪ `prinft` ≫ que doit-on chercher dans le programme ?

- ☐ un caractère interdit en C
- ☐ une faute de frappe dans un appel de fonction
- ☐ une variable non déclarée
- ☐ une directive préprocesseur `#include` manquante

17. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction

- ☐ `char tableau[5];`
- ☐ `int[] new tableau(5);`
- ☐ `int toto[taille=5];`
- ☐ `int tab[] = 5;`
- ☐ `int toto[5];`

18. Un bit est :

- ☐ l'instruction qui met fin à un programme
- ☐ un battement d'horloge processeur
- ☐ la longueur d'un mot mémoire
- ☐ un chiffre binaire (0 ou 1)

19. Sous unix (ou linux), la commande `cd` permet de :

- ☐ détruire un fichier
- ☐ jouer de la musique
- ☐ récupérer un programme arrêté avec la commande `ab`
- ☐ ouvir un bureau partagé (common desktop)
- ☐ changer de répertoire courant

20. Sous unix (ou linux), la commande `ls` permet de :

- ☐ compiler un programme
- ☐ afficher la liste de fichiers contenus dans un répertoire
- ☐ afficher le contenu d'un fichier texte
- ☐ voir des clips musicaux

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. 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 3
   - ☐ le programme affiche "Faux"
   - ☐ la variable x vaut 5 et la variable y vaut 3

2. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :

   - ☐ 8 2
   - ☐ 8 6 4 2 0
   - ☐ 0 2 4 6 8
   - ☐ 8 6 4 2

3. 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=%d et y=%d\n",x y);`
   - ☐ `printf("x=%d et y=%d\n,x,y");`
   - ☐ `printf("x=%x et y=%y\n");`

4. Quels calculs peut-on programmer 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
   - ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée

5. Un fichier source est :

   - ☐ un fichier texte qui sera traduit en instructions processeur
   - ☐ un document illisible pour les humains
   - ☐ un document qui doit être protégé
   - ☐ un document de référence du système
   - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur

6. 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 = 0`
   - ☐ `j = %d`
   - ☐ `j = 4`
   - ☐ `j = 5`

7. Le code suivant :

```
int i;
for (i = 0; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :

   - ☐ 4 3 2 1 0
   - ☐ 0 1 2 3 4
   - ☐ 0 1 2 3
   - ☐ 4 3 2 1

8. 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=0;i<=5;i=i+1)`
   - ☐ `for(i=1;i<5;i=i+1)`
   - ☐ `for(i=1;i<=5;i=i+1)`
   - ☐ `for(i=0;i<5;i=i+1)`

9. Dans la commande gcc, l'option `-Wall` signifie :

   - ☐ qu'il faut lancer un déboggueur
   - ☐ que l'on veut voir tous les avertissements
   - ☐ qu'on veut changer alétoirement de fond d'écran
   - ☐ qu'il faut indenter le fichier source

10. Une *segmentation fault* est une erreur qui survient lorsque :

    - ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
    - ☐ 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

11. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ rien
- ☐ `Mineur`
  `Majeur`
- ☐ `Majeur`
- ☐ `Mineur`

12. 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  }
```

- ☐ la variable x vaut 0
- ☐ la variable y vaut 5
- ☐ le programme affiche "Faux"
- ☐ la variable x vaut 5 et la variable y vaut 0

13. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
printf("Majeur\n");
```

affichera :

- ☐ `Majeur`
- ☐ `Mineur`
- ☐ rien
- ☐ `Mineur`
  `Majeur`

14. Sous unix (ou linux), la commande `cd` permet de :

- ☐ jouer de la musique
- ☐ changer de répertoire courant
- ☐ ouvir un bureau partagé (common desktop)
- ☐ récupérer un programme arrêté avec la commande `ab`
- ☐ détruire un fichier

15. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Majeur`
- ☐ `Mineur`
  `Majeur`
- ☐ `Mineur`
- ☐ rien

16. 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 :

- ☐ 16
- ☐ 0
- ☐ 4
- ☐ 8

17. Sur unix (ou linux), la commande `mkdir` permet de :

- ☐ créer un répertoire
- ☐ créer un fichier texte
- ☐ ouvrir un fichier texte
- ☐ changer de répertoire courant

18. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :

- ☐ en temps d'accès
- ☐ certaines données de la mémoire de travail
- ☐ des processus
- ☐ les fichiers du disque

19. Les lignes

```
int i;
int x=0;
for(i=0,i<5,i=i+1)
{
  x=x+1;
}
```

- ☐ comportent une erreur qui sera détectée au cours de l'édition de lien
- ☐ comportent une erreur qui sera détectée au cours de l'analyse syntaxique
- ☐ comportent une erreur qui ne sera pas détectée
- ☐ ne comportent aucune erreur

20. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :

- ☐ `kwrite TP4`
- ☐ `mkdir TP4`
- ☐ `new TP4`
- ☐ `yppasswd`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. L'ordonnancement par tourniquet permet :
   - ☐ de doubler la mémoire disponible
   - ☐ de ne pas perdre de temps avec la commutation de contexte
   - ☐ d'entretenir l'illusion que les processus tournent en parallèle
   - ☐ d'afficher des ronds colorés à l'écran

2. Les lignes

```
int i;
int x=0;
for(i=0,i<5,i=i+1)
{
  x=x+1;
}
```

   - ☐ comportent une erreur qui sera détectée au cours de l'analyse syntaxique
   - ☐ comportent une erreur qui ne sera pas détectée
   - ☐ comportent une erreur qui sera détectée au cours de l'édition de lien
   - ☐ ne comportent aucune erreur

3. 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 ?
   - ☐ `**** **** **** ****`
   - ☐ `** *** **** *****`
   - ☐ `** ** ** ** ** **`
   - ☐ `***** **** *** ***`

4. 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 2 0 1 2`
   - ☐ `0 1 0 1 0 1 0 1`

5. 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 2 0 1 2`
   - ☐ `0 1 0 1 0 1`
   - ☐ `1 2 3 1 2`
   - ☐ `0 0 1 1 2 2`

6. Dans la commande gcc, l'option `-Wall` signifie :
   - ☐ qu'il faut indenter le fichier source
   - ☐ qu'il faut lancer un déboggueur
   - ☐ qu'on veut changer alétoirement de fond d'écran
   - ☐ que l'on veut voir tous les avertissements

7. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#appart <stdlib.h>`
   - ☐ `#include <studlib.h>`
   - ☐ `#include <stdio.h>`
   - ☐ `#include <studio.h>`

8. Sur unix (ou linux), la commande `mkdir` permet de :
   - ☐ ouvrir un fichier texte
   - ☐ créer un répertoire
   - ☐ créer un fichier texte
   - ☐ changer de répertoire courant

9. Le bus système sert à :
   - ☐ Arriver à l'heure en cours
   - ☐ transporter les processus du tourniquet au processeur
   - ☐ Transférer des données et intructions entre processeur et mémoire
   - ☐ Écrire des données sur le dique dur

10. Un fichier source est :
    - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
    - ☐ un document de référence du système
    - ☐ un document qui doit être protégé
    - ☐ un document illisible pour les humains
    - ☐ un fichier texte qui sera traduit en instructions processeur

11. 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 x vaut 3
    - ☐ la variable y vaut 5
    - ☐ le programme affiche "Faux"
    - ☐ la variable x vaut 5 et la variable y vaut 3

12. 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 :

- ☐ 4
- ☐ 16
- ☐ 0
- ☐ 8

13. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Majeur`
- ☐ `Mineur`
  `Majeur`
- ☐ rien
- ☐ `Mineur`

14. Pour compiler un programme `prog.c`, on utilise la ligne de commande :

- ☐ `gcc prog.c -o -Wall prog.exe`
- ☐ `gcc prog.exe -Wall -o prog.c`
- ☐ `gcc -Wall prog.exe -o prog.c`
- ☐ `gcc -Wall prog.c -o prog.exe`

15. 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  }
```

- ☐ la variable x vaut 5 et la variable y vaut 0
- ☐ la variable y vaut 5
- ☐ la variable x vaut 0
- ☐ le programme affiche "Faux"

16. Le code suivant :

```
int i;
for (i = 4; i > 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 0 1 2 3 4
- ☐ 0 1 2 3
- ☐ 4 3 2 1
- ☐ 4 3 2 1 0

17. Le langage C est un langage

- ☐ lu, écrit, parlé
- ☐ interprété
- ☐ composé
- ☐ compilé

18. 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 bus explose
- ☐ le terminal affiche 8
- ☐ la case mémoire 8 contiendra 0
- ☐ la case mémoire 8 contiendra 16

19. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :

- ☐ tous ensemble
- ☐ tour à tour, un petit peu à chaque fois
- ☐ en parallèle, chacun dans un registre
- ☐ chacun son tour, après que le processus précédent a terminé

20. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :

- ☐ `mkdir TP4`
- ☐ `kwrite TP4`
- ☐ `new TP4`
- ☐ `yppasswd`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Sous unix (ou linux), la commande `ls` permet de :
   - ☐ voir des clips musicaux
   - ☐ afficher le contenu d'un fichier texte
   - ☐ compiler un programme
   - ☐ afficher la liste de fichiers contenus dans un répertoire
2. Le bus système sert à :
   - ☐ transporter les processus du tourniquet au processeur
   - ☐ Écrire des données sur le dique dur
   - ☐ Transférer des données et intructions entre processeur et mémoire
   - ☐ Arriver à l'heure en cours
3. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
   - ☐ `yppasswd`
   - ☐ `mkdir TP4`
   - ☐ `kwrite TP4`
   - ☐ `new TP4`
4. 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=%x et y=%y\n");`
   - ☐ `printf("x=%d et y=%d\n,x,y");`
   - ☐ `printf("x=%d et y=%d\n",x,y);`
   - ☐ `printf("x=%d et y=%d\n",x y);`
5. Sous unix (ou linux), la commande `cd` permet de :
   - ☐ changer de répertoire courant
   - ☐ récupérer un programme arrêté avec la commande `ab`
   - ☐ ouvir un bureau partagé (common desktop)
   - ☐ détruire un fichier
   - ☐ jouer de la musique
6. 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
   - ☐ 0 1 2 3 4
7. Le code suivant :

```
int i;
for (i = 0; i < 7; i = i + 2)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :
   - ☐ 0 2 4 6 8
   - ☐ 0 2 4 6
   - ☐ 0 1 2 3 4 5 6
   - ☐ 0 1 2 3 4 5 6 7
8. L'ordonnancement par tourniquet permet :
   - ☐ d'entretenir l'illusion que les processus tournent en parallèle
   - ☐ de ne pas perdre de temps avec la commutation de contexte
   - ☐ d'afficher des ronds colorés à l'écran
   - ☐ de doubler la mémoire disponible
9. Un registre du processeur est :
   - ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs
   - ☐ un composant qui contient la liste des fichiers du système
   - ☐ une gamme de fréquence de fonctionnement du processeur
   - ☐ une unité de calcul spécialisée de l'ordinateur
10. Si cette erreur apparaît à la compilation : `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
    - ☐ un point-virgule en trop
    - ☐ une accolade en trop
    - ☐ une accolade manquante
    - ☐ un point-virgule manquant
11. Dans la commande gcc, l'option `-Wall` signifie :
    - ☐ qu'il faut indenter le fichier source
    - ☐ qu'on veut changer alétoirement de fond d'écran
    - ☐ que l'on veut voir tous les avertissements
    - ☐ qu'il faut lancer un déboggueur
12. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
    - ☐ analyse harmonique
    - ☐ analyse lexicale
    - ☐ analyse syntaxique
    - ☐ analyse sémantique
13. 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 `x = 2`
    - ☐ le terminal affiche `5`
    - ☐ le terminal affiche "Faux"
    - ☐ le terminal affiche `x = 5`
14. Une *segmentation fault* est une erreur qui survient lorsque :
    - ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
    - ☐ 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

- ☐ 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

15. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 0 2 4 6 8
- ☐ 8 6 4 2 0
- ☐ 8 6 4 2
- ☐ 8 2

16. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#include <stdio.h>`
- ☐ `#include <studlib.h>`
- ☐ `#appart <stdlib.h>`
- ☐ `#include <studio.h>`

17. Si cette erreur apparaît à la compilation : `Undefined symbols :"_prinft"` ou `référence indéfinie vers « prinft »` que doit-on chercher dans le programme ?

- ☐ une faute de frappe dans un appel de fonction
- ☐ une directive préprocesseur `#include` manquante
- ☐ un caractère interdit en C
- ☐ une variable non déclarée

18. 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 :

- ☐ 4
- ☐ 8
- ☐ 0
- ☐ 16

19. 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 1 2 0 1 2`
- ☐ `0 0 0 1 1 1`
- ☐ `1 2 1 2 3`
- ☐ `0 1 0 1 0 1 0 1`

20. Le langage C est un langage

- ☐ compilé
- ☐ lu, écrit, parlé
- ☐ composé
- ☐ interprété

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Un bit est :
   - ☐ un chiffre binaire (0 ou 1)
   - ☐ la longueur d'un mot mémoire
   - ☐ un battement d'horloge processeur
   - ☐ l'instruction qui met fin à un programme

2. 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=0;i<5;i=i+1)`
   - ☐ `for(i=1;i<5;i=i+1)`
   - ☐ `for(i=0;i<=5;i=i+1)`
   - ☐ `for(i=1;i<=5;i=i+1)`

3. 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 = 5`
   - ☐ `j = 4`
   - ☐ `j = %d`
   - ☐ `j = 0`

4. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

   affichera :
   - ☐ `Mineur`
   - ☐ `Majeur`
   - ☐ `Mineur`
     `Majeur`
   - ☐ rien

5. Le bus système sert à :
   - ☐ Arriver à l'heure en cours
   - ☐ Transférer des données et intructions entre processeur et mémoire
   - ☐ Écrire des données sur le dique dur
   - ☐ transporter les processus du tourniquet au processeur

6. Si cette erreur apparaît à la compilation :
   `Undefined symbols :"_prinft"` ou
   `référence indéfinie vers` « `prinft` » que doit-on chercher dans le programme ?
   - ☐ une variable non déclarée
   - ☐ un caractère interdit en C
   - ☐ une faute de frappe dans un appel de fonction
   - ☐ une directive préprocesseur `#include` manquante

7. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
printf("Majeur\n");
```

   affichera :
   - ☐ rien
   - ☐ `Majeur`
   - ☐ `Mineur`
     `Majeur`
   - ☐ `Mineur`

8. 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 `x = 2`
   - ☐ le terminal affiche `5`
   - ☐ le terminal affiche `x = 5`
   - ☐ le terminal affiche "Faux"

9. Dans la commande gcc, l'option `-Wall` signifie :
   - ☐ qu'on veut changer alétoirement de fond d'écran
   - ☐ qu'il faut lancer un déboggueur
   - ☐ qu'il faut indenter le fichier source
   - ☐ que l'on veut voir tous les avertissements

10. 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
    - ☐ 4
    - ☐ 8
    - ☐ 16

11. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
    - ☐ `gcc -Wall prog.exe -o prog.c`
    - ☐ `gcc prog.c -o -Wall prog.exe`
    - ☐ `gcc -Wall prog.c -o prog.exe`
    - ☐ `gcc prog.exe -Wall -o prog.c`

12. Le code suivant :

```
int i;
for (i = 1; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 0 1 2 3 4
- ☐ 4 3 2 1
- ☐ 1 2 3 4
- ☐ 4 3 2 1 0

13. Quel est l'opérateur de différence en C :

- ☐ `<>`
- ☐ `!=`
- ☐ `!`
- ☐ $\neq$

14. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ rien
- ☐ `Majeur`
- ☐ `Mineur`
- ☐ `Mineur`
  `Majeur`

15. Le code suivant :

```
int somme = 0;
int i;
for (i = 1; i < 4; i = i + 1)
{
  somme = somme + i;
}
printf("%d", somme);
```

affichera :

- ☐ 0
- ☐ 42
- ☐ 6
- ☐ 1

16. 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
- ☐ 0
- ☐ 4
- ☐ 8

17. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :

- ☐ tour à tour, un petit peu à chaque fois
- ☐ chacun son tour, après que le processus précédent a terminé
- ☐ en parallèle, chacun dans un registre
- ☐ tous ensemble

18. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#include <studio.h>`
- ☐ `#include <stdio.h>`
- ☐ `#appart <stdlib.h>`
- ☐ `#include <studlib.h>`

19. 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 bus explose
- ☐ la case mémoire 8 contiendra 0
- ☐ la case mémoire 8 contiendra 16
- ☐ le terminal affiche 8

20. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 8 2
- ☐ 8 6 4 2
- ☐ 0 2 4 6 8
- ☐ 8 6 4 2 0

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Un fichier source est :
   - ☐ un document qui doit être protégé
   - ☐ un document de référence du système
   - ☐ un document illisible pour les humains
   - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
   - ☐ un fichier texte qui sera traduit en instructions processeur
2. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `int tab[] = 5;`
   - ☐ `int[] new tableau(5);`
   - ☐ `int toto[taille=5];`
   - ☐ `char tableau[5];`
   - ☐ `int toto[5];`
3. 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=%d et y=%d\n",x y);`
   - ☐ `printf("x=%d et y=%d\n",x,y);`
   - ☐ `printf("x=%x et y=%y\n");`
4. 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 1 2 0 1 2`
   - ☐ `0 1 0 1 0 1 0 1`
   - ☐ `1 2 1 2 3`
   - ☐ `0 0 0 1 1 1`
5. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :
   - ☐ les fichiers du disque
   - ☐ certaines données de la mémoire de travail
   - ☐ en temps d'accès
   - ☐ des processus
6. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
   - ☐ `mkdir TP4`
   - ☐ `new TP4`
   - ☐ `yppasswd`
   - ☐ `kwrite TP4`
7. 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
8. Un registre du processeur est :
   - ☐ une gamme de fréquence de fonctionnement du processeur
   - ☐ un composant qui contient la liste des fichiers du système
   - ☐ une unité de calcul spécialisée de l'ordinateur
   - ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs
9. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
   - ☐ retourner un bloc
   - ☐ répéter un bloc tant qu'une condition est vérifée
   - ☐ mettre les blocs en séquence les uns à la suite des autres
   - ☐ sélectionner entre deux blocs à l'aide d'une condition
10. 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 = 0`
    - ☐ `j = 4`
    - ☐ `j = %d`
11. Dans la commande gcc, l'option `-Wall` signifie :
    - ☐ que l'on veut voir tous les avertissements
    - ☐ qu'on veut changer alétoirement de fond d'écran
    - ☐ qu'il faut indenter le fichier source
    - ☐ qu'il faut lancer un déboggueur
12. Le bus système sert à :
    - ☐ Arriver à l'heure en cours
    - ☐ Écrire des données sur le dique dur
    - ☐ transporter les processus du tourniquet au processeur
    - ☐ Transférer des données et intructions entre processeur et mémoire

13. 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
14. L'ordonnancement par tourniquet permet :
    - ☐ d'afficher des ronds colorés à l'écran
    - ☐ de ne pas perdre de temps avec la commutation de contexte
    - ☐ d'entretenir l'illusion que les processus tournent en parallèle
    - ☐ de doubler la mémoire disponible
15. Si cette erreur apparaît à la compilation : `Undefined symbols :"_prinft"` ou `référence indéfinie vers « prinft »` que doit-on chercher dans le programme ?
    - ☐ une faute de frappe dans un appel de fonction
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ un caractère interdit en C
    - ☐ une variable non déclarée
16. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
    - ☐ `#include <studlib.h>`
    - ☐ `#include <studio.h>`
    - ☐ `#appart <stdlib.h>`
    - ☐ `#include <stdio.h>`
17. Le code suivant :

```
int i;
for (i = 1; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

    affichera :
    - ☐ 1 2 3 4
    - ☐ 0 1 2 3 4
    - ☐ 4 3 2 1
    - ☐ 4 3 2 1 0
18. Le code suivant :

```
int i;
for (i = 4; i > 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

    affichera :
    - ☐ 0 1 2 3
    - ☐ 4 3 2 1 0
    - ☐ 0 1 2 3 4
    - ☐ 4 3 2 1
19. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

    affichera :
    - ☐ `Mineur`
    - ☐ rien
    - ☐ `Mineur`
`Majeur`
    - ☐ `Majeur`
20. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ créer un répertoire
    - ☐ ouvrir un fichier texte
    - ☐ créer un fichier texte
    - ☐ changer de répertoire courant

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Une *segmentation fault* est une erreur qui survient lorsque :
   - ☐ la division du programme en zones homogènes échoue
   - ☐ 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'accèder à une partie de la mémoire qui ne lui est pas réservée
   - ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal

2. 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=1;i<5;i=i+1)`
   - ☐ `for(i=0;i<=5;i=i+1)`

3. Sous unix (ou linux), la commande `ls` permet de :
   - ☐ afficher le contenu d'un fichier texte
   - ☐ compiler un programme
   - ☐ voir des clips musicaux
   - ☐ afficher la liste de fichiers contenus dans un répertoire

4. 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'analyse harmonique
   - ☐ l'analyse sémantique
   - ☐ l'édition de liens
   - ☐ l'analyse des entrées clavier

5. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
   - ☐ `#define taille = 3`
   - ☐ `#define taille = N`
   - ☐ `#define N 3`
   - ☐ `#define N =  3`

6. Quels calculs peut-on programmer 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
   - ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée
   - ☐ certains programmes sont de vrais plats de spaghetti

7. Le bus système sert à :
   - ☐ Arriver à l'heure en cours
   - ☐ transporter les processus du tourniquet au processeur
   - ☐ Transférer des données et intructions entre processeur et mémoire
   - ☐ Écrire des données sur le dique dur

8. Un bit est :
   - ☐ la longueur d'un mot mémoire
   - ☐ un chiffre binaire (0 ou 1)
   - ☐ l'instruction qui met fin à un programme
   - ☐ un battement d'horloge processeur

9. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

   affichera :
   - ☐ `Mineur`
   - ☐ `Mineur`
`Majeur`
   - ☐ `Majeur`
   - ☐ rien

10. 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  }
```

    - ☐ la variable x vaut 16
    - ☐ le programme affiche x
    - ☐ la variable x vaut $-\frac{1}{2}$
    - ☐ le programme affiche `****`

11. 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 = 5`
    - ☐ `j = 4`
    - ☐ `j = 0`
    - ☐ `j = %d`

12. Dans la commande gcc, l'option `-Wall` signifie :
    - ☐ qu'on veut changer alétoirement de fond d'écran
    - ☐ qu'il faut indenter le fichier source
    - ☐ que l'on veut voir tous les avertissements
    - ☐ qu'il faut lancer un déboggueur
13. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
    - ☐ en parallèle, chacun dans un registre
    - ☐ tour à tour, un petit peu à chaque fois
    - ☐ tous ensemble
    - ☐ chacun son tour, après que le processus précédent a terminé
14. Un fichier source est :
    - ☐ un document illisible pour les humains
    - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
    - ☐ un document de référence du système
    - ☐ un document qui doit être protégé
    - ☐ un fichier texte qui sera traduit en instructions processeur
15. Le code suivant :

```
int i;
for (i = 4; i >= 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

    affichera :
    - ☐ 1 2 3 4
    - ☐ 0 1 2 3 4
    - ☐ 4 3 2 1
    - ☐ 4 3 2 1 0
16. 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
    - ☐ 1 2 3 4
    - ☐ 4 3 2 1 0
17. 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
    - ☐ 0 1 2 3 4
18. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
    - ☐ mettre les blocs en séquence les uns à la suite des autres
    - ☐ retourner un bloc
    - ☐ répéter un bloc tant qu'une condition est vérifée
    - ☐ sélectionner entre deux blocs à l'aide d'une condition
19. 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
20. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

    affichera :
    - ☐ `Mineur`
`Majeur`
    - ☐ `Majeur`
    - ☐ `Mineur`
    - ☐ rien

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Les lignes

```
int i;
int x=0;
for(i=0,i<5,i=i+1)
{
  x=x+1;
}
```

☐ ne comportent aucune erreur
☐ comportent une erreur qui sera détectée au cours de l'édition de lien
☐ comportent une erreur qui ne sera pas détectée
☐ comportent une erreur qui sera détectée au cours de l'analyse syntaxique

2. 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 x vaut 5 et la variable y vaut 3
☐ la variable x vaut 3
☐ la variable y vaut 5
☐ le programme affiche "Faux"

3. 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 :

☐ 8
☐ 4
☐ 16
☐ 0

4. Quels calculs peut-on programmer 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
☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée
☐ certains programmes sont de vrais plats de spaghetti

5. Sous unix (ou linux), la commande `ls` permet de :

☐ compiler un programme
☐ afficher le contenu d'un fichier texte
☐ voir des clips musicaux
☐ afficher la liste de fichiers contenus dans un répertoire

6. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction

☐ `loop i;`
☐ `int loop n;`
☐ `int k;`
☐ `int %d;`

7. Un bit est :

☐ la longueur d'un mot mémoire
☐ un battement d'horloge processeur
☐ l'instruction qui met fin à un programme
☐ un chiffre binaire (0 ou 1)

8. 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 ?

☐ `** ** ** ** ** **`
☐ `***** **** *** ***`
☐ `** *** **** *****`
☐ `**** **** **** ****`

9. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

☐ 8 2
☐ 8 6 4 2 0
☐ 8 6 4 2
☐ 0 2 4 6 8

10. 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  }
```

☐ la variable y vaut 5

☐ la variable x vaut 0

☐ le programme affiche "Faux"

☐ la variable x vaut 5 et la variable y vaut 0

11. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
printf("Majeur\n");
```

affichera :

☐ `Majeur`

☐ `Mineur`
`Majeur`

☐ `Mineur`

☐ rien

12. 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 `x = 5`

☐ le terminal affiche "Faux"

☐ le terminal affiche `5`

☐ le terminal affiche `x = 2`

13. Sous unix (ou linux), la commande `cd` permet de :

☐ changer de répertoire courant

☐ détruire un fichier

☐ ouvir un bureau partagé (common desktop)

☐ jouer de la musique

☐ récupérer un programme arrêté avec la commande `ab`

14. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

☐ `Mineur`
`Majeur`

☐ `Mineur`

☐ `Majeur`

☐ rien

15. 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'analyse des entrées clavier

☐ l'analyse sémantique

☐ l'édition de liens

☐ l'analyse harmonique

16. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

☐ `#include <stdio.h>`

☐ `#include <studlib.h>`

☐ `#include <studio.h>`

☐ `#appart <stdlib.h>`

17. Le code suivant :

```
int i;
for (i = 0; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

☐ 4 3 2 1

☐ 0 1 2 3

☐ 4 3 2 1 0

☐ 0 1 2 3 4

18. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :

☐ sélectionner entre deux blocs à l'aide d'une condition

☐ répéter un bloc tant qu'une condition est vérifée

☐ retourner un bloc

☐ mettre les blocs en séquence les uns à la suite des autres

19. 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 :

☐ 4

☐ 16

☐ 8

☐ 0

20. Le code suivant :

```
int i;
for (i = 4; i >= 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

☐ 4 3 2 1

☐ 4 3 2 1 0

☐ 1 2 3 4

☐ 0 1 2 3 4

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. 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  }
```

   - ☐ la variable x vaut 0
   - ☐ la variable y vaut 5
   - ☐ la variable x vaut 5 et la variable y vaut 0
   - ☐ le programme affiche "Faux"

2. Le bus système sert à :
   - ☐ transporter les processus du tourniquet au processeur
   - ☐ Transférer des données et intructions entre processeur et mémoire
   - ☐ Arriver à l'heure en cours
   - ☐ Écrire des données sur le dique dur

3. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
printf("Majeur\n");
```

   affichera :
   - ☐ `Mineur`
   - ☐ `Majeur`
   - ☐ `Mineur`
`Majeur`
   - ☐ rien

4. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
   - ☐ `loop i;`
   - ☐ `int k;`
   - ☐ `int loop n;`
   - ☐ `int %d;`

5. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

   affichera :
   - ☐ `Mineur`
   - ☐ `Majeur`
   - ☐ `Mineur`
`Majeur`
   - ☐ rien

6. Si cette erreur apparaît à la compilation :
   `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
   - ☐ un point-virgule en trop
   - ☐ une accolade en trop
   - ☐ un point-virgule manquant
   - ☐ une accolade manquante

7. 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 0
   - ☐ le bus explose
   - ☐ la case mémoire 8 contiendra 16

8. 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=0;i<5;i=i+1)`
   - ☐ `for(i=0;i<=5;i=i+1)`
   - ☐ `for(i=1;i<=5;i=i+1)`
   - ☐ `for(i=1;i<5;i=i+1)`

9. 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 ?
   - ☐ `1 2 3 1 2`
   - ☐ `0 1 2 0 1 2`
   - ☐ `0 1 0 1 0 1`
   - ☐ `0 0 1 1 2 2`

10. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ rien
- ☐ `Mineur`
  `Majeur`
- ☐ `Mineur`
- ☐ `Majeur`

11. Le code suivant :

```
int i;
for (i = 0; i < 7; i = i + 2)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 0 2 4 6
- ☐ 0 2 4 6 8
- ☐ 0 1 2 3 4 5 6 7
- ☐ 0 1 2 3 4 5 6

12. 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 = 5`
- ☐ `j = 4`
- ☐ `j = %d`
- ☐ `j = 0`

13. 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=%d et y=%d\n",x y);`
- ☐ `printf("x=%x et y=%y\n");`
- ☐ `printf("x=%d et y=%d\n",x,y);`

14. 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 1 0 1 0 1 0 1`
- ☐ `0 0 0 1 1 1`
- ☐ `1 2 1 2 3`
- ☐ `0 1 2 0 1 2`

15. 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 x vaut 5 et la variable y vaut 3
- ☐ le programme affiche "Faux"
- ☐ la variable x vaut 3
- ☐ la variable y vaut 5

16. 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

17. Une *segmentation fault* est une erreur qui survient lorsque :

- ☐ le programme tente d'accèder à une partie de la mémoire qui ne lui est pas réservée
- ☐ la division du programme en zones homogènes échoue
- ☐ 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

18. Pour compiler un programme `prog.c`, on utilise la ligne de commande :

- ☐ `gcc prog.exe -Wall -o prog.c`
- ☐ `gcc prog.c -o -Wall prog.exe`
- ☐ `gcc -Wall prog.c -o prog.exe`
- ☐ `gcc -Wall prog.exe -o prog.c`

19. 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

20. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :

- ☐ `kwrite TP4`
- ☐ `new TP4`
- ☐ `yppasswd`
- ☐ `mkdir TP4`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. 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 = 4`
- ☐ `j = 0`
- ☐ `j = 5`
- ☐ `j = %d`

2. 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'analyse harmonique
   - ☐ l'analyse sémantique
   - ☐ l'édition de liens
   - ☐ l'analyse des entrées clavier
3. Une *segmentation fault* est une erreur qui survient lorsque :
   - ☐ le programme tente d'accèder à une partie de la mémoire qui ne lui est pas réservée
   - ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
   - ☐ la division du programme en zones homogènes échoue
   - ☐ 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
4. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Majeur`
- ☐ rien
- ☐ `Mineur`
  `Majeur`
- ☐ `Mineur`

5. 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 `5`
- ☐ le terminal affiche `x = 5`
- ☐ le terminal affiche `x = 2`

6. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
   - ☐ `gcc -Wall prog.c -o prog.exe`
   - ☐ `gcc prog.exe -Wall -o prog.c`
   - ☐ `gcc prog.c -o -Wall prog.exe`
   - ☐ `gcc -Wall prog.exe -o prog.c`
7. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
   - ☐ `mkdir TP4`
   - ☐ `yppasswd`
   - ☐ `kwrite TP4`
   - ☐ `new TP4`
8. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ init
   - ☐ main
   - ☐ begin
   - ☐ include
9. Un registre du processeur est :
   - ☐ un composant qui contient la liste des fichiers du système
   - ☐ une gamme de fréquence de fonctionnement du processeur
   - ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs
   - ☐ une unité de calcul spécialisée de l'ordinateur
10. Quels calculs peut-on programmer en programmation structurée ?
    - ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine
    - ☐ 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
    - ☐ certains programmes sont de vrais plats de spaghetti

11. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
printf("Majeur\n");
```

affichera :

- ☐ `Mineur`
  `Majeur`
- ☐ rien
- ☐ `Mineur`
- ☐ `Majeur`

12. 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
- ☐ 8
- ☐ 4
- ☐ 16

13. Sous unix (ou linux), la commande `ls` permet de :

- ☐ voir des clips musicaux
- ☐ afficher le contenu d'un fichier texte
- ☐ compiler un programme
- ☐ afficher la liste de fichiers contenus dans un répertoire

14. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :

- ☐ retourner un bloc
- ☐ sélectionner entre deux blocs à l'aide d'une condition
- ☐ mettre les blocs en séquence les uns à la suite des autres
- ☐ répéter un bloc tant qu'une condition est vérifée

15. Le code suivant :

```
int i;
for (i = 0; i < 7; i = i + 2)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 0 2 4 6
- ☐ 0 1 2 3 4 5 6 7
- ☐ 0 1 2 3 4 5 6
- ☐ 0 2 4 6 8

16. 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  }
```

- ☐ la variable x vaut 5 et la variable y vaut 0
- ☐ la variable x vaut 0
- ☐ la variable y vaut 5
- ☐ le programme affiche "Faux"

17. Un fichier source est :

- ☐ un fichier texte qui sera traduit en instructions processeur
- ☐ un document qui doit être protégé
- ☐ un document illisible pour les humains
- ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
- ☐ un document de référence du système

18. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#appart <stdlib.h>`
- ☐ `#include <studlib.h>`
- ☐ `#include <stdio.h>`
- ☐ `#include <studio.h>`

19. 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

20. 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)`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Le code suivant :

```
int i;
for (i = 1; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 1 2 3 4
- ☐ 0 1 2 3 4
- ☐ 4 3 2 1 0
- ☐ 4 3 2 1

2. Le code suivant :

```
int i;
for (i = 0; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 0 1 2 3 4
- ☐ 0 1 2 3
- ☐ 4 3 2 1 0
- ☐ 4 3 2 1

3. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Mineur`
`Majeur`
- ☐ `Mineur`
- ☐ rien
- ☐ `Majeur`

4. Un registre du processeur est :
   - ☐ un composant qui contient la liste des fichiers du système
   - ☐ une gamme de fréquence de fonctionnement du processeur
   - ☐ une unité de calcul spécialisée de l'ordinateur
   - ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs

5. 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
   - ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine
   - ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine
   - ☐ certains programmes sont de vrais plats de spaghetti

6. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse syntaxique
   - ☐ analyse lexicale
   - ☐ analyse harmonique
   - ☐ analyse sémantique

7. Le langage C est un langage
   - ☐ composé
   - ☐ interprété
   - ☐ compilé
   - ☐ lu, écrit, parlé

8. Si cette erreur apparaît à la compilation : `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
   - ☐ un point-virgule en trop
   - ☐ une accolade en trop
   - ☐ une accolade manquante
   - ☐ un point-virgule manquant

9. Le code suivant :

```
int i;
for (i = 0; i < 7; i = i + 2)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 0 2 4 6
- ☐ 0 2 4 6 8
- ☐ 0 1 2 3 4 5 6
- ☐ 0 1 2 3 4 5 6 7

10. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
    - ☐ `#appart <stdlib.h>`
    - ☐ `#include <studlib.h>`
    - ☐ `#include <studio.h>`
    - ☐ `#include <stdio.h>`

11. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
printf("Majeur\n");
```

affichera :

- ☐ rien
- ☐ `Majeur`
- ☐ `Mineur`
- ☐ `Mineur`
`Majeur`

12. Une *segmentation fault* est une erreur qui survient lorsque :
    - ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
    - ☐ 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'accèder à une partie de la mémoire qui ne lui est pas réservée
    - ☐ la division du programme en zones homogènes échoue

13. Si cette erreur apparaît à la compilation : `Undefined symbols :"_prinft"` ou `référence indéfinie vers` ≪ `prinft` ≫ que doit-on chercher dans le programme ?
    - ☐ une variable non déclarée
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ un caractère interdit en C
    - ☐ une faute de frappe dans un appel de fonction

14. Un bit est :
    - ☐ l'instruction qui met fin à un programme
    - ☐ un battement d'horloge processeur
    - ☐ la longueur d'un mot mémoire
    - ☐ un chiffre binaire (0 ou 1)

15. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
    - ☐ `int k;`
    - ☐ `int loop n;`
    - ☐ `int %d;`
    - ☐ `loop i;`

16. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ ouvrir un fichier texte
    - ☐ créer un fichier texte
    - ☐ changer de répertoire courant
    - ☐ créer un répertoire

17. 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  }
```

    - ☐ la variable y vaut 5
    - ☐ la variable x vaut 0
    - ☐ le programme affiche "Faux"
    - ☐ la variable x vaut 5 et la variable y vaut 0

18. 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

19. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
    - ☐ retourner un bloc
    - ☐ sélectionner entre deux blocs à l'aide d'une condition
    - ☐ répéter un bloc tant qu'une condition est vérifée
    - ☐ mettre les blocs en séquence les uns à la suite des autres

20. Le code suivant :

```
int somme = 0;
int i;
for (i = 1; i < 4; i = i + 1)
{
  somme = somme + i;
}
printf("%d", somme);
```

    affichera :
    - ☐ 6
    - ☐ 1
    - ☐ 0
    - ☐ 42

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ init
   - ☐ main
   - ☐ include
   - ☐ begin
2. Sur unix (ou linux), la commande `mkdir` permet de :
   - ☐ changer de répertoire courant
   - ☐ ouvrir un fichier texte
   - ☐ créer un répertoire
   - ☐ créer un fichier texte
3. Le code suivant :

```
int i;
for (i = 4; i > 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :
   - ☐ 4 3 2 1
   - ☐ 0 1 2 3 4
   - ☐ 4 3 2 1 0
   - ☐ 0 1 2 3
4. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
   - ☐ chacun son tour, après que le processus précédent a terminé
   - ☐ tour à tour, un petit peu à chaque fois
   - ☐ tous ensemble
   - ☐ en parallèle, chacun dans un registre
5. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse sémantique
   - ☐ analyse syntaxique
   - ☐ analyse harmonique
   - ☐ analyse lexicale
6. 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
7. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `char tableau[5];`
   - ☐ `int toto[taille=5];`
   - ☐ `int tab[] = 5;`
   - ☐ `int[] new tableau(5);`
   - ☐ `int toto[5];`
8. Dans la commande gcc, l'option `-Wall` signifie :
   - ☐ qu'il faut lancer un déboggueur
   - ☐ qu'il faut indenter le fichier source
   - ☐ qu'on veut changer alétoirement de fond d'écran
   - ☐ que l'on veut voir tous les avertissements
9. Le bus système sert à :
   - ☐ Écrire des données sur le dique dur
   - ☐ Transférer des données et intructions entre processeur et mémoire
   - ☐ transporter les processus du tourniquet au processeur
   - ☐ Arriver à l'heure en cours
10. 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 x
    - ☐ la variable x vaut 16
    - ☐ le programme affiche ****
    - ☐ la variable x vaut $-\frac{1}{2}$
11. Le code suivant :

```
int i;
for (i = 4; i >= 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

    affichera :
    - ☐ 0 1 2 3 4
    - ☐ 4 3 2 1
    - ☐ 4 3 2 1 0
    - ☐ 1 2 3 4
12. 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>`
13. 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 :
    - ☐ 3
    - ☐ 6
    - ☐ 20
    - ☐ 16

14. 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 = 0`
- ☐ `j = 5`
- ☐ `j = 4`
- ☐ `j = %d`

15. Quels calculs peut-on programmer en programmation structurée ?
    - ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine
    - ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée
    - ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine
    - ☐ certains programmes sont de vrais plats de spaghetti

16. 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  }
```

- ☐ la variable x vaut 0
- ☐ le programme affiche "Faux"
- ☐ la variable y vaut 5
- ☐ la variable x vaut 5 et la variable y vaut 0

17. Les lignes

```
int i;
int x=0;
for(i=0,i<5,i=i+1)
{
  x=x+1;
}
```

- ☐ comportent une erreur qui sera détectée au cours de l'analyse syntaxique
- ☐ ne comportent aucune erreur
- ☐ comportent une erreur qui sera détectée au cours de l'édition de lien
- ☐ comportent une erreur qui ne sera pas détectée

18. Le code suivant :

```
int somme = 0;
int i;
for (i = 1; i < 4; i = i + 1)
{
  somme = somme + i;
}
printf("%d", somme);
```

affichera :

- ☐ 6
- ☐ 0
- ☐ 42
- ☐ 1

19. Un bit est :
    - ☐ l'instruction qui met fin à un programme
    - ☐ la longueur d'un mot mémoire
    - ☐ un chiffre binaire (0 ou 1)
    - ☐ un battement d'horloge processeur

20. Le langage C est un langage
    - ☐ interprété
    - ☐ lu, écrit, parlé
    - ☐ composé
    - ☐ compilé

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. 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

2. 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 ?
   - ☐ 1 2 1 2 3
   - ☐ 0 1 2 0 1 2
   - ☐ 0 1 0 1 0 1 0 1
   - ☐ 0 0 0 1 1 1

3. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :
   - ☐ 8 6 4 2
   - ☐ 0 2 4 6 8
   - ☐ 8 6 4 2 0
   - ☐ 8 2

4. Un fichier source est :
   - ☐ un document illisible pour les humains
   - ☐ un document de référence du système
   - ☐ un fichier texte qui sera traduit en instructions processeur
   - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
   - ☐ un document qui doit être protégé

5. 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 2 0 1 2
   - ☐ 1 2 3 1 2
   - ☐ 0 1 0 1 0 1
   - ☐ 0 0 1 1 2 2

6. 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
   - ☐ 4
   - ☐ 8

7. Le code suivant :

```
int i;
for (i = 0; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :
   - ☐ 4 3 2 1 0
   - ☐ 0 1 2 3
   - ☐ 4 3 2 1
   - ☐ 0 1 2 3 4

8. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
   - ☐ tour à tour, un petit peu à chaque fois
   - ☐ en parallèle, chacun dans un registre
   - ☐ tous ensemble
   - ☐ chacun son tour, après que le processus précédent a terminé

9. Un bit est :
   - ☐ l'instruction qui met fin à un programme
   - ☐ un chiffre binaire (0 ou 1)
   - ☐ un battement d'horloge processeur
   - ☐ la longueur d'un mot mémoire

10. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
    - ☐ `#define taille = 3`
    - ☐ `#define N 3`
    - ☐ `#define N =  3`
    - ☐ `#define taille = N`

11. 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
- ☐ 0 1 2 3 4
- ☐ 4 3 2 1

12. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Mineur`
- ☐ rien
- ☐ `Majeur`
- ☐ `Mineur`
  `Majeur`

13. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Mineur`
  `Majeur`
- ☐ `Mineur`
- ☐ `Majeur`
- ☐ rien

14. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
    - ☐ `#include <stdio.h>`
    - ☐ `#include <studio.h>`
    - ☐ `#appart <stdlib.h>`
    - ☐ `#include <studlib.h>`
15. Le langage C est un langage
    - ☐ interprété
    - ☐ compilé
    - ☐ lu, écrit, parlé
    - ☐ composé
16. 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 bus explose
    - ☐ la case mémoire 8 contiendra 0
    - ☐ le terminal affiche 8
    - ☐ la case mémoire 8 contiendra 16
17. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ créer un fichier texte
    - ☐ changer de répertoire courant
    - ☐ créer un répertoire
    - ☐ ouvrir un fichier texte
18. Un programme en langage C doit comporter une et une seule définition de la fonction :
    - ☐ begin
    - ☐ init
    - ☐ main
    - ☐ include
19. Les lignes

```
int i;
int x=0;
for(i=0,i<5,i=i+1)
{
  x=x+1;
}
```

    - ☐ ne comportent aucune erreur
    - ☐ 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
    - ☐ comportent une erreur qui ne sera pas détectée
20. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :
    - ☐ en temps d'accès
    - ☐ certaines données de la mémoire de travail
    - ☐ les fichiers du disque
    - ☐ des processus

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Un registre du processeur est :
   - ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs
   - ☐ un composant qui contient la liste des fichiers du système
   - ☐ une unité de calcul spécialisée de l'ordinateur
   - ☐ une gamme de fréquence de fonctionnement du processeur

2. 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 bus explose
   - ☐ le terminal affiche 8
   - ☐ la case mémoire 8 contiendra 0
   - ☐ la case mémoire 8 contiendra 16

3. Un bit est :
   - ☐ un battement d'horloge processeur
   - ☐ l'instruction qui met fin à un programme
   - ☐ un chiffre binaire (0 ou 1)
   - ☐ la longueur d'un mot mémoire

4. 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=%d et y=%d\n",x,y);`
   - ☐ `printf("x=%d et y=%d\n,x,y");`
   - ☐ `printf("x=%x et y=%y\n");`

5. L'ordonnancement par tourniquet permet :
   - ☐ de ne pas perdre de temps avec la commutation de contexte
   - ☐ de doubler la mémoire disponible
   - ☐ d'afficher des ronds colorés à l'écran
   - ☐ d'entretenir l'illusion que les processus tournent en parallèle

6. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
printf("Majeur\n");
```

   affichera :
   - ☐ `Majeur`
   - ☐ `Mineur`
   - ☐ `Mineur`
`Majeur`
   - ☐ rien

7. Les lignes

```
int i;
int x=0;
for(i=0,i<5,i=i+1)
{
  x=x+1;
}
```

   - ☐ comportent une erreur qui sera détectée au cours de l'édition de lien
   - ☐ comportent une erreur qui ne sera pas détectée
   - ☐ ne comportent aucune erreur
   - ☐ comportent une erreur qui sera détectée au cours de l'analyse syntaxique

8. 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`
   - ☐ `1 2 3 1 2`
   - ☐ `0 1 2 0 1 2`

9. 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  }
```

   - ☐ la variable x vaut 5 et la variable y vaut 0
   - ☐ la variable y vaut 5
   - ☐ la variable x vaut 0
   - ☐ le programme affiche "Faux"

10. Le bus système sert à :
    - ☐ Écrire des données sur le dique dur
    - ☐ Arriver à l'heure en cours
    - ☐ transporter les processus du tourniquet au processeur
    - ☐ Transférer des données et intructions entre processeur et mémoire

11. Sous unix (ou linux), la commande `cd` permet de :
    - ☐ ouvir un bureau partagé (common desktop)
    - ☐ détruire un fichier
    - ☐ jouer de la musique
    - ☐ récupérer un programme arrêté avec la commande `ab`
    - ☐ changer de répertoire courant

12. 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 :
    - ☐ 15
    - ☐ 6
    - ☐ 10
    - ☐ 0

13. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

    affichera :
    - ☐ `Mineur`
`Majeur`
    - ☐ `Mineur`
    - ☐ `Majeur`
    - ☐ rien

14. 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 x vaut 3
    - ☐ la variable x vaut 5 et la variable y vaut 3
    - ☐ la variable y vaut 5
    - ☐ le programme affiche "Faux"

15. 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=0;i<5;i=i+1)`
    - ☐ `for(i=1;i<=5;i=i+1)`
    - ☐ `for(i=0;i<=5;i=i+1)`
    - ☐ `for(i=1;i<5;i=i+1)`

16. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
    - ☐ `new TP4`
    - ☐ `mkdir TP4`
    - ☐ `yppasswd`
    - ☐ `kwrite TP4`

17. 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 1 2 0 1 2 3`
    - ☐ `0 1 2 3 0 1 2`
    - ☐ `0 1 2 0 1 2`
    - ☐ `0 0 1 1 2 2 3`

18. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ ouvrir un fichier texte
    - ☐ créer un répertoire
    - ☐ changer de répertoire courant
    - ☐ créer un fichier texte

19. 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
    - ☐ les fichiers du disque
    - ☐ des processus

20. Le code suivant :

```
int i;
for (i = 0; i < 7; i = i + 2)
{
    printf("%d ", i);
}
printf("\n");
```

    affichera :
    - ☐ 0 2 4 6 8
    - ☐ 0 1 2 3 4 5 6 7
    - ☐ 0 1 2 3 4 5 6
    - ☐ 0 2 4 6

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
   - ☐ sélectionner entre deux blocs à l'aide d'une condition
   - ☐ retourner un bloc
   - ☐ mettre les blocs en séquence les uns à la suite des autres
   - ☐ répéter un bloc tant qu'une condition est vérifée
2. 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`
3. Un bit est :
   - ☐ un chiffre binaire (0 ou 1)
   - ☐ la longueur d'un mot mémoire
   - ☐ un battement d'horloge processeur
   - ☐ l'instruction qui met fin à un programme
4. 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  }
```

   - ☐ la variable x vaut 16
   - ☐ le programme affiche x
   - ☐ la variable x vaut $-\frac{1}{2}$
   - ☐ le programme affiche `****`
5. 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 0
   - ☐ le bus explose
   - ☐ la case mémoire 8 contiendra 16
6. 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 `x = 5`
   - ☐ le terminal affiche `x = 2`
   - ☐ le terminal affiche "Faux"
   - ☐ le terminal affiche `5`
7. 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=%d et y=%d\n",x,y);`
   - ☐ `printf("x=%d et y=%d\n,x,y");`
   - ☐ `printf("x=%x et y=%y\n");`
8. 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 1 2 0 1 2`
   - ☐ `1 2 1 2 3`
   - ☐ `0 1 0 1 0 1 0 1`
   - ☐ `0 0 0 1 1 1`
9. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

   affichera :
   - ☐ `Majeur`
   - ☐ `Mineur`
     `Majeur`
   - ☐ `Mineur`
   - ☐ rien
10. 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 1 2 0 1 2

☐ 0 0 1 1 2 2 3

☐ 0 1 2 3 0 1 2

☐ 0 1 2 0 1 2 3

11. Un registre du processeur est :

☐ un composant qui contient la liste des fichiers du système

☐ une gamme de fréquence de fonctionnement du processeur

☐ une unité de calcul spécialisée de l'ordinateur

☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs

12. Le code suivant :

```
int i;
for (i = 4; i > 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

☐ 4 3 2 1

☐ 0 1 2 3

☐ 4 3 2 1 0

☐ 0 1 2 3 4

13. 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 2 0 1 2

☐ 1 2 3 1 2

☐ 0 1 0 1 0 1

☐ 0 0 1 1 2 2

14. Dans la commande gcc, l'option `-Wall` signifie :

☐ qu'il faut lancer un déboggueur

☐ qu'on veut changer alétoirement de fond d'écran

☐ que l'on veut voir tous les avertissements

☐ qu'il faut indenter le fichier source

15. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
printf("Majeur\n");
```

affichera :

☐ `Mineur`
`Majeur`

☐ `Mineur`

☐ rien

☐ `Majeur`

16. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :

☐ en temps d'accès

☐ certaines données de la mémoire de travail

☐ les fichiers du disque

☐ des processus

17. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

☐ `Mineur`
`Majeur`

☐ `Mineur`

☐ rien

☐ `Majeur`

18. 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 :

☐ 16

☐ 0

☐ 8

☐ 4

19. Sous unix (ou linux), la commande `ls` permet de :

☐ compiler un programme

☐ afficher le contenu d'un fichier texte

☐ voir des clips musicaux

☐ afficher la liste de fichiers contenus dans un répertoire

20. 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  }
```

☐ le programme affiche "Faux"

☐ la variable x vaut 3

☐ la variable y vaut 5

☐ la variable x vaut 5 et la variable y vaut 3

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. 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 = 5`
- ☐ `j = %d`
- ☐ `j = 0`
- ☐ `j = 4`

2. 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'édition de lien
- ☐ comportent une erreur qui sera détectée au cours de l'analyse syntaxique
- ☐ ne comportent aucune erreur

3. 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'analyse des entrées clavier
   - ☐ l'analyse sémantique
   - ☐ l'analyse harmonique
   - ☐ l'édition de liens

4. 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  }
```

- ☐ la variable x vaut $-\frac{1}{2}$
- ☐ le programme affiche x
- ☐ le programme affiche `****`
- ☐ la variable x vaut 16

5. 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 `x = 2`
- ☐ le terminal affiche `x = 5`
- ☐ le terminal affiche "Faux"
- ☐ le terminal affiche `5`

6. Sur unix (ou linux), la commande `mkdir` permet de :
   - ☐ créer un répertoire
   - ☐ créer un fichier texte
   - ☐ ouvrir un fichier texte
   - ☐ changer de répertoire courant

7. Un bit est :
   - ☐ un battement d'horloge processeur
   - ☐ un chiffre binaire (0 ou 1)
   - ☐ la longueur d'un mot mémoire
   - ☐ l'instruction qui met fin à un programme

8. Si cette erreur apparaît à la compilation :
`error: expected ';' before '}' token` que doit-on chercher dans le programme ?
   - ☐ une accolade manquante
   - ☐ un point-virgule en trop
   - ☐ un point-virgule manquant
   - ☐ une accolade en trop

9. Sous unix (ou linux), la commande `ls` permet de :
   - ☐ voir des clips musicaux
   - ☐ afficher le contenu d'un fichier texte
   - ☐ compiler un programme
   - ☐ afficher la liste de fichiers contenus dans un répertoire

10. Le code suivant :

```
int i;
for (i = 0; i < 7; i = i + 2)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :
   - ☐ 0 2 4 6
   - ☐ 0 2 4 6 8
   - ☐ 0 1 2 3 4 5 6 7
   - ☐ 0 1 2 3 4 5 6

11. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#include <stdio.h>`
   - ☐ `#include <studlib.h>`
   - ☐ `#appart <stdlib.h>`
   - ☐ `#include <studio.h>`

12. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
   - ☐ `int %d;`
   - ☐ `int loop n;`
   - ☐ `loop i;`
   - ☐ `int k;`

13. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ rien
- ☐ `Majeur`
- ☐ `Mineur`
  `Majeur`
- ☐ `Mineur`

14. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.

- ☐ `#define N 3`
- ☐ `#define taille = N`
- ☐ `#define N =  3`
- ☐ `#define taille = 3`

15. 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
- ☐ 0 1 2 3 4
- ☐ 4 3 2 1

16. Si cette erreur apparaît à la compilation :
`Undefined symbols :"_prinft"` ou
`référence indéfinie vers` « `prinft` » que doit-on chercher dans le programme ?

- ☐ une variable non déclarée
- ☐ une directive préprocesseur `#include` manquante
- ☐ un caractère interdit en C
- ☐ une faute de frappe dans un appel de fonction

17. Un programme en langage C doit comporter une et une seule définition de la fonction :

- ☐ begin
- ☐ init
- ☐ include
- ☐ main

18. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ rien
- ☐ `Mineur`
- ☐ `Majeur`
- ☐ `Mineur`
  `Majeur`

19. Quel est l'opérateur de différence en C :

- ☐ `!=`
- ☐ ≠
- ☐ `<>`
- ☐ `!`

20. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :

- ☐ analyse lexicale
- ☐ analyse syntaxique
- ☐ analyse harmonique
- ☐ analyse sémantique

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#include <studio.h>`
   - ☐ `#appart <stdlib.h>`
   - ☐ `#include <stdio.h>`
   - ☐ `#include <studlib.h>`
2. Quel est l'opérateur de différence en C :
   - ☐ `!=`
   - ☐ `<>`
   - ☐ $\neq$
   - ☐ `!`
3. 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 7
   - ☐ 0 1 2 3 4 5 6
   - ☐ 0 2 4 6 8
   - ☐ 0 2 4 6
4. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :
   - ☐ des processus
   - ☐ les fichiers du disque
   - ☐ en temps d'accès
   - ☐ certaines données de la mémoire de travail
5. 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 16
   - ☐ le programme affiche x
   - ☐ la variable x vaut $-\frac{1}{2}$
6. Une *segmentation fault* est une erreur qui survient lorsque :
   - ☐ le programme tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
   - ☐ la division du programme en zones homogènes échoue
   - ☐ 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'accèder à une partie de la mémoire qui ne lui est pas réservée
7. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
   - ☐ 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
   - ☐ retourner un bloc
8. 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=0;i<=5;i=i+1)`
   - ☐ `for(i=1;i<=5;i=i+1)`
   - ☐ `for(i=1;i<5;i=i+1)`
   - ☐ `for(i=0;i<5;i=i+1)`
9. 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
   - ☐ 15
   - ☐ 10
   - ☐ 6
10. 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  }
```

    - ☐ la variable x vaut 5 et la variable y vaut 0
    - ☐ le programme affiche "Faux"
    - ☐ la variable y vaut 5
    - ☐ la variable x vaut 0
11. 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 = %d`
- ☐ `j = 5`
- ☐ `j = 0`

12. Si cette erreur apparaît à la compilation : `Undefined symbols :"_prinft"` ou `référence indéfinie vers` « `prinft` » que doit-on chercher dans le programme ?
    - ☐ une variable non déclarée
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ un caractère interdit en C
    - ☐ une faute de frappe dans un appel de fonction
13. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
    - ☐ `new TP4`
    - ☐ `mkdir TP4`
    - ☐ `kwrite TP4`
    - ☐ `yppasswd`
14. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 8 6 4 2
- ☐ 8 2
- ☐ 0 2 4 6 8
- ☐ 8 6 4 2 0

15. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Mineur`
- ☐ `Majeur`
- ☐ rien
- ☐ `Mineur`
  `Majeur`

16. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Mineur`
- ☐ `Mineur`
  `Majeur`
- ☐ `Majeur`
- ☐ rien

17. Un programme en langage C doit comporter une et une seule définition de la fonction :
    - ☐ init
    - ☐ main
    - ☐ include
    - ☐ begin
18. 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 `x = 2`
- ☐ le terminal affiche "Faux"
- ☐ le terminal affiche `x = 5`
- ☐ le terminal affiche `5`

19. Un fichier source est :
    - ☐ un fichier texte qui sera traduit en instructions processeur
    - ☐ un document illisible pour les humains
    - ☐ un document de référence du système
    - ☐ un document qui doit être protégé
    - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
20. Le code suivant :

```
int i;
for (i = 0; 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
- ☐ 0 1 2 3

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Le code suivant :

```
int i;
for (i = 1; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 1 2 3 4
- ☐ 4 3 2 1
- ☐ 0 1 2 3 4
- ☐ 4 3 2 1 0

2. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Mineur`
  `Majeur`
- ☐ rien
- ☐ `Mineur`
- ☐ `Majeur`

3. 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 :

- ☐ 20
- ☐ 3
- ☐ 16
- ☐ 6

4. 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"

5. Si cette erreur apparaît à la compilation : `Undefined symbols :"_prinft"` ou `référence indéfinie vers « prinft »` que doit-on chercher dans le programme ?

- ☐ une variable non déclarée
- ☐ une directive préprocesseur `#include` manquante
- ☐ une faute de frappe dans un appel de fonction
- ☐ un caractère interdit en C

6. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :

- ☐ les fichiers du disque
- ☐ des processus
- ☐ certaines données de la mémoire de travail
- ☐ en temps d'accès

7. Sur unix (ou linux), la commande `mkdir` permet de :

- ☐ changer de répertoire courant
- ☐ créer un répertoire
- ☐ créer un fichier texte
- ☐ ouvrir un fichier texte

8. L'ordonnancement par tourniquet permet :

- ☐ de ne pas perdre de temps avec la commutation de contexte
- ☐ de doubler la mémoire disponible
- ☐ d'afficher des ronds colorés à l'écran
- ☐ d'entretenir l'illusion que les processus tournent en parallèle

9. Un fichier source est :

- ☐ un document qui doit être protégé
- ☐ un document illisible pour les humains
- ☐ un document de référence du système
- ☐ un fichier texte qui sera traduit en instructions processeur
- ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur

10. 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 `5`
- ☐ le terminal affiche `x = 5`

11. 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=%d et y=%d\n",x y);`
- ☐ `printf("x=%x et y=%y\n");`
- ☐ `printf("x=%d et y=%d\n,x,y");`

12. Pour compiler un programme `prog.c`, on utilise la ligne de commande :

- ☐ `gcc -Wall prog.exe -o prog.c`
- ☐ `gcc -Wall prog.c -o prog.exe`
- ☐ `gcc prog.exe -Wall -o prog.c`
- ☐ `gcc prog.c -o -Wall prog.exe`

13. 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  }
```

- ☐ la variable x vaut 16
- ☐ la variable x vaut $-\frac{1}{2}$
- ☐ le programme affiche `****`
- ☐ le programme affiche x

14. Un bit est :

- ☐ un chiffre binaire (0 ou 1)
- ☐ la longueur d'un mot mémoire
- ☐ un battement d'horloge processeur
- ☐ l'instruction qui met fin à un programme

15. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#include <studlib.h>`
- ☐ `#include <stdio.h>`
- ☐ `#include <studio.h>`
- ☐ `#appart <stdlib.h>`

16. Un registre du processeur est :

- ☐ une gamme de fréquence de fonctionnement du processeur
- ☐ un composant qui contient la liste des fichiers du système
- ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs
- ☐ une unité de calcul spécialisée de l'ordinateur

17. 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
```

- ☐ la case mémoire 8 contiendra 16
- ☐ la case mémoire 8 contiendra 0
- ☐ le terminal affiche 8
- ☐ le bus explose

18. 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 = 0`
- ☐ `j = %d`
- ☐ `j = 4`
- ☐ `j = 5`

19. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Majeur`
- ☐ `Mineur`
  `Majeur`
- ☐ `Mineur`
- ☐ rien

20. Le code suivant :

```
int somme = 0;
int i;
for (i = 1; i < 4; i = i + 1)
{
  somme = somme + i;
}
printf("%d", somme);
```

affichera :

- ☐ 42
- ☐ 0
- ☐ 6
- ☐ 1

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Un fichier source est :
   - ☐ un document illisible pour les humains
   - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
   - ☐ un fichier texte qui sera traduit en instructions processeur
   - ☐ un document de référence du système
   - ☐ un document qui doit être protégé
2. Le langage C est un langage
   - ☐ lu, écrit, parlé
   - ☐ compilé
   - ☐ interprété
   - ☐ composé
3. Le code suivant :

```
int i;
for (i = 0; i < 7; i = i + 2)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :
   - ☐ 0 2 4 6
   - ☐ 0 1 2 3 4 5 6
   - ☐ 0 1 2 3 4 5 6 7
   - ☐ 0 2 4 6 8
4. Si cette erreur apparaît à la compilation : `Undefined symbols :"_prinft"` ou `référence indéfinie vers` « `prinft` » que doit-on chercher dans le programme ?
   - ☐ un caractère interdit en C
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une faute de frappe dans un appel de fonction
   - ☐ une variable non déclarée
5. 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=0;i<=5;i=i+1)`
   - ☐ `for(i=0;i<5;i=i+1)`
   - ☐ `for(i=1;i<=5;i=i+1)`
   - ☐ `for(i=1;i<5;i=i+1)`
6. Le code suivant :

```
int somme = 0;
int i;
for (i = 1; i < 4; i = i + 1)
{
  somme = somme + i;
}
printf("%d", somme);
```

   affichera :
   - ☐ 0
   - ☐ 42
   - ☐ 1
   - ☐ 6
7. 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 :
   - ☐ 0
   - ☐ 4
   - ☐ 16
   - ☐ 8
8. 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 :
   - ☐ 10
   - ☐ 0
   - ☐ 6
   - ☐ 15
9. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
   - ☐ `gcc prog.c -o -Wall prog.exe`
   - ☐ `gcc -Wall prog.c -o prog.exe`
   - ☐ `gcc -Wall prog.exe -o prog.c`
   - ☐ `gcc prog.exe -Wall -o prog.c`
10. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
    - ☐ `#include <stdio.h>`
    - ☐ `#appart <stdlib.h>`
    - ☐ `#include <studlib.h>`
    - ☐ `#include <studio.h>`
11. 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  }
```

    - ☐ le programme affiche "Faux"
    - ☐ la variable x vaut 5 et la variable y vaut 3
    - ☐ la variable x vaut 3
    - ☐ la variable y vaut 5

12. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
    - ☐ `mkdir TP4`
    - ☐ `kwrite TP4`
    - ☐ `new TP4`
    - ☐ `yppasswd`
13. Un registre du processeur est :
    - ☐ une unité de calcul spécialisée de l'ordinateur
    - ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs
    - ☐ un composant qui contient la liste des fichiers du système
    - ☐ une gamme de fréquence de fonctionnement du processeur
14. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
    - ☐ `#define taille = N`
    - ☐ `#define N 3`
    - ☐ `#define taille = 3`
    - ☐ `#define N =  3`
15. 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 = 0`
- ☐ `j = 5`
- ☐ `j = %d`

16. Le code suivant :

```
int i;
for (i = 4; i >= 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 4 3 2 1
- ☐ 4 3 2 1 0
- ☐ 1 2 3 4
- ☐ 0 1 2 3 4

17. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :
    - ☐ répéter un bloc tant qu'une condition est vérifée
    - ☐ mettre les blocs en séquence les uns à la suite des autres
    - ☐ sélectionner entre deux blocs à l'aide d'une condition
    - ☐ retourner un bloc
18. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 8 6 4 2 0
- ☐ 8 2
- ☐ 8 6 4 2
- ☐ 0 2 4 6 8

19. Le code suivant :

```
int i;
for (i = 4; i > 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 4 3 2 1 0
- ☐ 0 1 2 3
- ☐ 0 1 2 3 4
- ☐ 4 3 2 1

20. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :
    - ☐ en temps d'accès
    - ☐ les fichiers du disque
    - ☐ certaines données de la mémoire de travail
    - ☐ des processus

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Sous unix (ou linux), la commande `cd` permet de :
   - ☐ détruire un fichier
   - ☐ changer de répertoire courant
   - ☐ récupérer un programme arrêté avec la commande `ab`
   - ☐ jouer de la musique
   - ☐ ouvir un bureau partagé (common desktop)
2. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
   - ☐ tous ensemble
   - ☐ tour à tour, un petit peu à chaque fois
   - ☐ en parallèle, chacun dans un registre
   - ☐ chacun son tour, après que le processus précédent a terminé
3. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `int[] new tableau(5);`
   - ☐ `int toto[taille=5];`
   - ☐ `int tab[] = 5;`
   - ☐ `char tableau[5];`
   - ☐ `int toto[5];`
4. 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  }
```

   - ☐ la variable x vaut 5 et la variable y vaut 0
   - ☐ la variable x vaut 0
   - ☐ le programme affiche "Faux"
   - ☐ la variable y vaut 5
5. Un fichier source est :
   - ☐ un fichier texte qui sera traduit en instructions processeur
   - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
   - ☐ un document illisible pour les humains
   - ☐ un document de référence du système
   - ☐ un document qui doit être protégé
6. 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
   - ☐ ne comportent aucune erreur
   - ☐ 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
7. 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 1 2 0 1 2`
   - ☐ `0 1 2 0 1 2 3`
   - ☐ `0 0 1 1 2 2 3`
   - ☐ `0 1 2 3 0 1 2`
8. 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
   - ☐ 0 1 2 3 4
9. 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=%d et y=%d\n",x y);`
   - ☐ `printf("x=%d et y=%d\n",x,y);`
   - ☐ `printf("x=%x et y=%y\n");`
10. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
    - ☐ `int k;`
    - ☐ `loop i;`
    - ☐ `int loop n;`
    - ☐ `int %d;`
11. Dans la commande gcc, l'option `-Wall` signifie :
    - ☐ qu'on veut changer alétoirement de fond d'écran
    - ☐ qu'il faut lancer un déboggueur
    - ☐ qu'il faut indenter le fichier source
    - ☐ que l'on veut voir tous les avertissements
12. Si cette erreur apparaît à la compilation :
    `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
    - ☐ un point-virgule en trop
    - ☐ une accolade en trop
    - ☐ une accolade manquante
    - ☐ un point-virgule manquant

13. 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 :

- ☐ 10
- ☐ 0
- ☐ 6
- ☐ 15

14. 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=1;i<=5;i=i+1)`
- ☐ `for(i=0;i<=5;i=i+1)`

15. 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

16. 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
- ☐ 6
- ☐ 3
- ☐ 20

17. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Majeur`
- ☐ `Mineur`
- ☐ `Mineur`
  `Majeur`
- ☐ rien

18. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

- ☐ `#include <studlib.h>`
- ☐ `#include <stdio.h>`
- ☐ `#appart <stdlib.h>`
- ☐ `#include <studio.h>`

19. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.

- ☐ `#define taille = N`
- ☐ `#define N 3`
- ☐ `#define N =  3`
- ☐ `#define taille = 3`

20. 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 `x = 5`
- ☐ le terminal affiche `x = 2`
- ☐ le terminal affiche "Faux"
- ☐ le terminal affiche `5`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Un bit est :
   - ☐ un battement d'horloge processeur
   - ☐ l'instruction qui met fin à un programme
   - ☐ la longueur d'un mot mémoire
   - ☐ un chiffre binaire (0 ou 1)

2. 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
   - ☐ une faute de frappe dans un appel de fonction
   - ☐ une variable non déclarée
   - ☐ un caractère interdit en C

3. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#include <studio.h>`
   - ☐ `#include <studlib.h>`
   - ☐ `#include <stdio.h>`
   - ☐ `#appart <stdlib.h>`

4. Un registre du processeur est :
   - ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs
   - ☐ une unité de calcul spécialisée de l'ordinateur
   - ☐ un composant qui contient la liste des fichiers du système
   - ☐ une gamme de fréquence de fonctionnement du processeur

5. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :
   - ☐ 8 6 4 2
   - ☐ 0 2 4 6 8
   - ☐ 8 2
   - ☐ 8 6 4 2 0

6. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
   - ☐ `int %d;`
   - ☐ `int loop n;`
   - ☐ `loop i;`
   - ☐ `int k;`

7. 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)`

8. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
   - ☐ `#define N =  3`
   - ☐ `#define taille = 3`
   - ☐ `#define taille = N`
   - ☐ `#define N 3`

9. 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 2 0 1 2`
   - ☐ `1 2 3 1 2`
   - ☐ `0 0 1 1 2 2`
   - ☐ `0 1 0 1 0 1`

10. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :
   - ☐ rien
   - ☐ `Majeur`
   - ☐ `Mineur`
     `Majeur`
   - ☐ `Mineur`

11. 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 :
   - ☐ 10
   - ☐ 0
   - ☐ 6
   - ☐ 15

12. 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 `x = 2`

☐ le terminal affiche `5`

☐ le terminal affiche `x = 5`

☐ le terminal affiche "Faux"

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 = %d`

☐ `j = 5`

☐ `j = 0`

☐ `j = 4`

14. 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=%x et y=%y\n");`

☐ `printf("x=%d et y=%d\n,x,y");`

☐ `printf("x=%d et y=%d\n",x y);`

☐ `printf("x=%d et y=%d\n",x,y);`

15. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

☐ `Majeur`

☐ `Mineur`

☐ `Mineur`
`Majeur`

☐ rien

16. Un fichier source est :

☐ un document de référence du système

☐ un fichier texte qui sera traduit en instructions processeur

☐ un document qui doit être protégé

☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur

☐ un document illisible pour les humains

17. 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 2 4 6 8

☐ 0 1 2 3 4 5 6 7

☐ 0 2 4 6

18. Le code suivant :

```
int i;
for (i = 4; i > 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

☐ 4 3 2 1 0

☐ 0 1 2 3 4

☐ 0 1 2 3

☐ 4 3 2 1

19. L'ordonnancement par tourniquet permet :

☐ de doubler la mémoire disponible

☐ 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

20. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

☐ rien

☐ `Mineur`
`Majeur`

☐ `Mineur`

☐ `Majeur`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Le code suivant :

```
int i;
for (i = 1; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 1 2 3 4
- ☐ 0 1 2 3 4
- ☐ 4 3 2 1 0
- ☐ 4 3 2 1

2. 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  }
```

- ☐ le programme affiche "Faux"
- ☐ la variable x vaut 5 et la variable y vaut 3
- ☐ la variable x vaut 3
- ☐ la variable y vaut 5

3. 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 1 2 0 1 2`
- ☐ `1 2 1 2 3`
- ☐ `0 1 0 1 0 1 0 1`
- ☐ `0 0 0 1 1 1`

4. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Mineur`
  `Majeur`
- ☐ `Mineur`
- ☐ rien
- ☐ `Majeur`

5. Quels calculs peut-on programmer en programmation structurée ?
   - ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine
   - ☐ certains programmes sont de vrais plats de spaghetti
   - ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine
   - ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée

6. Dans la commande gcc, l'option `-Wall` signifie :
   - ☐ qu'il faut lancer un déboggueur
   - ☐ que l'on veut voir tous les avertissements
   - ☐ qu'on veut changer alétoirement de fond d'écran
   - ☐ qu'il faut indenter le fichier source

7. 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 ?

- ☐ `***** **** *** ***`
- ☐ `**** **** **** ****`
- ☐ `** ** ** ** ** **`
- ☐ `** *** **** *****`

8. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ begin
   - ☐ include
   - ☐ init
   - ☐ main

9. Si cette erreur apparaît à la compilation :
   `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
   - ☐ une accolade en trop
   - ☐ une accolade manquante
   - ☐ un point-virgule en trop
   - ☐ un point-virgule manquant

10. 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  }
```

- ☐ la variable x vaut 16
- ☐ le programme affiche ****
- ☐ le programme affiche x
- ☐ la variable x vaut $-\frac{1}{2}$

11. Le bus système sert à :
    - ☐ transporter les processus du tourniquet au processeur
    - ☐ Écrire des données sur le dique dur
    - ☐ Transférer des données et intructions entre processeur et mémoire
    - ☐ Arriver à l'heure en cours
12. Le langage C est un langage
    - ☐ compilé
    - ☐ interprété
    - ☐ composé
    - ☐ lu, écrit, parlé
13. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
    - ☐ en parallèle, chacun dans un registre
    - ☐ tous ensemble
    - ☐ chacun son tour, après que le processus précédent a terminé
    - ☐ tour à tour, un petit peu à chaque fois
14. 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 `5`
- ☐ le terminal affiche `x = 2`
- ☐ le terminal affiche "Faux"
- ☐ le terminal affiche `x = 5`

15. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :
    - ☐ en temps d'accès
    - ☐ certaines données de la mémoire de travail
    - ☐ les fichiers du disque
    - ☐ des processus
16. Un fichier source est :
    - ☐ un fichier texte qui sera traduit en instructions processeur
    - ☐ un document illisible pour les humains
    - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
    - ☐ un document de référence du système
    - ☐ un document qui doit être protégé
17. 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 sémantique
    - ☐ l'analyse des entrées clavier
    - ☐ l'analyse harmonique
18. L'ordonnancement par tourniquet permet :
    - ☐ de doubler la mémoire disponible
    - ☐ d'afficher des ronds colorés à l'écran
    - ☐ de ne pas perdre de temps avec la commutation de contexte
    - ☐ d'entretenir l'illusion que les processus tournent en parallèle
19. 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=1;i<5;i=i+1)`
    - ☐ `for(i=0;i<5;i=i+1)`
    - ☐ `for(i=0;i<=5;i=i+1)`
20. 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
- ☐ 6
- ☐ 3
- ☐ 20

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Le code suivant :

```
int i;
for (i = 4; i >= 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

☐ 4 3 2 1

☐ 1 2 3 4

☐ 0 1 2 3 4

☐ 4 3 2 1 0

2. 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  }
```

☐ la variable y vaut 5

☐ la variable x vaut 0

☐ la variable x vaut 5 et la variable y vaut 0

☐ le programme affiche "Faux"

3. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction

☐ `int tab[] = 5;`

☐ `char tableau[5];`

☐ `int[] new tableau(5);`

☐ `int toto[taille=5];`

☐ `int toto[5];`

4. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

☐ rien

☐ `Mineur`
`Majeur`

☐ `Majeur`

☐ `Mineur`

5. Sous unix (ou linux), la commande `cd` permet de :

☐ ouvir un bureau partagé (common desktop)

☐ récupérer un programme arrêté avec la commande `ab`

☐ jouer de la musique

☐ détruire un fichier

☐ changer de répertoire courant

6. Un programme en langage C doit comporter une et une seule définition de la fonction :

☐ include

☐ main

☐ begin

☐ init

7. Le code suivant :

```
int somme = 0;
int i;
for (i = 1; i < 4; i = i + 1)
{
  somme = somme + i;
}
printf("%d", somme);
```

affichera :

☐ 42

☐ 1

☐ 6

☐ 0

8. 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

☐ 20

☐ 6

9. 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 `x = 5`

☐ le terminal affiche "Faux"

☐ le terminal affiche `x = 2`

☐ le terminal affiche `5`

10. 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  }
```

- ☐ la variable x vaut $-\frac{1}{2}$
- ☐ le programme affiche ****
- ☐ la variable x vaut 16
- ☐ le programme affiche x

11. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
    - ☐ `int %d;`
    - ☐ `int k;`
    - ☐ `loop i;`
    - ☐ `int loop n;`
12. Le code suivant :

```
int i;
for (i = 0; i < 7; i = i + 2)
{
    printf("%d ", i);
}
printf("\n");
```

    affichera :
    - ☐ 0 2 4 6
    - ☐ 0 2 4 6 8
    - ☐ 0 1 2 3 4 5 6
    - ☐ 0 1 2 3 4 5 6 7
13. 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=0;i<5;i=i+1)`
    - ☐ `for(i=1;i<5;i=i+1)`
    - ☐ `for(i=0;i<=5;i=i+1)`
    - ☐ `for(i=1;i<=5;i=i+1)`
14. Un registre du processeur est :
    - ☐ un composant qui contient la liste des fichiers du système
    - ☐ 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
15. Le bus système sert à :
    - ☐ transporter les processus du tourniquet au processeur
    - ☐ Arriver à l'heure en cours
    - ☐ Écrire des données sur le dique dur
    - ☐ Transférer des données et intructions entre processeur et mémoire
16. 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 bus explose
    - ☐ la case mémoire 8 contiendra 0
    - ☐ la case mémoire 8 contiendra 16
    - ☐ le terminal affiche 8
17. 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
    - ☐ 0 1 2 3 4
18. Dans la commande gcc, l'option `-Wall` signifie :
    - ☐ qu'il faut lancer un déboggueur
    - ☐ qu'on veut changer alétoirement de fond d'écran
    - ☐ qu'il faut indenter le fichier source
    - ☐ que l'on veut voir tous les avertissements
19. 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 tente d'afficher des caractères sur une ligne qui va au delà de la largeur de la fenêtre du terminal
    - ☐ 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
20. 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 = %d`
    - ☐ `j = 0`
    - ☐ `j = 4`
    - ☐ `j = 5`

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse lexicale
   - ☐ analyse sémantique
   - ☐ analyse harmonique
   - ☐ analyse syntaxique
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. 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'analyse sémantique
   - ☐ l'analyse des entrées clavier
   - ☐ l'édition de liens
   - ☐ l'analyse harmonique
4. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
   - ☐ `#define N =  3`
   - ☐ `#define N 3`
   - ☐ `#define taille = N`
   - ☐ `#define taille = 3`
5. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

   affichera :
   - ☐ `Majeur`
   - ☐ `Mineur`
   - ☐ `Mineur`
     `Majeur`
   - ☐ rien
6. Le code suivant :

```
int somme = 0;
int i;
for (i = 1; i < 4; i = i + 1)
{
  somme = somme + i;
}
printf("%d", somme);
```

   affichera :
   - ☐ 1
   - ☐ 0
   - ☐ 42
   - ☐ 6
7. Si cette erreur apparaît à la compilation :
   `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
   - ☐ une accolade en trop
   - ☐ un point-virgule en trop
   - ☐ un point-virgule manquant
   - ☐ une accolade manquante
8. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :
   - ☐ 8 6 4 2
   - ☐ 8 2
   - ☐ 0 2 4 6 8
   - ☐ 8 6 4 2 0
9. 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 :
   - ☐ 6
   - ☐ 16
   - ☐ 3
   - ☐ 20
10. Le code suivant :

```
int i;
for (i = 1; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

    affichera :
    - ☐ 1 2 3 4
    - ☐ 4 3 2 1
    - ☐ 0 1 2 3 4
    - ☐ 4 3 2 1 0
11. 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

12. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
    - ☐ `yppasswd`
    - ☐ `mkdir TP4`
    - ☐ `kwrite TP4`
    - ☐ `new TP4`
13. Dans la commande gcc, l'option `-Wall` signifie :
    - ☐ que l'on veut voir tous les avertissements
    - ☐ qu'on veut changer alétoirement de fond d'écran
    - ☐ qu'il faut lancer un déboggueur
    - ☐ qu'il faut indenter le fichier source
14. Sous unix (ou linux), la commande `cd` permet de :
    - ☐ ouvir un bureau partagé (common desktop)
    - ☐ détruire un fichier
    - ☐ récupérer un programme arrêté avec la commande `ab`
    - ☐ changer de répertoire courant
    - ☐ jouer de la musique
15. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
    - ☐ `gcc -Wall prog.exe -o prog.c`
    - ☐ `gcc prog.c -o -Wall prog.exe`
    - ☐ `gcc prog.exe -Wall -o prog.c`
    - ☐ `gcc -Wall prog.c -o prog.exe`
16. Un registre du processeur est :
    - ☐ un composant qui contient la liste des fichiers du système
    - ☐ une gamme de fréquence de fonctionnement du processeur
    - ☐ une unité de calcul spécialisée de l'ordinateur
    - ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs
17. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
    - ☐ tous ensemble
    - ☐ en parallèle, chacun dans un registre
    - ☐ tour à tour, un petit peu à chaque fois
    - ☐ chacun son tour, après que le processus précédent a terminé
18. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ créer un fichier texte
    - ☐ changer de répertoire courant
    - ☐ ouvrir un fichier texte
    - ☐ créer un répertoire
19. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

    affichera :
    - ☐ `Mineur`
    - ☐ rien
    - ☐ `Majeur`
    - ☐ `Mineur`
      `Majeur`
20. 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 :
    - ☐ 4
    - ☐ 16
    - ☐ 0
    - ☐ 8

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. 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 `5`
   - ☐ le terminal affiche `x = 2`
   - ☐ le terminal affiche `x = 5`
   - ☐ le terminal affiche "Faux"

2. 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  }
```

   - ☐ la variable x vaut 5 et la variable y vaut 0
   - ☐ la variable x vaut 0
   - ☐ le programme affiche "Faux"
   - ☐ la variable y vaut 5

3. 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 ?
   - ☐ `1 2 3 1 2`
   - ☐ `0 1 0 1 0 1`
   - ☐ `0 1 2 0 1 2`
   - ☐ `0 0 1 1 2 2`

4. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

   affichera :
   - ☐ `Majeur`
   - ☐ rien
   - ☐ `Mineur`
   - ☐ `Mineur`
     `Majeur`

5. 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 = 0`
   - ☐ `j = 4`
   - ☐ `j = 5`
   - ☐ `j = %d`

6. Sous unix (ou linux), la commande `cd` permet de :
   - ☐ détruire un fichier
   - ☐ ouvir un bureau partagé (common desktop)
   - ☐ changer de répertoire courant
   - ☐ jouer de la musique
   - ☐ récupérer un programme arrêté avec la commande `ab`

7. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
   - ☐ `#define N 3`
   - ☐ `#define taille = 3`
   - ☐ `#define N =  3`
   - ☐ `#define taille = N`

8. Pour compiler un programme `prog.c`, on utilise la ligne de commande :
   - ☐ `gcc -Wall prog.exe -o prog.c`
   - ☐ `gcc prog.exe -Wall -o prog.c`
   - ☐ `gcc -Wall prog.c -o prog.exe`
   - ☐ `gcc prog.c -o -Wall prog.exe`

9. 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  }
```

   - ☐ la variable x vaut 16
   - ☐ le programme affiche `****`
   - ☐ la variable x vaut $-\frac{1}{2}$
   - ☐ le programme affiche x

10. 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=1;i<5;i=i+1)`
    - ☐ `for(i=0;i<5;i=i+1)`

11. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Mineur`
  `Majeur`
- ☐ rien
- ☐ `Mineur`
- ☐ `Majeur`

12. 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 :

- ☐ 8
- ☐ 0
- ☐ 4
- ☐ 16

13. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction

- ☐ `int loop n;`
- ☐ `int %d;`
- ☐ `loop i;`
- ☐ `int k;`

14. Sur unix (ou linux), la commande `mkdir` permet de :

- ☐ changer de répertoire courant
- ☐ créer un répertoire
- ☐ créer un fichier texte
- ☐ ouvrir un fichier texte

15. Le code suivant :

```
int i;
for (i = 0; i < 7; i = i + 2)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 0 2 4 6 8
- ☐ 0 2 4 6
- ☐ 0 1 2 3 4 5 6 7
- ☐ 0 1 2 3 4 5 6

16. Le code suivant :

```
int i;
for (i = 0; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 4 3 2 1 0
- ☐ 4 3 2 1
- ☐ 0 1 2 3
- ☐ 0 1 2 3 4

17. Quels calculs peut-on programmer en programmation structurée ?

- ☐ certains programmes sont de vrais plats de spaghetti
- ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine
- ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée
- ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine

18. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Mineur`
  `Majeur`
- ☐ `Mineur`
- ☐ rien
- ☐ `Majeur`

19. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction

- ☐ `int tab[] = 5;`
- ☐ `int[] new tableau(5);`
- ☐ `int toto[5];`
- ☐ `char tableau[5];`
- ☐ `int toto[taille=5];`

20. Le code suivant :

```
int somme = 0;
int i;
for (i = 1; i < 4; i = i + 1)
{
  somme = somme + i;
}
printf("%d", somme);
```

affichera :

- ☐ 1
- ☐ 42
- ☐ 6
- ☐ 0

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Le code suivant :

```
int i;
for (i = 1; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 4 3 2 1
- ☐ 4 3 2 1 0
- ☐ 1 2 3 4
- ☐ 0 1 2 3 4

2. 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

3. 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 x
- ☐ la variable x vaut 16
- ☐ le programme affiche `****`
- ☐ la variable x vaut $-\frac{1}{2}$

4. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ `Majeur`
- ☐ `Mineur`
- ☐ `Mineur`
  `Majeur`
- ☐ rien

5. 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 ?

- ☐ `** ** ** ** ** **`
- ☐ `**** **** **** ****`
- ☐ `** *** **** *****`
- ☐ `***** **** *** ***`

6. 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
- ☐ le bus explose
- ☐ la case mémoire 8 contiendra 0

7. 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 `x = 5`
- ☐ le terminal affiche `5`
- ☐ le terminal affiche "Faux"
- ☐ le terminal affiche `x = 2`

8. Dans la commande gcc, l'option `-Wall` signifie :

- ☐ qu'on veut changer alétoirement de fond d'écran
- ☐ qu'il faut lancer un déboggueur
- ☐ qu'il faut indenter le fichier source
- ☐ que l'on veut voir tous les avertissements

9. 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 ?

☐ 1 2 3 1 2

☐ 0 1 0 1 0 1

☐ 0 1 2 0 1 2

☐ 0 0 1 1 2 2

10. 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

☐ 0 1 2 0 1 2

☐ 0 1 0 1 0 1 0 1

☐ 1 2 1 2 3

11. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

☐ `#include <studio.h>`

☐ `#appart <stdlib.h>`

☐ `#include <stdio.h>`

☐ `#include <studlib.h>`

12. Le code suivant :

```
int i;
for (i = 4; i > 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

☐ 4 3 2 1

☐ 0 1 2 3

☐ 0 1 2 3 4

☐ 4 3 2 1 0

13. 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  }
```

☐ le programme affiche "Faux"

☐ la variable x vaut 3

☐ la variable x vaut 5 et la variable y vaut 3

☐ la variable y vaut 5

14. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :

☐ `#include <studio.h>`

☐ `#include <studlib.h>`

☐ `#appart <stdlib.h>`

☐ `#include <stdio.h>`

15. 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 :

☐ 10

☐ 6

☐ 15

☐ 0

16. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.

☐ `#define taille = 3`

☐ `#define N 3`

☐ `#define N =  3`

☐ `#define taille = N`

17. Un fichier source est :

☐ un document illisible pour les humains

☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur

☐ un fichier texte qui sera traduit en instructions processeur

☐ un document de référence du système

☐ un document qui doit être protégé

18. Pour compiler un programme `prog.c`, on utilise la ligne de commande :

☐ `gcc prog.exe -Wall -o prog.c`

☐ `gcc -Wall prog.c -o prog.exe`

☐ `gcc -Wall prog.exe -o prog.c`

☐ `gcc prog.c -o -Wall prog.exe`

19. 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=1;i<5;i=i+1)`

☐ `for(i=0;i<5;i=i+1)`

☐ `for(i=0;i<=5;i=i+1)`

20. Quels calculs peut-on programmer en programmation structurée ?

☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine

☐ certains programmes sont de vrais plats de spaghetti

☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée

☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `int tab[] = 5;`
   - ☐ `int toto[5];`
   - ☐ `int toto[taille=5];`
   - ☐ `char tableau[5];`
   - ☐ `int[] new tableau(5);`
2. 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 bus explose
   - ☐ la case mémoire 8 contiendra 0
   - ☐ le terminal affiche 8
   - ☐ la case mémoire 8 contiendra 16
3. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
   - ☐ `loop i;`
   - ☐ `int loop n;`
   - ☐ `int k;`
   - ☐ `int %d;`
4. 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 :
   - ☐ 3
   - ☐ 20
   - ☐ 6
   - ☐ 16
5. Quel est l'opérateur de différence en C :
   - ☐ `!`
   - ☐ `<>`
   - ☐ `!=`
   - ☐ $\neq$
6. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

   affichera :
   - ☐ rien
   - ☐ `Majeur`
   - ☐ `Mineur`
`Majeur`
   - ☐ `Mineur`
7. Le bus système sert à :
   - ☐ transporter les processus du tourniquet au processeur
   - ☐ Transférer des données et intructions entre processeur et mémoire
   - ☐ Arriver à l'heure en cours
   - ☐ Écrire des données sur le dique dur
8. 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 1 0 1 0 1 0 1`
   - ☐ `0 1 2 0 1 2`
   - ☐ `0 0 0 1 1 1`
   - ☐ `1 2 1 2 3`
9. Dans la commande gcc, l'option `-Wall` signifie :
   - ☐ qu'il faut lancer un déboggueur
   - ☐ qu'on veut changer alétoirement de fond d'écran
   - ☐ qu'il faut indenter le fichier source
   - ☐ que l'on veut voir tous les avertissements
10. Le code suivant :

```
int i;
for (i = 1; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

    affichera :
    - ☐ 0 1 2 3 4
    - ☐ 4 3 2 1 0
    - ☐ 4 3 2 1
    - ☐ 1 2 3 4
11. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 8 6 4 2
- ☐ 8 6 4 2 0
- ☐ 8 2
- ☐ 0 2 4 6 8

12. 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 7
- ☐ 0 1 2 3 4 5 6
- ☐ 0 2 4 6
- ☐ 0 2 4 6 8

13. 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
- ☐ 6
- ☐ 15

14. Le code suivant :

```
int i;
for (i = 4; i > 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 4 3 2 1 0
- ☐ 0 1 2 3 4
- ☐ 4 3 2 1
- ☐ 0 1 2 3

15. Le code suivant :

```
int somme = 0;
int i;
for (i = 1; i < 4; i = i + 1)
{
  somme = somme + i;
}
printf("%d", somme);
```

affichera :

- ☐ 42
- ☐ 1
- ☐ 0
- ☐ 6

16. Sous unix (ou linux), la commande `cd` permet de :

- ☐ ouvir un bureau partagé (common desktop)
- ☐ récupérer un programme arrêté avec la commande `ab`
- ☐ jouer de la musique
- ☐ changer de répertoire courant
- ☐ détruire un fichier

17. Sur unix (ou linux), la commande `mkdir` permet de :

- ☐ ouvrir un fichier texte
- ☐ créer un fichier texte
- ☐ créer un répertoire
- ☐ changer de répertoire courant

18. 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=0;i<5;i=i+1)`
- ☐ `for(i=1;i<5;i=i+1)`
- ☐ `for(i=0;i<=5;i=i+1)`
- ☐ `for(i=1;i<=5;i=i+1)`

19. Le langage C est un langage

- ☐ interprété
- ☐ compilé
- ☐ composé
- ☐ lu, écrit, parlé

20. L'ordonnancement par tourniquet permet :

- ☐ de doubler la mémoire disponible
- ☐ d'entretenir l'illusion que les processus tournent en parallèle
- ☐ de ne pas perdre de temps avec la commutation de contexte
- ☐ d'afficher des ronds colorés à l'écran

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. 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  }
```

   - ☐ la variable x vaut $-\frac{1}{2}$
   - ☐ le programme affiche `****`
   - ☐ la variable x vaut 16
   - ☐ le programme affiche x

2. 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=%d et y=%d\n",x,y);`
   - ☐ `printf("x=%d et y=%d\n,x,y");`
   - ☐ `printf("x=%x et y=%y\n");`

3. 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 x vaut 3
   - ☐ le programme affiche "Faux"
   - ☐ la variable y vaut 5
   - ☐ la variable x vaut 5 et la variable y vaut 3

4. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

   affichera :
   - ☐ `Majeur`
   - ☐ rien
   - ☐ `Mineur`
   - ☐ `Mineur`
`Majeur`

5. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
printf("Majeur\n");
```

   affichera :
   - ☐ `Mineur`
`Majeur`
   - ☐ `Mineur`
   - ☐ rien
   - ☐ `Majeur`

6. 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;`

7. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

   affichera :
   - ☐ `Mineur`
`Majeur`
   - ☐ `Mineur`
   - ☐ `Majeur`
   - ☐ rien

8. 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 1 2 0 1 2`
   - ☐ `0 1 0 1 0 1 0 1`
   - ☐ `0 0 0 1 1 1`
   - ☐ `1 2 1 2 3`

9. 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
- ☐ ne comportent aucune erreur
- ☐ 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

10. Le code suivant :

```
int age = 18;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

- ☐ rien
- ☐ `Mineur`
- ☐ `Majeur`
- ☐ `Mineur`
  `Majeur`

11. Si cette erreur apparaît à la compilation :
`error: expected ';' before '}' token` que doit-on chercher dans le programme ?
    - ☐ une accolade manquante
    - ☐ un point-virgule manquant
    - ☐ une accolade en trop
    - ☐ un point-virgule en trop

12. 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
- ☐ 6
- ☐ 15

13. La virtualisation de la mémoire permet notamment de stocker des portions inactives de la mémoire de travail sur le disque dur. Mais on perd :
    - ☐ des processus
    - ☐ certaines données de la mémoire de travail
    - ☐ les fichiers du disque
    - ☐ en temps d'accès

14. Le code suivant :

```
int somme = 0;
int i;
for (i = 1; i < 4; i = i + 1)
{
  somme = somme + i;
}
printf("%d", somme);
```

affichera :

- ☐ 0
- ☐ 1
- ☐ 42
- ☐ 6

15. Quel est l'opérateur de différence en C :
    - ☐ $\neq$
    - ☐ `!=`
    - ☐ `!`
    - ☐ `<>`

16. 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=0;i<=5;i=i+1)`
    - ☐ `for(i=1;i<=5;i=i+1)`
    - ☐ `for(i=0;i<5;i=i+1)`
    - ☐ `for(i=1;i<5;i=i+1)`

17. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
    - ☐ `yppasswd`
    - ☐ `new TP4`
    - ☐ `mkdir TP4`
    - ☐ `kwrite TP4`

18. Le code suivant :

```
int i;
for (i = 1; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 0 1 2 3 4
- ☐ 1 2 3 4
- ☐ 4 3 2 1
- ☐ 4 3 2 1 0

19. 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`

20. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ ouvrir un fichier texte
    - ☐ créer un répertoire
    - ☐ créer un fichier texte
    - ☐ changer de répertoire courant

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. 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 x vaut 5 et la variable y vaut 3
   ☐ le programme affiche "Faux"
   ☐ la variable x vaut 3
   ☐ la variable y vaut 5

2. 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 0 1 1 2 2`
   ☐ `1 2 3 1 2`
   ☐ `0 1 2 0 1 2`
   ☐ `0 1 0 1 0 1`

3. Quel est l'opérateur de différence en C :
   ☐ `!=`
   ☐ `!`
   ☐ `<>`
   ☐ $\neq$

4. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :
   ☐ 8 6 4 2 0
   ☐ 8 6 4 2
   ☐ 8 2
   ☐ 0 2 4 6 8

5. Si cette erreur apparaît à la compilation : `Undefined symbols :"_prinft"` ou `référence indéfinie vers « prinft »` que doit-on chercher dans le programme ?
   ☐ une faute de frappe dans un appel de fonction
   ☐ une directive préprocesseur `#include` manquante
   ☐ un caractère interdit en C
   ☐ une variable non déclarée

6. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
   ☐ tour à tour, un petit peu à chaque fois
   ☐ chacun son tour, après que le processus précédent a terminé
   ☐ tous ensemble
   ☐ en parallèle, chacun dans un registre

7. Le code suivant :

```
int i;
for (i = 4; i > 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :
   ☐ 4 3 2 1
   ☐ 0 1 2 3 4
   ☐ 4 3 2 1 0
   ☐ 0 1 2 3

8. 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`

9. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   ☐ `#appart <stdlib.h>`
   ☐ `#include <stdio.h>`
   ☐ `#include <studio.h>`
   ☐ `#include <studlib.h>`

10. 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 = 0`
    ☐ `j = %d`
    ☐ `j = 5`

11. Les lignes

```
int i;
int x=0;
for(i=0,i<5,i=i+1)
{
  x=x+1;
}
```

    ☐ comportent une erreur qui sera détectée au cours de l'analyse syntaxique

- ☐ ne comportent aucune erreur
- ☐ comportent une erreur qui ne sera pas détectée
- ☐ comportent une erreur qui sera détectée au cours de l'édition de lien

12. L'ordonnancement par tourniquet permet :
    - ☐ d'entretenir l'illusion que les processus tournent en parallèle
    - ☐ de ne pas perdre de temps avec la commutation de contexte
    - ☐ d'afficher des ronds colorés à l'écran
    - ☐ de doubler la mémoire disponible
13. Quels calculs peut-on programmer en programmation structurée ?
    - ☐ certains programmes sont de vrais plats de spaghetti
    - ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine
    - ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine
    - ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée
14. Le code suivant :

```
int i;
for (i = 4; i >= 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

- ☐ 1 2 3 4
- ☐ 4 3 2 1 0
- ☐ 4 3 2 1
- ☐ 0 1 2 3 4

15. Sous unix (ou linux), la commande `cd` permet de :
    - ☐ détruire un fichier
    - ☐ jouer de la musique
    - ☐ ouvir un bureau partagé (common desktop)
    - ☐ récupérer un programme arrêté avec la commande `ab`
    - ☐ changer de répertoire courant
16. 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=1;i<5;i=i+1)`
    - ☐ `for(i=0;i<5;i=i+1)`
17. 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=%d et y=%d\n",x,y);`
    - ☐ `printf("x=%d et y=%d\n,x,y");`
    - ☐ `printf("x=%x et y=%y\n");`
18. 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  }
```

- ☐ la variable x vaut $-\frac{1}{2}$
- ☐ le programme affiche x
- ☐ le programme affiche `****`
- ☐ la variable x vaut 16

19. Dans la commande gcc, l'option `-Wall` signifie :
    - ☐ qu'il faut indenter le fichier source
    - ☐ que l'on veut voir tous les avertissements
    - ☐ qu'on veut changer alétoirement de fond d'écran
    - ☐ qu'il faut lancer un déboggueur
20. 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 :

- ☐ 4
- ☐ 8
- ☐ 0
- ☐ 16

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
   - ☐ chacun son tour, après que le processus précédent a terminé
   - ☐ tous ensemble
   - ☐ en parallèle, chacun dans un registre
   - ☐ tour à tour, un petit peu à chaque fois

2. 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=1;i<5;i=i+1)`
   - ☐ `for(i=0;i<5;i=i+1)`
   - ☐ `for(i=0;i<=5;i=i+1)`

3. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#include <studio.h>`
   - ☐ `#appart <stdlib.h>`
   - ☐ `#include <stdio.h>`
   - ☐ `#include <studlib.h>`

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 = %d`
   - ☐ `j = 4`
   - ☐ `j = 0`

5. 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=%d et y=%d\n,x,y");`
   - ☐ `printf("x=%x et y=%y\n");`
   - ☐ `printf("x=%d et y=%d\n",x,y);`

6. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

   affichera :
   - ☐ `Mineur`
`Majeur`
   - ☐ rien
   - ☐ `Mineur`
   - ☐ `Majeur`

7. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ begin
   - ☐ init
   - ☐ main
   - ☐ include

8. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
   - ☐ `loop i;`
   - ☐ `int %d;`
   - ☐ `int loop n;`
   - ☐ `int k;`

9. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
printf("Majeur\n");
```

   affichera :
   - ☐ `Mineur`
`Majeur`
   - ☐ `Mineur`
   - ☐ rien
   - ☐ `Majeur`

10. 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 = %d`
    - ☐ `j = 0`
    - ☐ `j = 4`
    - ☐ `j = 5`

11. Une *segmentation fault* est une erreur qui survient lorsque :
    - ☐ la division du programme en zones homogènes échoue
    - ☐ 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

☐ le programme tente d'accèder à une partie de la mémoire qui ne lui est pas réservée

12. 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
☐ des processus
☐ les fichiers du disque
☐ en temps d'accès

13. 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 ?

☐ `**** **** **** ****`
☐ `** ** ** ** ** **`
☐ `** *** **** *****`
☐ `***** **** *** ***`

14. 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

15. Sur unix (ou linux), la commande `mkdir` permet de :

☐ ouvrir un fichier texte
☐ changer de répertoire courant
☐ créer un répertoire
☐ créer un fichier texte

16. 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  }
```

☐ la variable x vaut 5 et la variable y vaut 0
☐ le programme affiche "Faux"
☐ la variable y vaut 5
☐ la variable x vaut 0

17. Le code suivant :

```
int i;
for (i = 0; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

affichera :

☐ 4 3 2 1 0
☐ 0 1 2 3
☐ 0 1 2 3 4
☐ 4 3 2 1

18. Un fichier source est :

☐ un document qui doit être protégé
☐ un document illisible pour les humains
☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
☐ un fichier texte qui sera traduit en instructions processeur
☐ un document de référence du système

19. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction

☐ `int toto[taille=5];`
☐ `int tab[] = 5;`
☐ `int[] new tableau(5);`
☐ `char tableau[5];`
☐ `int toto[5];`

20. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :

☐ sélectionner entre deux blocs à l'aide d'une condition
☐ mettre les blocs en séquence les uns à la suite des autres
☐ répéter un bloc tant qu'une condition est vérifée
☐ retourner un bloc

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** $-0,5$ **points par réponse fausse. Durée :** 20 **minutes.**

1. Quel est l'opérateur de différence en C :
   - ☐ `!`
   - ☐ $\neq$
   - ☐ `<>`
   - ☐ `!=`

2. 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 `5`
   - ☐ le terminal affiche `x = 5`
   - ☐ le terminal affiche "Faux"
   - ☐ le terminal affiche `x = 2`

3. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
   - ☐ `#include <stdio.h>`
   - ☐ `#appart <stdlib.h>`
   - ☐ `#include <studio.h>`
   - ☐ `#include <studlib.h>`

4. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `char tableau[5];`
   - ☐ `int[] new tableau(5);`
   - ☐ `int toto[5];`
   - ☐ `int tab[] = 5;`
   - ☐ `int toto[taille=5];`

5. Si cette erreur apparaît à la compilation : `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
   - ☐ un point-virgule manquant
   - ☐ un point-virgule en trop
   - ☐ une accolade en trop
   - ☐ une accolade manquante

6. 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 ?
   - ☐ `1 2 1 2 3`
   - ☐ `0 0 0 1 1 1`
   - ☐ `0 1 0 1 0 1 0 1`
   - ☐ `0 1 2 0 1 2`

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
   - ☐ certains programmes sont de vrais plats de spaghetti
   - ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine
   - ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine

8. 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 :
   - ☐ 10
   - ☐ 0
   - ☐ 15
   - ☐ 6

9. Le code suivant :

```
int i;
for (i = 4; i > 0; i = i - 1)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :
   - ☐ `0 1 2 3 4`
   - ☐ `4 3 2 1 0`
   - ☐ `0 1 2 3`
   - ☐ `4 3 2 1`

10. Le langage C est un langage
    - ☐ composé
    - ☐ lu, écrit, parlé
    - ☐ interprété
    - ☐ compilé

11. Le code suivant :

```
int age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

    affichera :
    - ☐ `Majeur`
    - ☐ `Mineur`
    - ☐ rien
    - ☐ `Mineur`
      `Majeur`

12. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
    - ☐ analyse syntaxique
    - ☐ analyse harmonique
    - ☐ analyse sémantique
    - ☐ analyse lexicale

13. Si cette erreur apparaît à la compilation : `Undefined symbols :"_prinft"` ou `référence indéfinie vers` « `prinft` » que doit-on chercher dans le programme ?
    - ☐ une variable non déclarée
    - ☐ une faute de frappe dans un appel de fonction
    - ☐ un caractère interdit en C
    - ☐ une directive préprocesseur `#include` manquante

14. 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 y vaut 5
    - ☐ la variable x vaut 0
    - ☐ la variable x vaut 5 et la variable y vaut 0

15. L'ordonnancement par tourniquet permet :
    - ☐ d'entretenir l'illusion que les processus tournent en parallèle
    - ☐ de doubler la mémoire disponible
    - ☐ d'afficher des ronds colorés à l'écran
    - ☐ de ne pas perdre de temps avec la commutation de contexte

16. Le code suivant :

```
int i;
for (i = 8; i > 0; i = i - 2)
{
    printf("%d ", i);
}
printf("\n");
```

    affichera :
    - ☐ 8 6 4 2
    - ☐ 8 2
    - ☐ 0 2 4 6 8
    - ☐ 8 6 4 2 0

17. 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=%x et y=%y\n");`
    - ☐ `printf("x=%d et y=%d\n",x y);`
    - ☐ `printf("x=%d et y=%d\n",x,y);`
    - ☐ `printf("x=%d et y=%d\n,x,y");`

18. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
    - ☐ `#define N =  3`
    - ☐ `#define N 3`
    - ☐ `#define taille = N`
    - ☐ `#define taille = 3`

19. Une *segmentation fault* est une erreur qui survient lorsque :
    - ☐ 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
    - ☐ le programme tente d'accèder à une partie de la mémoire qui ne lui est pas réservée
    - ☐ la division du programme en zones homogènes échoue

20. Un registre du processeur est :
    - ☐ une unité de calcul spécialisée de l'ordinateur
    - ☐ un composant qui contient la liste des fichiers du système
    - ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs
    - ☐ une gamme de fréquence de fonctionnement du processeur

Licence 1 **Éléments d'informatique – contrôle continue** Prénom : Nom :
N° INE : .

Barème : 1 **points par réponse juste (unique) ;** −0,5 **points par réponse fausse. Durée :** 20 **minutes.**

1. 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  }
```

   ☐ le programme affiche "Faux"
   ☐ la variable x vaut 5 et la variable y vaut 3
   ☐ la variable y vaut 5
   ☐ la variable x vaut 3

2. Un programme en langage C doit comporter une et une seule définition de la fonction :
   ☐ begin
   ☐ init
   ☐ include
   ☐ main

3. 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 :
   ☐ 0
   ☐ 16
   ☐ 4
   ☐ 8

4. Le code suivant :

```
int i;
for (i = 0; i < 5; i = i + 1)
{
    printf("%d ", i);
}
printf("\n");
```

   affichera :
   ☐ 4 3 2 1
   ☐ 4 3 2 1 0
   ☐ 0 1 2 3 4
   ☐ 0 1 2 3

5. 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>`

6. 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 :
   ☐ 15
   ☐ 6
   ☐ 0
   ☐ 10

7. Afin de représenter la taille d'un tableau, définir une constante symbolique N valant 3.
   ☐ `#define taille = 3`
   ☐ `#define N =  3`
   ☐ `#define N 3`
   ☐ `#define taille = N`

8. Un fichier source est :
   ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
   ☐ un document illisible pour les humains
   ☐ un fichier texte qui sera traduit en instructions processeur
   ☐ un document qui doit être protégé
   ☐ un document de référence du système

9. Si cette erreur apparaît à la compilation : `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
   ☐ une accolade manquante
   ☐ un point-virgule en trop
   ☐ un point-virgule manquant
   ☐ une accolade en trop

10. 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 7
   ☐ 0 2 4 6 8
   ☐ 0 2 4 6
   ☐ 0 1 2 3 4 5 6

11. Quels calculs peut-on programmer en programmation structurée ?
   ☐ il y a des calculs programmables en programmation structurée qui ne sont pas programmables en langage machine
   ☐ il y a des calculs programmables en langage machine et qui ne sont pas programmables en programmation structurée
   ☐ certains programmes sont de vrais plats de spaghetti
   ☐ en programmation structurée on peut programmer tous les calculs programmables en langage machine

12. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :

- ☐ analyse syntaxique
- ☐ analyse harmonique
- ☐ analyse sémantique
- ☐ analyse lexicale

13. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :

- ☐ `new TP4`
- ☐ `kwrite TP4`
- ☐ `mkdir TP4`
- ☐ `yppasswd`

14. 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

15. 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 3 0 1 2`
- ☐ `0 1 2 0 1 2 3`
- ☐ `0 1 2 0 1 2`

16. 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  }
```

- ☐ la variable x vaut $-\frac{1}{2}$
- ☐ la variable x vaut 16
- ☐ le programme affiche `****`
- ☐ le programme affiche x

17. 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
```

- ☐ la case mémoire 8 contiendra 16
- ☐ la case mémoire 8 contiendra 0
- ☐ le bus explose
- ☐ le terminal affiche 8

18. 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 = 0`
- ☐ `j = %d`
- ☐ `j = 4`
- ☐ `j = 5`

19. 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 = 5`
- ☐ `j = 4`
- ☐ `j = %d`
- ☐ `j = 0`

20. Une de ces manière de composer les blocs de programmes ne fait pas partie des opérations de la programmation structurée :

- ☐ mettre les blocs en séquence les uns à la suite des autres
- ☐ répéter un bloc tant qu'une condition est vérifée
- ☐ retourner un bloc
- ☐ sélectionner entre deux blocs à l'aide d'une condition