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. Vous utilisez une boucle `while` quand :
   - ☐ vous avez déjà fait un `for` dans le même programme principal
   - ☐ vous n'avez pas déclaré de fonction
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
2. Si cette erreur apparaît à la compilation : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une fonction déclarée mais non définie
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
3. Le code suivant :

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

   affichera :
   - ☐ `Mineur`
   - ☐ `Mineur`
     `Majeur`
   - ☐ `Majeur`
   - ☐ rien
4. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
   - ☐ `void saisie_utilisateur(char c);`
   - ☐ `saisie_utilisateur(scanf(%d));`
   - ☐ `void saisie_utilisateur(int n);`
   - ☐ `int saisie_utilisateur();`
5. 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>`
6. 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 document de référence du système
   - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
7. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   - ☐ 0
   - ☐ 7
   - ☐ 5
8. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ dans lequel vous avez déclaré ces fonction
   - ☐ dans lequel ces fonctions sont appelées dans le main
   - ☐ un ordre quelconque
   - ☐ alphabétique
9. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ begin
   - ☐ include
   - ☐ main
   - ☐ init
10. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que `x` est une variable réelle définie et initialisée, il est incorrect d'écrire :
    - ☐ `x = racine(2/3);`
    - ☐ `x = racine(x * x) - racine(x);`
    - ☐ `x = racine(racine(x)*racine(x));`
    - ☐ `x - 1 = racine(x);`
11. Sous unix (ou linux), la commande `cd` permet de :
    - ☐ ouvir un bureau partagé (common desktop)
    - ☐ changer de répertoire courant
    - ☐ jouer de la musique
    - ☐ détruire un fichier
    - ☐ récupérer un programme arrêté avec la commande `ab`
12. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
    - ☐ `void exposant(double x^n);`
    - ☐ `exposant(double x, int n, int r);`
    - ☐ `int exposant(double n, int x);`
    - ☐ `double exposant(double x, int n);`
13. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
    - ☐ un débogueur
    - ☐ `scanf("%d", &n);`
    - ☐ `printf("Valeur de n ? %g\n", n);`
    - ☐ `printf("Valeur de n ? %d\n", n);`

14. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une fonction déclarée mais non définie
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
15. 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
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
    - ☐ répéter un bloc tant qu'une condition est vérifée
    - ☐ mettre les blocs en séquence les uns à la suite des autres
    - ☐ retourner un bloc
17. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses chants
    - ☐ ses blocs
    - ☐ ses champs
    - ☐ ses cases
18. Avant de faire appel à une fonction il est nécessaire de :
    - ☐ l'avoir déclarée
    - ☐ l'avoir déclarée et définie
    - ☐ avoir défini une constante symbolique de la taille de cette fonction
    - ☐ l'avoir définie
19. 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 ?

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

20. Une variable booléenne est un variable :
    - ☐ à laquelle une valeur vient d'être affectée
    - ☐ réelle positive
    - ☐ qui est vraie ou fausse
    - ☐ NaN (not a number, qui n'est pas un nombre)
    - ☐ jamais nulle

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 < 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`

2. 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

3. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

- ☐ `ccccccc`
- ☐ `A`
- ☐ `ABCDEF`
- ☐ `i`

4. 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

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

6. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il ne compile pas
- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il n'affiche rien
- ☐ il comporte une boucle infinie

7. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :

- ☐ `int factorielle(double n);`
- ☐ `struct int factorielle(int n);`
- ☐ `int factorielle(int x);`
- ☐ `int factorielle();`

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

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

9. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 0
- ☐ 7
- ☐ 5

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

11. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
    - ☐ `void afficher_menu();`
    - ☐ `int afficher_menu(int char);`
    - ☐ `int afficher_menu();`
    - ☐ `double afficher_menu();`
    - ☐ `char afficher_menu(printf("menu"));`
12. 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>`
13. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
    - ☐ `n = factorielle();`
    - ☐ `int factorielle(int 2);`
    - ☐ `n = factorielle(p, q);`
    - ☐ `printf("%d", factorielle(n));`
14. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

    Alors l'expression `f(0)` prendra la valeur :
    - ☐ 3
    - ☐ 4
    - ☐ 0
15. L'écriture 111 en binaire correspond au nombre naturel :
    - ☐ 7
    - ☐ 111
    - ☐ 8
    - ☐ 3
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 :
    - ☐ 10
    - ☐ 6
    - ☐ 0
    - ☐ 15
17. Le type des réels en C est :
    - ☐ `double`
    - ☐ `int`
    - ☐ `char`
    - ☐ `real`
18. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses champs
    - ☐ ses chants
    - ☐ ses cases
    - ☐ ses blocs
19. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
    - ☐ `int char;`
    - ☐ `char 'c';`
    - ☐ `char c;`
    - ☐ `char "c";`
20. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

    appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

    L'affichage dans le main est le suivant :
    - ☐ `f(a,b)=8, a=3, b=5`
    - ☐ `f(a,b)=13, a=8, b=5`
    - ☐ `f(3,5)=8, a=3, b=5`
    - ☐ `f(a,b)=8, a=8, b=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. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `a{n, x} == b{n, x}`
- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a == b`
- ☐ `a = b`

2. Si cette erreur apparaît à la compilation : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
   - ☐ une fonction déclarée mais non définie
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une fonction appelée avant sa déclaration

3. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il risque d'afficher bonjour à la place de coucou
   - ☐ il ne compile pas
   - ☐ il comporte une boucle infinie
   - ☐ il n'affiche rien

4. 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>`

5. Le code suivant :

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

affichera :

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

6. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
   - ☐ `int factorielle(double n);`
   - ☐ `int factorielle();`
   - ☐ `struct int factorielle(int n);`
   - ☐ `int factorielle(int x);`

7. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
   - ☐ `void afficher_menu();`
   - ☐ `int afficher_menu();`
   - ☐ `double afficher_menu();`
   - ☐ `char afficher_menu(printf("menu"));`
   - ☐ `int afficher_menu(int char);`

8. 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

9. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

   - ☐ `f(a,b)=13, a=8, b=5`
   - ☐ `f(a,b)=8, a=3, b=5`
   - ☐ `f(a,b)=8, a=8, b=5`
   - ☐ `f(3,5)=8, a=3, b=5`

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

11. Vous utilisez une boucle `while` quand :
   - ☐ vous n'avez pas déclaré de fonction
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous avez déjà fait un `for` dans le même programme principal

12. 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
    - ☐ une accolade manquante
    - ☐ un point-virgule manquant
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
    - ☐ 4 3 2 1 0
    - ☐ 4 3 2 1
    - ☐ 1 2 3 4
14. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ créer un fichier texte
    - ☐ changer de répertoire courant
    - ☐ créer un répertoire
    - ☐ ouvrir un fichier texte
15. Avant de faire appel à une fonction il est nécessaire de :
    - ☐ avoir défini une constante symbolique de la taille de cette fonction
    - ☐ l'avoir déclarée
    - ☐ l'avoir déclarée et définie
    - ☐ l'avoir définie
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 = 0`
    - ☐ `j = 4`
    - ☐ `j = %d`
    - ☐ `j = 5`
17. 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`
18. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

    Alors l'expression `f(0)` prendra la valeur :
    - ☐ 0
    - ☐ 4
    - ☐ 3
19. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

    Alors l'expression `f(1)` prendra la valeur :
    - ☐ 1
    - ☐ 5
    - ☐ 4
    - ☐ 0
20. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
    - ☐ `int pgcd(int x, y);`
    - ☐ `int pgcd(int y, int x);`
    - ☐ `void pgcd(int x, int y);`
    - ☐ `int pgcd(int x, int x);`

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 le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :

- ☐ `int toto.n = 3;`
- ☐ `toto_s struct z = {3, 0.5};`
- ☐ `struct toto_s toto;`
- ☐ `toto_s n, x;`
- ☐ `int struct toto_s = {3, -1e10};`

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'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

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. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

Alors l'affichage sera :

- ☐ `C`
- ☐ `B`
- ☐ `A`
- ☐ `b`

5. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une fonction déclarée mais non définie
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une fonction appelée avant sa déclaration

6. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(3,5)=8, a=3, b=5`

7. Avant de faire appel à une fonction il est nécessaire de :
   - ☐ l'avoir déclarée et définie
   - ☐ l'avoir définie
   - ☐ l'avoir déclarée
   - ☐ avoir défini une constante symbolique de la taille de cette fonction

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. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
   - ☐ 3
   - ☐ 111
   - ☐ 7
   - ☐ 8

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

11. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :
    - ☐ `n = pgcd(n, 3);`
    - ☐ `n = pgcd(int p, int q);`
    - ☐ `int n = pgcd();`
    - ☐ `int pgcd(2);`
12. 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
    - ☐ retourner un bloc
    - ☐ mettre les blocs en séquence les uns à la suite des autres
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=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)`
14. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :
- ☐ 5
- ☐ 7
- ☐ 0

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

16. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
    - ☐ un débogueur
    - ☐ `printf("Valeur de n ? %g\n", n);`
    - ☐ `scanf("%d", &n);`
    - ☐ `printf("Valeur de n ? %d\n", n);`
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 = 2`
- ☐ le terminal affiche `x = 5`
- ☐ le terminal affiche "Faux"
- ☐ le terminal affiche `5`

18. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses chants
    - ☐ ses blocs
    - ☐ ses champs
    - ☐ ses cases
19. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
    - ☐ `char c;`
    - ☐ `char 'c';`
    - ☐ `char "c";`
    - ☐ `int char;`
20. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
    - ☐ `int afficher_menu(int char);`
    - ☐ `int afficher_menu();`
    - ☐ `char afficher_menu(printf("menu"));`
    - ☐ `double afficher_menu();`
    - ☐ `void afficher_menu();`

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 le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   ☐ il ne compile pas
   ☐ il comporte une boucle infinie
   ☐ il n'affiche rien
   ☐ il risque d'afficher bonjour à la place de coucou

2. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   ☐ un ordre quelconque
   ☐ dans lequel ces fonctions sont appelées dans le main
   ☐ alphabétique
   ☐ dans lequel vous avez déclaré ces fonction

3. 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)

4. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
   ☐ `x = racine(2/3);`
   ☐ `x = racine(x * x) - racine(x);`
   ☐ `x - 1 = racine(x);`
   ☐ `x = racine(racine(x)*racine(x));`

5. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   ☐ `f(a,b)=8, a=3, b=5`
   ☐ `f(3,5)=8, a=3, b=5`
   ☐ `f(a,b)=13, a=8, b=5`
   ☐ `f(a,b)=8, a=8, b=5`

6. L'écriture 101 en binaire correspond au nombre naturel :
   ☐ 4
   ☐ 5
   ☐ 101
   ☐ 3

7. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
   ☐ `n = carre(n);`
   ☐ `int carre(2);`
   ☐ `int n = carre();`
   ☐ `n = carre(int n);`

8. 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`

9. 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

10. 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

11. 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

12. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

    Alors l'affichage sera :
    ☐ `cccccc`
    ☐ `A`
    ☐ `ABCDEF`
    ☐ `i`

13. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
    - ☐ `struct date_s afficher_date(struct date_s d);`
    - ☐ `void afficher_date(date_s d);`
    - ☐ `void afficher_date(struct date_s d);`
    - ☐ `int afficher_date(date_s d);`
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`
    - ☐ `1 2 3 1 2`
    - ☐ `0 0 1 1 2 2`
    - ☐ `0 1 2 0 1 2`
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. Si cette erreur apparaît à la compilation : `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
    - ☐ un point-virgule manquant
    - ☐ une accolade en trop
    - ☐ un point-virgule en trop
    - ☐ une accolade manquante
17. Le code suivant :

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

    affichera :
    - ☐ `Majeur`
    - ☐ rien
    - ☐ `Mineur`
`Majeur`
    - ☐ `Mineur`
18. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

    Alors l'expression `g(0)` prendra la valeur :
    - ☐ 5
    - ☐ 7
    - ☐ 0
19. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
    - ☐ `void pgcd(int x, int y);`
    - ☐ `int pgcd(int y, int x);`
    - ☐ `int pgcd(int x, int x);`
    - ☐ `int pgcd(int x, y);`
20. Le code suivant :

```
int age = 20;
if (age < 18)
{
    printf("Mineur\n");
}
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. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
   - ☐ un débogueur
   - ☐ `printf("Valeur de n ? %d\n", n);`
   - ☐ `scanf("%d", &n);`
   - ☐ `printf("Valeur de n ? %g\n", n);`
2. 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>`
3. Si cette erreur apparaît à la compilation :
   `erreur: conflicting types for ‘max’` , que doit-on chercher dans le programme ?
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une fonction déclarée mais non définie
4. Sous unix (ou linux), la commande `cd` permet de :
   - ☐ détruire un fichier
   - ☐ récupérer un programme arrêté avec la commande `ab`
   - ☐ changer de répertoire courant
   - ☐ ouvir un bureau partagé (common desktop)
   - ☐ jouer de la musique
5. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
   - ☐ `int carre(2);`
   - ☐ `n = carre(n);`
   - ☐ `n = carre(int n);`
   - ☐ `int n = carre();`
6. Si cet avertissement apparaît à la compilation :
   `warning: implicit declaration of function ‘max’`
   , que doit-on chercher dans le programme ?
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une fonction déclarée mais non définie
   - ☐ une fonction appelée avant sa déclaration
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
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
   - ☐ 15
   - ☐ 6
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 :
   - ☐ 4
   - ☐ 0
   - ☐ 8
   - ☐ 16
9. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse harmonique
   - ☐ analyse sémantique
   - ☐ analyse lexicale
   - ☐ analyse syntaxique
10. 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
11. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

    appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

    L'affichage dans le main est le suivant :
    - ☐ `f(a,b)=8, a=8, b=5`
    - ☐ `f(a,b)=8, a=3, b=5`
    - ☐ `f(a,b)=13, a=8, b=5`
    - ☐ `f(3,5)=8, a=3, b=5`
12. L'écriture $\underline{101}$ en binaire correspond au nombre naturel :
    - ☐ 3
    - ☐ 4
    - ☐ 5
    - ☐ 101

13. 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

14. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

- ☐ `i`
- ☐ `A`
- ☐ `ABCDEF`
- ☐ `cccccc`

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  }
```

- ☐ 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

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

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

17. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :

- ☐ 1
- ☐ 0
- ☐ 0.5
- ☐ 1.5

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 :

- ☐ 8
- ☐ 0
- ☐ 16
- ☐ 4

19. Vous utilisez une boucle `while` quand :

- ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
- ☐ vous n'avez pas déclaré de fonction
- ☐ vous avez déjà fait un `for` dans le même programme principal
- ☐ l'incrément de la variable de boucle n'est pas 1

20. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

- ☐ `char c;`
- ☐ `int char;`
- ☐ `char "c";`
- ☐ `char '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. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   ☐ il n'affiche rien
   ☐ il risque d'afficher bonjour à la place de coucou
   ☐ il ne compile pas
   ☐ il comporte une boucle infinie

2. 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

3. 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

4. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
   ☐ `printf("Valeur de n ? %g\n", n);`
   ☐ un débogueur
   ☐ `printf("Valeur de n ? %d\n", n);`
   ☐ `scanf("%d", &n);`

5. 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

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 :
   ☐ 16
   ☐ 3
   ☐ 20
   ☐ 6

7. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :
   ☐ 4
   ☐ 3
   ☐ 0

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
   ☐ 0
   ☐ 8

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

11. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que `x` est une variable réelle définie et initialisée, il est incorrect d'écrire :
    ☐ `x = racine(2/3);`
    ☐ `x = racine(racine(x)*racine(x));`
    ☐ `x - 1 = racine(x);`
    ☐ `x = racine(x * x) - racine(x);`

12. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :
    ☐ `int n = pgcd();`
    ☐ `int pgcd(2);`
    ☐ `n = pgcd(int p, int q);`
    ☐ `n = pgcd(n, 3);`

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

14. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 5
- ☐ 0
- ☐ 1
- ☐ 4

15. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :

- ☐ `void afficher_date(struct date_s d);`
- ☐ `struct date_s afficher_date(struct date_s d);`
- ☐ `void afficher_date(date_s d);`
- ☐ `int afficher_date(date_s d);`

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

17. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

- ☐ un ordre quelconque
- ☐ dans lequel vous avez déclaré ces fonction
- ☐ dans lequel ces fonctions sont appelées dans le main
- ☐ alphabétique

18. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?

- ☐ `A && B`
- ☐ `!(!A || B) == (A && !B)`
- ☐ `(A == TRUE) && (B == TRUE)`
- ☐ `(!A || B)`

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

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

20. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :

- ☐ `int afficher_menu(int char);`
- ☐ `char afficher_menu(printf("menu"));`
- ☐ `int afficher_menu();`
- ☐ `void afficher_menu();`
- ☐ `double afficher_menu();`

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
   - ☐ ouvrir un fichier texte
   - ☐ créer un répertoire
2. Vous utilisez une boucle `while` quand :
   - ☐ vous n'avez pas déclaré de fonction
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ vous avez déjà fait un `for` dans le même programme principal
   - ☐ l'incrément de la variable de boucle n'est pas 1
3. Le code suivant :

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

   affichera :
   - ☐ rien
   - ☐ `Mineur`
   - ☐ `Majeur`
   - ☐ `Mineur`
     `Majeur`
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 variable non déclarée
   - ☐ une faute de frappe dans un appel de fonction
5. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

   Alors l'expression `f(1)` prendra la valeur :
   - ☐ 0
   - ☐ 5
   - ☐ 4
   - ☐ 1
6. 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
7. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ begin
   - ☐ include
   - ☐ init
   - ☐ main
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
   - ☐ comportent une erreur qui ne sera pas détectée
   - ☐ ne comportent aucune erreur
   - ☐ comportent une erreur qui sera détectée au cours de l'édition de lien
9. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
   - ☐ `exposant(double x, int n, int r);`
   - ☐ `double exposant(double x, int n);`
   - ☐ `int exposant(double n, int x);`
   - ☐ `void exposant(double x^n);`
10. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
    - ☐ `void pgcd(int x, int y);`
    - ☐ `int pgcd(int x, int x);`
    - ☐ `int pgcd(int x, y);`
    - ☐ `int pgcd(int y, int x);`
11. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

    appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

    L'affichage dans le main est le suivant :
    - ☐ `f(3,5)=8, a=3, b=5`
    - ☐ `f(a,b)=8, a=3, b=5`
    - ☐ `f(a,b)=13, a=8, b=5`
    - ☐ `f(a,b)=8, a=8, b=5`

12. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
    - ☐ `saisie_utilisateur(scanf(%d));`
    - ☐ `int saisie_utilisateur();`
    - ☐ `void saisie_utilisateur(char c);`
    - ☐ `void saisie_utilisateur(int n);`
13. 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
14. 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`
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
    - ☐ 4 3 2 1
    - ☐ 1 2 3 4

16. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

    Alors l'affichage sera :
    - ☐ `cccccc`
    - ☐ `ABCDEF`
    - ☐ `A`
    - ☐ `i`

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 = 5`
    - ☐ `j = 0`
    - ☐ `j = 4`

18. Une variable booléenne est un variable :
    - ☐ NaN (not a number, qui n'est pas un nombre)
    - ☐ à laquelle une valeur vient d'être affectée
    - ☐ qui est vraie ou fausse
    - ☐ jamais nulle
    - ☐ réelle positive

19. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

    Alors l'expression `f(0)` prendra la valeur :
    - ☐ 3
    - ☐ 4
    - ☐ 0

20. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
    - ☐ `int afficher_date(date_s d);`
    - ☐ `struct date_s afficher_date(struct date_s d);`
    - ☐ `void afficher_date(struct date_s d);`
    - ☐ `void afficher_date(date_s 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. 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`
2. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse syntaxique
   - ☐ analyse lexicale
   - ☐ analyse harmonique
   - ☐ analyse sémantique
3. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

   Alors l'affichage sera :
   - ☐ `A`
   - ☐ `ABCDEF`
   - ☐ `cccccc`
   - ☐ `i`
4. 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
5. 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"
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 :
   - ☐ certaines données de la mémoire de travail
   - ☐ les fichiers du disque
   - ☐ en temps d'accès
   - ☐ des processus
7. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

   Alors pour tester l'égalité de a et de b on utilise la condition :
   - ☐ `(a.n == b.n) && (a.x == b.x)`
   - ☐ `a{n, x} == b{n, x}`
   - ☐ `a = b`
   - ☐ `a == b`
8. L'écriture 101 en binaire correspond au nombre naturel :
   - ☐ 101
   - ☐ 3
   - ☐ 5
   - ☐ 4
9. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
   - ☐ `void saisie_utilisateur(int n);`
   - ☐ `int saisie_utilisateur();`
   - ☐ `saisie_utilisateur(scanf(%d));`
   - ☐ `void saisie_utilisateur(char c);`
10. Vous utilisez une boucle `while` quand :
    - ☐ vous n'avez pas déclaré de fonction
    - ☐ vous avez déjà fait un `for` dans le même programme principal
    - ☐ l'incrément de la variable de boucle n'est pas 1
    - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
11. Si cet avertissement apparaît à la compilation :
    `warning: implicit declaration of function 'max'` , que doit-on chercher dans le programme ?
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction déclarée mais non définie
    - ☐ une fonction appelée avant sa déclaration
12. Avant de faire appel à une fonction il est nécessaire de :
    - ☐ l'avoir déclarée
    - ☐ l'avoir déclarée et définie
    - ☐ avoir défini une constante symbolique de la taille de cette fonction
    - ☐ l'avoir définie
13. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(a,b)=13, a=8, b=5`

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=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)`
15. Le langage C est un langage
    - ☐ lu, écrit, parlé
    - ☐ composé
    - ☐ compilé
    - ☐ interprété
16. Le code suivant :

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

affichera :

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

17. L'écriture 111 en binaire correspond au nombre naturel :
    - ☐ 8
    - ☐ 7
    - ☐ 111
    - ☐ 3
18. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 4
- ☐ 5
- ☐ 1
- ☐ 0

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 :

- ☐ 0
- ☐ 4
- ☐ 8
- ☐ 16

20. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
    - ☐ `int n = carre();`
    - ☐ `n = carre(n);`
    - ☐ `n = carre(int n);`
    - ☐ `int carre(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. 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

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
   - ☐ 16
   - ☐ 6
   - ☐ 3

3. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   - ☐ 5
   - ☐ 0
   - ☐ 7

4. 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
   - ☐ 0 1 2 3 4
   - ☐ 4 3 2 1

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

6. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ dans lequel ces fonctions sont appelées dans le main
   - ☐ alphabétique
   - ☐ un ordre quelconque
   - ☐ dans lequel vous avez déclaré ces fonction

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

8. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
   - ☐ `x = racine(racine(x)*racine(x));`
   - ☐ `x = racine(x * x) - racine(x);`
   - ☐ `x - 1 = racine(x);`
   - ☐ `x = racine(2/3);`

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 2 4 6
   - ☐ 0 1 2 3 4 5 6 7

10. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
    - ☐ `int mccarthy(int 2);`
    - ☐ `n = mccarthy(p, q);`
    - ☐ `x = mccarthy(n);`
    - ☐ `n = mccarthy();`

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 case mémoire interne au processeur qui sera manipulée directement lors des calculs
    - ☐ une unité de calcul spécialisée de l'ordinateur

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

13. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=8, b=5`

14. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `(n<=a) && (n<=b)`
- ☐ `(a<n) || (n>b)`
- ☐ `a<=n<=b`
- ☐ `(a<=n) && (n<=b)`

15. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :

- ☐ un débogueur
- ☐ `printf("Valeur de n ? %d\n", n);`
- ☐ `scanf("%d", &n);`
- ☐ `printf("Valeur de n ? %g\n", n);`

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

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

17. Le code suivant :

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

affichera :

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

18. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :

- ☐ 0
- ☐ 0.5
- ☐ 1
- ☐ 1.5

19. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

Alors l'affichage sera :

- ☐ `b`
- ☐ `B`
- ☐ `A`
- ☐ `C`

20. Si cette erreur apparaît à la compilation :
`erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?

- ☐ un désaccord entre la déclaration et la définition d'une fonction
- ☐ une fonction déclarée mais non définie
- ☐ une fonction appelée avant sa déclaration
- ☐ une directive préprocesseur `#include` manquante

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. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `(a<n) || (n>b)`
- ☐ `(a<=n) && (n<=b)`
- ☐ `(n<=a) && (n<=b)`
- ☐ `a<=n<=b`

2. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :

- ☐ `x = racine(2/3);`
- ☐ `x = racine(racine(x)*racine(x));`
- ☐ `x = racine(x * x) - racine(x);`
- ☐ `x - 1 = racine(x);`

3. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

Alors l'affichage sera :

- ☐ `b`
- ☐ `A`
- ☐ `C`
- ☐ `B`

4. 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

5. 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;`

6. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :

- ☐ `void saisie_utilisateur(char c);`
- ☐ `int saisie_utilisateur();`
- ☐ `saisie_utilisateur(scanf(%d));`
- ☐ `void saisie_utilisateur(int n);`

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`
- ☐ `0 1 2 0 1 2`
- ☐ `1 2 1 2 3`

8. 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

9. Avant de faire appel à une fonction il est nécessaire de :

- ☐ l'avoir définie
- ☐ avoir défini une constante symbolique de la taille de cette fonction
- ☐ l'avoir déclarée
- ☐ l'avoir déclarée et définie

10. 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

11. Le type des réels en C est :

- ☐ `double`
- ☐ `real`
- ☐ `char`
- ☐ `int`

12. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :

- ☐ `toto_s n, x;`

- ☐ `int toto.n = 3;`
- ☐ `struct toto_s toto;`
- ☐ `int struct toto_s = {3, -1e10};`
- ☐ `toto_s struct z = {3, 0.5};`

13. 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
14. 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
15. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
    - ☐ `int factorielle(int 2);`
    - ☐ `n = factorielle();`
    - ☐ `n = factorielle(p, q);`
    - ☐ `printf("%d", factorielle(n));`
16. 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)
17. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
    - ☐ `char 'c';`
    - ☐ `char c;`
    - ☐ `char "c";`
    - ☐ `int char;`
18. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ un ordre quelconque
    - ☐ dans lequel vous avez déclaré ces fonction
    - ☐ dans lequel ces fonctions sont appelées dans le main
    - ☐ alphabétique
19. Sous unix (ou linux), pour créer un répertoire TP4 dans le répertoire courant on peut utiliser la commande :
    - ☐ `yppasswd`
    - ☐ `kwrite TP4`
    - ☐ `mkdir TP4`
    - ☐ `new TP4`
20. Un programme en langage C doit comporter une et une seule définition de la fonction :
    - ☐ include
    - ☐ begin
    - ☐ init
    - ☐ main

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  }
```

   - ☐ 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

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. Une variable booléenne est un variable :

   - ☐ à laquelle une valeur vient d'être affectée
   - ☐ NaN (not a number, qui n'est pas un nombre)
   - ☐ réelle positive
   - ☐ qui est vraie ou fausse
   - ☐ jamais nulle

4. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

   Quelle est la condition `cond` :

   - ☐ `a<=n<=b`
   - ☐ `(a<n) || (n>b)`
   - ☐ `(n<=a) && (n<=b)`
   - ☐ `(a<=n) && (n<=b)`

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. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

   - ☐ `char "c";`
   - ☐ `char c;`
   - ☐ `int char;`
   - ☐ `char 'c';`

7. 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

8. 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>`

9. Vous utilisez une boucle `while` quand :

   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous avez déjà fait un `for` dans le même programme principal
   - ☐ vous n'avez pas déclaré de fonction

10. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :

    - ☐ `int factorielle(int x);`
    - ☐ `int factorielle(double n);`
    - ☐ `int factorielle();`
    - ☐ `struct int factorielle(int n);`

11. 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 tab[] = 5;`
    - ☐ `int toto[5];`

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
- ☐ 16
- ☐ 6
- ☐ 3

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. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
    - ☐ `n = mccarthy();`
    - ☐ `x = mccarthy(n);`
    - ☐ `int mccarthy(int 2);`
    - ☐ `n = mccarthy(p, q);`
15. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
    - ☐ `struct date_s afficher_date(struct date_s d);`
    - ☐ `void afficher_date(date_s d);`
    - ☐ `void afficher_date(struct date_s d);`
    - ☐ `int afficher_date(date_s d);`
16. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=8, b=5`

17. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses blocs
    - ☐ ses chants
    - ☐ ses champs
    - ☐ ses cases
18. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 4
- ☐ 1
- ☐ 5
- ☐ 0

19. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
    - ☐ `saisie_utilisateur(scanf(%d));`
    - ☐ `void saisie_utilisateur(int n);`
    - ☐ `void saisie_utilisateur(char c);`
    - ☐ `int saisie_utilisateur();`
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  }
```

- ☐ 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

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 variable non déclarée
   - ☐ une faute de frappe dans un appel de fonction
   - ☐ un caractère interdit en C
   - ☐ une directive préprocesseur `#include` manquante
2. Avant de faire appel à une fonction il est nécessaire de :
   - ☐ l'avoir déclarée et définie
   - ☐ avoir défini une constante symbolique de la taille de cette fonction
   - ☐ l'avoir déclarée
   - ☐ l'avoir définie
3. 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
4. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
   - ☐ `char c;`
   - ☐ `char "c";`
   - ☐ `char 'c';`
   - ☐ `int char;`
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 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
6. 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
7. Si cette erreur apparaît à la compilation : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une fonction déclarée mais non définie
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une directive préprocesseur `#include` manquante
8. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
   - ☐ `struct int factorielle(int n);`
   - ☐ `int factorielle(int x);`
   - ☐ `int factorielle();`
   - ☐ `int factorielle(double n);`
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 0 1 0 1`
   - ☐ `0 0 1 1 2 2`
   - ☐ `1 2 3 1 2`
   - ☐ `0 1 2 0 1 2`
10. L'écriture 111 en binaire correspond au nombre naturel :
    - ☐ 8
    - ☐ 7
    - ☐ 111
    - ☐ 3
11. Le type des réels en C est :
    - ☐ `int`
    - ☐ `real`
    - ☐ `char`
    - ☐ `double`
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=%x et y=%y\n");`
    - ☐ `printf("x=%d et y=%d\n",x,y);`
    - ☐ `printf("x=%d et y=%d\n",x y);`
13. L'écriture 101 en binaire correspond au nombre naturel :
    - ☐ 101
    - ☐ 4
    - ☐ 3
    - ☐ 5
14. Vous utilisez une boucle `while` quand :
    - ☐ l'incrément de la variable de boucle n'est pas 1
    - ☐ vous avez déjà fait un `for` dans le même programme principal
    - ☐ vous n'avez pas déclaré de fonction
    - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance

15. Le code suivant :

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

affichera :

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

16. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :

- ☐ `int carre(2);`
- ☐ `int n = carre();`
- ☐ `n = carre(n);`
- ☐ `n = carre(int n);`

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 :

- ☐ 3
- ☐ 16
- ☐ 20
- ☐ 6

18. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

- ☐ dans lequel ces fonctions sont appelées dans le main
- ☐ alphabétique
- ☐ un ordre quelconque
- ☐ dans lequel vous avez déclaré ces fonction

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 `****`
- ☐ le programme affiche x

20. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :

- ☐ `saisie_utilisateur(scanf(%d));`
- ☐ `void saisie_utilisateur(int n);`
- ☐ `void saisie_utilisateur(char c);`
- ☐ `int saisie_utilisateur();`

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 :

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

2. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

- ☐ `n = factorielle(p, q);`
- ☐ `printf("%d", factorielle(n));`
- ☐ `int factorielle(int 2);`
- ☐ `n = factorielle();`

3. L'écriture 111 en binaire correspond au nombre naturel :

- ☐ 111
- ☐ 8
- ☐ 3
- ☐ 7

4. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :

- ☐ 1.5
- ☐ 1
- ☐ 0.5
- ☐ 0

5. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 0
- ☐ 4
- ☐ 3

6. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?

- ☐ `(A == TRUE) && (B == TRUE)`
- ☐ `A && B`
- ☐ `!(!A || B) == (A && !B)`
- ☐ `(!A || B)`

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

8. L'écriture 101 en binaire correspond au nombre naturel :

- ☐ 5
- ☐ 4
- ☐ 101
- ☐ 3

9. 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`

10. 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

11. Vous utilisez une boucle `while` quand :

- ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
- ☐ vous n'avez pas déclaré de fonction
- ☐ l'incrément de la variable de boucle n'est pas 1
- ☐ vous avez déjà fait un `for` dans le même programme principal

12. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il ne compile pas
- ☐ il n'affiche rien
- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il comporte une boucle infinie

13. Avant de faire appel à une fonction il est nécessaire de :
    - ☐ l'avoir déclarée et définie
    - ☐ l'avoir déclarée
    - ☐ avoir défini une constante symbolique de la taille de cette fonction
    - ☐ l'avoir définie

14. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
    - ☐ `char 'c';`
    - ☐ `int char;`
    - ☐ `char "c";`
    - ☐ `char c;`

15. 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

16. 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

17. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
    - ☐ `int factorielle();`
    - ☐ `int factorielle(double n);`
    - ☐ `int factorielle(int x);`
    - ☐ `struct int factorielle(int n);`

18. 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

19. 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

20. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `a == b`
- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a{n, x} == b{n, x}`
- ☐ `a = b`

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'écriture 111 en binaire correspond au nombre naturel :
   - ☐ 3
   - ☐ 7
   - ☐ 111
   - ☐ 8

2. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   - ☐ `f(a,b)=8, a=8, b=5`
   - ☐ `f(a,b)=13, a=8, b=5`
   - ☐ `f(a,b)=8, a=3, b=5`
   - ☐ `f(3,5)=8, a=3, b=5`

3. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une fonction déclarée mais non définie
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ un désaccord entre la déclaration et la définition d'une fonction

4. 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;`

5. Avant de faire appel à une fonction il est nécessaire de :
   - ☐ l'avoir déclarée
   - ☐ avoir défini une constante symbolique de la taille de cette fonction
   - ☐ l'avoir déclarée et définie
   - ☐ l'avoir définie

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

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
   - ☐ 0 1 2 3 4
   - ☐ 4 3 2 1 0

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

9. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
   - ☐ `void exposant(double x^n);`
   - ☐ `exposant(double x, int n, int r);`
   - ☐ `double exposant(double x, int n);`
   - ☐ `int exposant(double n, int x);`

10. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
    - ☐ `char c;`
    - ☐ `char 'c';`
    - ☐ `int char;`
    - ☐ `char "c";`

11. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

    - ☐ il comporte une boucle infinie
    - ☐ il ne compile pas
    - ☐ il risque d'afficher bonjour à la place de coucou
    - ☐ il n'affiche rien

12. 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

13. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
    - ☐ `x = racine(x * x) - racine(x);`
    - ☐ `x = racine(racine(x)*racine(x));`
    - ☐ `x = racine(2/3);`
    - ☐ `x - 1 = racine(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`
    - ☐ `1 2 1 2 3`
    - ☐ `0 0 0 1 1 1`
    - ☐ `0 1 0 1 0 1 0 1`

15. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
    - ☐ `(A == TRUE) && (B == TRUE)`
    - ☐ `!(!A || B) == (A && !B)`
    - ☐ `A && B`
    - ☐ `(!A || B)`

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 x
    - ☐ le programme affiche `****`

17. 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`

18. 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

19. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

    Alors l'expression `g(0)` prendra la valeur :
    - ☐ 7
    - ☐ 5
    - ☐ 0

20. Le langage C est un langage
    - ☐ compilé
    - ☐ interprété
    - ☐ composé
    - ☐ 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. 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 la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
   - ☐ `n = mccarthy();`
   - ☐ `x = mccarthy(n);`
   - ☐ `int mccarthy(int 2);`
   - ☐ `n = mccarthy(p, q);`
3. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
   - ☐ 1.5
   - ☐ 0
   - ☐ 1
   - ☐ 0.5
4. Si cette erreur apparaît à la compilation : `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
   - ☐ un point-virgule manquant
   - ☐ une accolade en trop
   - ☐ un point-virgule en trop
   - ☐ une accolade manquante
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 y vaut 5
   - ☐ la variable x vaut 5 et la variable y vaut 0
   - ☐ la variable x vaut 0
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 :
   - ☐ 10
   - ☐ 6
   - ☐ 0
   - ☐ 15
7. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   - ☐ `f(a,b)=13, a=8, b=5`
   - ☐ `f(a,b)=8, a=8, b=5`
   - ☐ `f(3,5)=8, a=3, b=5`
   - ☐ `f(a,b)=8, a=3, b=5`
8. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
   - ☐ `exposant(double x, int n, int r);`
   - ☐ `int exposant(double n, int x);`
   - ☐ `double exposant(double x, int n);`
   - ☐ `void exposant(double x^n);`
9. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
   - ☐ `printf("Valeur de n ? %g\n", n);`
   - ☐ `scanf("%d", &n);`
   - ☐ un débogueur
   - ☐ `printf("Valeur de n ? %d\n", n);`
10. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

    - ☐ il risque d'afficher bonjour à la place de coucou
    - ☐ il comporte une boucle infinie
    - ☐ il ne compile pas
    - ☐ il n'affiche rien
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 y vaut 5
    - ☐ la variable x vaut 3
    - ☐ la variable x vaut 5 et la variable y vaut 3
    - ☐ le programme affiche "Faux"

12. 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

13. Un programme en langage C doit comporter une et une seule définition de la fonction :
    - ☐ main
    - ☐ include
    - ☐ begin
    - ☐ init
14. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
    - ☐ analyse sémantique
    - ☐ analyse syntaxique
    - ☐ analyse harmonique
    - ☐ analyse lexicale
15. Le code suivant :

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

affichera :

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

16. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 7
- ☐ 0
- ☐ 5

17. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ alphabétique
    - ☐ dans lequel ces fonctions sont appelées dans le main
    - ☐ un ordre quelconque
    - ☐ dans lequel vous avez déclaré ces fonction
18. L'écriture 101 en binaire correspond au nombre naturel :
    - ☐ 101
    - ☐ 3
    - ☐ 4
    - ☐ 5
19. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

- ☐ `cccccc`
- ☐ `i`
- ☐ `A`
- ☐ `ABCDEF`

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 :

- ☐ 8
- ☐ 4
- ☐ 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. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :

- ☐ `toto_s struct z = {3, 0.5};`
- ☐ `toto_s n, x;`
- ☐ `struct toto_s toto;`
- ☐ `int struct toto_s = {3, -1e10};`
- ☐ `int toto.n = 3;`

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. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

- ☐ `char 'c';`
- ☐ `int char;`
- ☐ `char "c";`
- ☐ `char c;`

4. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 0
- ☐ 5
- ☐ 7

5. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?

- ☐ `(!A || B)`
- ☐ `A && B`
- ☐ `!(!A || B) == (A && !B)`
- ☐ `(A == TRUE) && (B == TRUE)`

6. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

- ☐ dans lequel ces fonctions sont appelées dans le main
- ☐ alphabétique
- ☐ dans lequel vous avez déclaré ces fonction
- ☐ un ordre quelconque

7. 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 = 0`
- ☐ `j = %d`

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
- ☐ 4 3 2 1
- ☐ 0 1 2 3 4

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

10. 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`

11. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

- ☐ `cccccc`
- ☐ `i`
- ☐ `A`
- ☐ `ABCDEF`

12. Le type des réels en C est :
    - ☐ `int`
    - ☐ `real`
    - ☐ `char`
    - ☐ `double`

13. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que `x` est une variable réelle définie et initialisée, il est incorrect d'écrire :
    - ☐ `x = racine(2/3);`
    - ☐ `x - 1 = racine(x);`
    - ☐ `x = racine(racine(x)*racine(x));`
    - ☐ `x = racine(x * x) - racine(x);`
14. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

    Alors l'expression `f(0)` prendra la valeur :
    - ☐ 3
    - ☐ 0
    - ☐ 4
15. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses cases
    - ☐ ses chants
    - ☐ ses blocs
    - ☐ ses champs
16. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
    - ☐ `int afficher_menu(int char);`
    - ☐ `int afficher_menu();`
    - ☐ `void afficher_menu();`
    - ☐ `char afficher_menu(printf("menu"));`
    - ☐ `double afficher_menu();`
17. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

    Alors l'affichage sera :
    - ☐ `C`
    - ☐ `b`
    - ☐ `A`
    - ☐ `B`
18. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
    - ☐ `x = mccarthy(n);`
    - ☐ `n = mccarthy(p, q);`
    - ☐ `n = mccarthy();`
    - ☐ `int mccarthy(int 2);`
19. Un bit est :
    - ☐ l'instruction qui met fin à un programme
    - ☐ un chiffre binaire (0 ou 1)
    - ☐ la longueur d'un mot mémoire
    - ☐ un battement d'horloge processeur
20. 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`

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 type des réels en C est :
   - ☐ `char`
   - ☐ `int`
   - ☐ `double`
   - ☐ `real`

2. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

   précède la fonction `main()`, alors on peut écrire en début de `main()` :
   - ☐ `int struct toto_s = {3, -1e10};`
   - ☐ `int toto.n = 3;`
   - ☐ `toto_s n, x;`
   - ☐ `toto_s struct z = {3, 0.5};`
   - ☐ `struct toto_s toto;`

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. Vous utilisez une boucle `while` quand :
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ vous avez déjà fait un `for` dans le même programme principal
   - ☐ vous n'avez pas déclaré de fonction
   - ☐ l'incrément de la variable de boucle n'est pas 1

5. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
   - ☐ `x = mccarthy(n);`
   - ☐ `n = mccarthy();`
   - ☐ `n = mccarthy(p, q);`
   - ☐ `int mccarthy(int 2);`

6. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :
   - ☐ 0
   - ☐ 4
   - ☐ 3

7. Si cette erreur apparaît à la compilation :
   `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une fonction déclarée mais non définie
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une directive préprocesseur `#include` manquante

8. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

   Alors l'affichage sera :
   - ☐ `C`
   - ☐ `b`
   - ☐ `A`
   - ☐ `B`

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

10. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

    - ☐ il ne compile pas
    - ☐ il comporte une boucle infinie
    - ☐ il n'affiche rien
    - ☐ il risque d'afficher bonjour à la place de coucou

11. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
    - ☐ `(!A || B)`
    - ☐ `(A == TRUE) && (B == TRUE)`
    - ☐ `!(!A || B) == (A && !B)`
    - ☐ `A && B`

12. Le code suivant :

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

affichera :

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

13. 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

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
    - ☐ de doubler la mémoire disponible
    - ☐ d'entretenir l'illusion que les processus tournent en parallèle

15. 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`

16. 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`

17. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :
    - ☐ `ABCDEF`
    - ☐ `A`
    - ☐ `i`
    - ☐ `cccccc`

18. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
    - ☐ `char "c";`
    - ☐ `char c;`
    - ☐ `char 'c';`
    - ☐ `int char;`

19. 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;`

20. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
    - ☐ `int exposant(double n, int x);`
    - ☐ `void exposant(double x^n);`
    - ☐ `double exposant(double x, int n);`
    - ☐ `exposant(double x, int n, int r);`

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 une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
   - ☐ `void saisie_utilisateur(int n);`
   - ☐ `saisie_utilisateur(scanf(%d));`
   - ☐ `void saisie_utilisateur(char c);`
   - ☐ `int saisie_utilisateur();`
2. 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>`
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 :
   - ☐ certaines données de la mémoire de travail
   - ☐ des processus
   - ☐ les fichiers du disque
   - ☐ en temps d'accès
4. 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
5. 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
6. 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
7. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
   - ☐ `x = racine(2/3);`
   - ☐ `x - 1 = racine(x);`
   - ☐ `x = racine(racine(x)*racine(x));`
   - ☐ `x = racine(x * x) - racine(x);`
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. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
   - ☐ 3
   - ☐ 7
   - ☐ 8
   - ☐ 111
10. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

    - ☐ il ne compile pas
    - ☐ il risque d'afficher bonjour à la place de coucou
    - ☐ il n'affiche rien
    - ☐ il comporte une boucle infinie
11. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

    Alors l'expression `g(0)` prendra la valeur :
    - ☐ 7
    - ☐ 0
    - ☐ 5
12. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
    - ☐ `(!A || B)`
    - ☐ `!(!A || B) == (A && !B)`
    - ☐ `A && B`
    - ☐ `(A == TRUE) && (B == TRUE)`

13. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 0
- ☐ 3
- ☐ 4

14. 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

15. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 5
- ☐ 1
- ☐ 0
- ☐ 4

16. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
    - ☐ `int afficher_menu(int char);`
    - ☐ `char afficher_menu(printf("menu"));`
    - ☐ `double afficher_menu();`
    - ☐ `int afficher_menu();`
    - ☐ `void afficher_menu();`

17. Une variable booléenne est un variable :
    - ☐ jamais nulle
    - ☐ réelle positive
    - ☐ NaN (not a number, qui n'est pas un nombre)
    - ☐ à laquelle une valeur vient d'être affectée
    - ☐ qui est vraie ou fausse

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'édition de liens
    - ☐ l'analyse harmonique
    - ☐ l'analyse sémantique

19. 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

20. Si cet avertissement apparaît à la compilation :
`warning: implicit declaration of function ‘max’`
, que doit-on chercher dans le programme ?
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction déclarée mais non définie

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 variable booléenne est un variable :
   - ☐ NaN (not a number, qui n'est pas un nombre)
   - ☐ qui est vraie ou fausse
   - ☐ réelle positive
   - ☐ jamais nulle
   - ☐ à laquelle une valeur vient d'être affectée

2. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :
   - ☐ 0
   - ☐ 3
   - ☐ 4

3. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ alphabétique
   - ☐ un ordre quelconque
   - ☐ dans lequel vous avez déclaré ces fonction
   - ☐ dans lequel ces fonctions sont appelées dans le main

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. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   - ☐ 5
   - ☐ 7
   - ☐ 0

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. Le code suivant :

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

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

8. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
   - ☐ `struct date_s afficher_date(struct date_s d);`
   - ☐ `void afficher_date(struct date_s d);`
   - ☐ `int afficher_date(date_s d);`
   - ☐ `void afficher_date(date_s d);`

9. Le type des réels en C est :
   - ☐ `char`
   - ☐ `real`
   - ☐ `int`
   - ☐ `double`

10. 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

11. Un fichier source est :
    - ☐ un document illisible pour les humains
    - ☐ un document de référence du système
    - ☐ 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

12. 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)

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

14. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
    - ☐ `double exposant(double x, int n);`
    - ☐ `int exposant(double n, int x);`
    - ☐ `void exposant(double x^n);`
    - ☐ `exposant(double x, int n, int r);`

15. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
    - ☐ `int factorielle(double n);`
    - ☐ `int factorielle();`
    - ☐ `struct int factorielle(int n);`
    - ☐ `int factorielle(int x);`

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

17. L'écriture 111 en binaire correspond au nombre naturel :
    - ☐ 111
    - ☐ 7
    - ☐ 8
    - ☐ 3

18. Avant de faire appel à une fonction il est nécessaire de :
    - ☐ l'avoir déclarée et définie
    - ☐ avoir défini une constante symbolique de la taille de cette fonction
    - ☐ l'avoir définie
    - ☐ l'avoir déclarée

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

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. 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

2. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
   - ☐ `n = factorielle(p, q);`
   - ☐ `printf("%d", factorielle(n));`
   - ☐ `n = factorielle();`
   - ☐ `int factorielle(int 2);`

3. L'écriture 101 en binaire correspond au nombre naturel :
   - ☐ 101
   - ☐ 5
   - ☐ 4
   - ☐ 3

4. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
   - ☐ `int pgcd(int x, int x);`
   - ☐ `void pgcd(int x, int y);`
   - ☐ `int pgcd(int x, y);`
   - ☐ `int pgcd(int y, int x);`

5. Le code suivant :

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

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

6. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :
   - ☐ 3
   - ☐ 4
   - ☐ 0

7. 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

8. 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`

9. 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

10. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il ne compile pas
   - ☐ il comporte une boucle infinie
   - ☐ il n'affiche rien
   - ☐ il risque d'afficher bonjour à la place de coucou

11. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
    - ☐ analyse lexicale
    - ☐ analyse harmonique
    - ☐ analyse sémantique
    - ☐ analyse syntaxique
12. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses cases
    - ☐ ses blocs
    - ☐ ses champs
    - ☐ ses chants
13. 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
14. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ dans lequel vous avez déclaré ces fonction
    - ☐ un ordre quelconque
    - ☐ dans lequel ces fonctions sont appelées dans le main
    - ☐ alphabétique
15. Le langage C est un langage
    - ☐ interprété
    - ☐ lu, écrit, parlé
    - ☐ compilé
    - ☐ composé
16. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

    Quelle est la condition `cond` :
    - ☐ `(n<=a) && (n<=b)`
    - ☐ `(a<=n) && (n<=b)`
    - ☐ `(a<n) || (n>b)`
    - ☐ `a<=n<=b`
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`
    - ☐ `1 2 1 2 3`
    - ☐ `0 1 2 0 1 2`
    - ☐ `0 0 0 1 1 1`
18. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
    - ☐ `exposant(double x, int n, int r);`
    - ☐ `void exposant(double x^n);`
    - ☐ `double exposant(double x, int n);`
    - ☐ `int exposant(double n, int x);`
19. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

    Alors l'expression `f(1)` prendra la valeur :
    - ☐ 5
    - ☐ 1
    - ☐ 0
    - ☐ 4
20. Si cette erreur apparaît à la compilation :
    `erreur: conflicting types for ‘max’` , que doit-on chercher dans le programme ?
    - ☐ une fonction déclarée mais non définie
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction appelée avant sa déclaration
    - ☐ un désaccord entre la déclaration et la définition d'une fonction

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 : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une fonction déclarée mais non définie
   - ☐ une fonction appelée avant sa déclaration
2. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses cases
   - ☐ ses chants
   - ☐ ses champs
   - ☐ ses blocs
3. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il ne compile pas
   - ☐ il comporte une boucle infinie
   - ☐ il risque d'afficher bonjour à la place de coucou
   - ☐ il n'affiche rien
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. 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
6. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :
   - ☐ 3
   - ☐ 4
   - ☐ 0
7. 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
8. 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
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 = 2`
   - ☐ le terminal affiche "Faux"
   - ☐ le terminal affiche `x = 5`
   - ☐ le terminal affiche `5`
10. 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
11. Un fichier source est :
    - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
    - ☐ un document qui doit être protégé
    - ☐ un document illisible pour les humains
    - ☐ un fichier texte qui sera traduit en instructions processeur
    - ☐ un document de référence du système
12. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
    - ☐ `int pgcd(int x, int x);`
    - ☐ `int pgcd(int x, y);`
    - ☐ `void pgcd(int x, int y);`
    - ☐ `int pgcd(int y, int x);`

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
- ☐ une gamme de fréquence de fonctionnement du processeur
- ☐ un composant qui contient la liste des fichiers du système

14. Vous utilisez une boucle `while` quand :

- ☐ vous n'avez pas déclaré de fonction
- ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
- ☐ vous avez déjà fait un `for` dans le même programme principal
- ☐ l'incrément de la variable de boucle n'est pas 1

15. 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

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

17. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :

- ☐ `void afficher_menu();`
- ☐ `double afficher_menu();`
- ☐ `char afficher_menu(printf("menu"));`
- ☐ `int afficher_menu(int char);`
- ☐ `int afficher_menu();`

18. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `(a<n) || (n>b)`
- ☐ `(n<=a) && (n<=b)`
- ☐ `a<=n<=b`
- ☐ `(a<=n) && (n<=b)`

19. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

- ☐ `char 'c';`
- ☐ `int char;`
- ☐ `char c;`
- ☐ `char "c";`

20. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :

- ☐ `n = mccarthy();`
- ☐ `x = mccarthy(n);`
- ☐ `n = mccarthy(p, q);`
- ☐ `int mccarthy(int 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. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
   - ☐ `n = mccarthy();`
   - ☐ `x = mccarthy(n);`
   - ☐ `int mccarthy(int 2);`
   - ☐ `n = mccarthy(p, q);`
2. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses cases
   - ☐ ses champs
   - ☐ ses chants
   - ☐ ses blocs
3. 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
4. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
   - ☐ `int factorielle(double n);`
   - ☐ `struct int factorielle(int n);`
   - ☐ `int factorielle();`
   - ☐ `int factorielle(int x);`
5. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
   - ☐ `(A == TRUE) && (B == TRUE)`
   - ☐ `A && B`
   - ☐ `(!A || B)`
   - ☐ `!(!A || B) == (A && !B)`
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 `5`
   - ☐ le terminal affiche "Faux"
   - ☐ le terminal affiche `x = 5`
   - ☐ le terminal affiche `x = 2`
7. Le langage C est un langage
   - ☐ compilé
   - ☐ lu, écrit, parlé
   - ☐ interprété
   - ☐ composé
8. Le code suivant :

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

   affichera :
   - ☐ rien
   - ☐ `Mineur`
   - ☐ `Majeur`
   - ☐ `Mineur`
     `Majeur`
9. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
   - ☐ `printf("Valeur de n ? %d\n", n);`
   - ☐ `scanf("%d", &n);`
   - ☐ un débogueur
   - ☐ `printf("Valeur de n ? %g\n", n);`
10. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

    Alors l'affichage sera :
    - ☐ `C`
    - ☐ `A`
    - ☐ `b`
    - ☐ `B`
11. 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
12. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function 'max'` , que doit-on chercher dans le programme ?
    - ☐ une fonction déclarée mais non définie
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction appelée avant sa déclaration
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
13. 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

14. 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

15. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :

- ☐ `int toto.n = 3;`
- ☐ `toto_s struct z = {3, 0.5};`
- ☐ `struct toto_s toto;`
- ☐ `toto_s n, x;`
- ☐ `int struct toto_s = {3, -1e10};`

16. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(a,b)=8, a=3, b=5`

17. 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

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

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

19. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :

- ☐ 7
- ☐ 111
- ☐ 3
- ☐ 8

20. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

- ☐ `char "c";`
- ☐ `char c;`
- ☐ `int char;`
- ☐ `char '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. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
   - ☐ `int pgcd(int y, int x);`
   - ☐ `int pgcd(int x, y);`
   - ☐ `void pgcd(int x, int y);`
   - ☐ `int pgcd(int x, int x);`
2. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :
   - ☐ 0
   - ☐ 4
   - ☐ 3
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 faute de frappe dans un appel de fonction
   - ☐ une directive préprocesseur `#include` manquante
4. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses chants
   - ☐ ses blocs
   - ☐ ses champs
   - ☐ ses cases
5. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
   - ☐ `int factorielle(int 2);`
   - ☐ `n = factorielle();`
   - ☐ `n = factorielle(p, q);`
   - ☐ `printf("%d", factorielle(n));`
6. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ main
   - ☐ include
   - ☐ init
   - ☐ begin
7. 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
8. Le code suivant :

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

   affichera :
   - ☐ `Majeur`
   - ☐ `Mineur`
   - ☐ `Mineur`
`Majeur`
   - ☐ rien
9. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il ne compile pas
   - ☐ il n'affiche rien
   - ☐ il comporte une boucle infinie
   - ☐ il risque d'afficher bonjour à la place de coucou
10. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
    - ☐ analyse lexicale
    - ☐ analyse harmonique
    - ☐ analyse syntaxique
    - ☐ analyse sémantique
11. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

    Alors l'expression `f(1)` prendra la valeur :
    - ☐ 1
    - ☐ 0
    - ☐ 5
    - ☐ 4
12. 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
13. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
    - ☐ `(!A || B)`
    - ☐ `A && B`
    - ☐ `!(!A || B) == (A && !B)`
    - ☐ `(A == TRUE) && (B == TRUE)`

14. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :

- ☐ `int mccarthy(int 2);`
- ☐ `n = mccarthy(p, q);`
- ☐ `n = mccarthy();`
- ☐ `x = mccarthy(n);`

15. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :

- ☐ `struct toto_s toto;`
- ☐ `int struct toto_s = {3, -1e10};`
- ☐ `toto_s n, x;`
- ☐ `int toto.n = 3;`
- ☐ `toto_s struct z = {3, 0.5};`

16. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :

- ☐ `void exposant(double x^n);`
- ☐ `exposant(double x, int n, int r);`
- ☐ `int exposant(double n, int x);`
- ☐ `double exposant(double x, int n);`

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 0
- ☐ 4 3 2 1

18. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `a == b`
- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a{n, x} == b{n, x}`
- ☐ `a = b`

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

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

- ☐ créer un répertoire
- ☐ changer de répertoire courant
- ☐ créer un fichier texte
- ☐ 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. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
   - ☐ `double exposant(double x, int n);`
   - ☐ `int exposant(double n, int x);`
   - ☐ `void exposant(double x^n);`
   - ☐ `exposant(double x, int n, int r);`
2. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
   - ☐ `n = mccarthy();`
   - ☐ `x = mccarthy(n);`
   - ☐ `int mccarthy(int 2);`
   - ☐ `n = mccarthy(p, q);`
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
   - ☐ un caractère interdit en C
   - ☐ une variable non déclarée
   - ☐ une faute de frappe dans un appel de fonction
4. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses chants
   - ☐ ses blocs
   - ☐ ses champs
   - ☐ ses cases
5. Si cette erreur apparaît à la compilation : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une fonction déclarée mais non définie
   - ☐ une directive préprocesseur `#include` manquante
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
   - ☐ la variable x vaut 0
   - ☐ le programme affiche "Faux"
   - ☐ la variable y vaut 5
7. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ dans lequel ces fonctions sont appelées dans le main
   - ☐ dans lequel vous avez déclaré ces fonction
   - ☐ alphabétique
   - ☐ un ordre quelconque
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
   - ☐ le programme affiche "Faux"
   - ☐ la variable x vaut 3
   - ☐ la variable y vaut 5
9. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
   - ☐ `void afficher_menu();`
   - ☐ `char afficher_menu(printf("menu"));`
   - ☐ `int afficher_menu();`
   - ☐ `double afficher_menu();`
   - ☐ `int afficher_menu(int char);`
10. Le langage C est un langage
    - ☐ composé
    - ☐ compilé
    - ☐ lu, écrit, parlé
    - ☐ interprété
11. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
    - ☐ `int pgcd(int x, int x);`
    - ☐ `int pgcd(int x, y);`
    - ☐ `void pgcd(int x, int y);`
    - ☐ `int pgcd(int y, int x);`
12. Le code suivant :

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

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

13. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :

- ☐ `int carre(2);`
- ☐ `int n = carre();`
- ☐ `n = carre(n);`
- ☐ `n = carre(int n);`

14. 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`

15. Le type des réels en C est :

- ☐ `int`
- ☐ `real`
- ☐ `double`
- ☐ `char`

16. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 7
- ☐ 5
- ☐ 0

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

18. 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`

19. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 3
- ☐ 4
- ☐ 0

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

- ☐ créer un répertoire
- ☐ ouvrir un fichier texte
- ☐ changer de répertoire courant
- ☐ 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. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `a<=n<=b`
- ☐ `(a<=n) && (n<=b)`
- ☐ `(n<=a) && (n<=b)`
- ☐ `(a<n) || (n>b)`

2. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(3,5)=8, a=3, b=5`

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

4. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :

- ☐ `toto_s n, x;`
- ☐ `struct toto_s toto;`
- ☐ `int struct toto_s = {3, -1e10};`
- ☐ `toto_s struct z = {3, 0.5};`
- ☐ `int toto.n = 3;`

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

6. Le type des réels en C est :
   - ☐ `real`
   - ☐ `char`
   - ☐ `int`
   - ☐ `double`

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

8. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
   - ☐ 111
   - ☐ 7
   - ☐ 8
   - ☐ 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=%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);`

10. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
```

```
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 0
- ☐ 3
- ☐ 4

11. Le code suivant :

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

affichera :

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

12. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :

- ☐ `void afficher_date(date_s d);`
- ☐ `void afficher_date(struct date_s d);`
- ☐ `int afficher_date(date_s d);`
- ☐ `struct date_s afficher_date(struct date_s d);`

13. 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;`

14. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :

- ☐ `void saisie_utilisateur(int n);`
- ☐ `saisie_utilisateur(scanf(%d));`
- ☐ `int saisie_utilisateur();`
- ☐ `void saisie_utilisateur(char c);`

15. Le code suivant :

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

affichera :

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

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

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

17. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

- ☐ un ordre quelconque
- ☐ alphabétique
- ☐ dans lequel ces fonctions sont appelées dans le main
- ☐ dans lequel vous avez déclaré ces fonction

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  }
```

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

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
- ☐ 20
- ☐ 16
- ☐ 3

20. Vous utilisez une boucle `while` quand :

- ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
- ☐ vous avez déjà fait un `for` dans le même programme principal
- ☐ l'incrément de la variable de boucle n'est pas 1
- ☐ vous n'avez pas déclaré de fonction

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. Si cet avertissement apparaît à la compilation :
`warning: implicit declaration of function ‘max’`
, que doit-on chercher dans le programme ?
   - ☐ une fonction déclarée mais non définie
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une fonction appelée avant sa déclaration

3. 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

4. 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

5. 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

6. Vous utilisez une boucle `while` quand :
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous n'avez pas déclaré de fonction
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ vous avez déjà fait un `for` dans le même programme principal

7. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
   - ☐ `n = carre(n);`
   - ☐ `n = carre(int n);`
   - ☐ `int n = carre();`
   - ☐ `int carre(2);`

8. 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`

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 document qui doit être protégé
   - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur

10. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `(n<=a) && (n<=b)`
- ☐ `(a<n) || (n>b)`
- ☐ `(a<=n) && (n<=b)`
- ☐ `a<=n<=b`

11. 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`

12. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :

    ☐ `x = mccarthy(n);`

    ☐ `int mccarthy(int 2);`

    ☐ `n = mccarthy(p, q);`

    ☐ `n = mccarthy();`

13. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

    ☐ il n'affiche rien

    ☐ il ne compile pas

    ☐ il comporte une boucle infinie

    ☐ il risque d'afficher bonjour à la place de coucou

14. 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>`

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 harmonique

    ☐ l'analyse des entrées clavier

    ☐ l'analyse sémantique

    ☐ l'édition de liens

16. Le type des réels en C est :

    ☐ `char`

    ☐ `real`

    ☐ `int`

    ☐ `double`

17. Un enregistrement permet de grouper plusieurs valeurs dans :

    ☐ ses chants

    ☐ ses champs

    ☐ ses cases

    ☐ ses blocs

18. Si cette erreur apparaît à la compilation :
    `erreur: conflicting types for ‘max’` , que doit-on chercher dans le programme ?

    ☐ un désaccord entre la déclaration et la définition d'une fonction

    ☐ une fonction déclarée mais non définie

    ☐ une fonction appelée avant sa déclaration

    ☐ une directive préprocesseur `#include` manquante

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

    ☐ main

    ☐ begin

    ☐ include

    ☐ init

20. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

    ☐ dans lequel vous avez déclaré ces fonction

    ☐ un ordre quelconque

    ☐ alphabétique

    ☐ dans lequel ces fonctions sont appelées dans le main

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. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

- ☐ `cccccc`
- ☐ `A`
- ☐ `i`
- ☐ `ABCDEF`

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. Le langage C est un langage
   - ☐ compilé
   - ☐ lu, écrit, parlé
   - ☐ interprété
   - ☐ composé
4. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 4
- ☐ 3
- ☐ 0

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

7. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

Alors l'affichage sera :

- ☐ `B`
- ☐ `b`
- ☐ `C`
- ☐ `A`

8. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
   - ☐ `scanf("%d", &n);`
   - ☐ un débogueur
   - ☐ `printf("Valeur de n ? %g\n", n);`
   - ☐ `printf("Valeur de n ? %d\n", n);`
9. Une variable booléenne est un variable :
   - ☐ jamais nulle
   - ☐ NaN (not a number, qui n'est pas un nombre)
   - ☐ réelle positive
   - ☐ qui est vraie ou fausse
   - ☐ à laquelle une valeur vient d'être affectée
10. Le type des réels en C est :
    - ☐ `double`
    - ☐ `real`
    - ☐ `int`
    - ☐ `char`
11. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 7
- ☐ 5
- ☐ 0

12. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses chants
    - ☐ ses champs
    - ☐ ses cases
    - ☐ ses blocs

13. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `(a<n) || (n>b)`
- ☐ `(n<=a) && (n<=b)`
- ☐ `a<=n<=b`
- ☐ `(a<=n) && (n<=b)`

14. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :

- ☐ `struct date_s afficher_date(struct date_s d);`
- ☐ `void afficher_date(struct date_s d);`
- ☐ `int afficher_date(date_s d);`
- ☐ `void afficher_date(date_s d);`

15. 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)

16. 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

17. Le code suivant :

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

affichera :

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

18. 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`

19. 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

20. Si cette erreur apparaît à la compilation : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?

- ☐ une fonction déclarée mais non définie
- ☐ une fonction appelée avant sa déclaration
- ☐ un désaccord entre la déclaration et la définition d'une fonction
- ☐ une directive préprocesseur `#include` manquante

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
☐ 0 1 2 3 4 5 6 7
☐ 0 2 4 6 8

2. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :

☐ `void pgcd(int x, int y);`
☐ `int pgcd(int x, int x);`
☐ `int pgcd(int x, y);`
☐ `int pgcd(int y, int x);`

3. 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)`

4. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

☐ 4
☐ 0
☐ 3

5. 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

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 :

☐ 8
☐ 4
☐ 16
☐ 0

7. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

☐ `char 'c';`
☐ `char c;`
☐ `int char;`
☐ `char "c";`

8. Vous utilisez une boucle `while` quand :

☐ vous avez déjà fait un `for` dans le même programme principal
☐ l'incrément de la variable de boucle n'est pas 1
☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
☐ vous n'avez pas déclaré de fonction

9. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :

☐ `double exposant(double x, int n);`
☐ `exposant(double x, int n, int r);`
☐ `void exposant(double x^n);`
☐ `int exposant(double n, int x);`

10. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

☐ dans lequel ces fonctions sont appelées dans le main
☐ un ordre quelconque
☐ alphabétique
☐ dans lequel vous avez déclaré ces fonction

11. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :

☐ `int afficher_menu();`
☐ `int afficher_menu(int char);`
☐ `void afficher_menu();`
☐ `char afficher_menu(printf("menu"));`
☐ `double afficher_menu();`

12. Un enregistrement permet de grouper plusieurs valeurs dans :

☐ ses champs
☐ ses blocs
☐ ses chants
☐ ses cases

13. Le code suivant :

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

affichera :

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

14. Si cet avertissement apparaît à la compilation :
`warning: implicit declaration of function ‘max’`
, que doit-on chercher dans le programme ?
    - ☐ une fonction déclarée mais non définie
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction appelée avant sa déclaration
    - ☐ un désaccord entre la déclaration et la définition d'une fonction

15. 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

16. 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)

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

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

19. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :
    - ☐ `int pgcd(2);`
    - ☐ `n = pgcd(int p, int q);`
    - ☐ `int n = pgcd();`
    - ☐ `n = pgcd(n, 3);`

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

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 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
   - ☐ un fichier texte qui sera traduit en instructions processeur
2. 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`
3. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
   - ☐ `char c;`
   - ☐ `char 'c';`
   - ☐ `char "c";`
   - ☐ `int char;`
4. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function 'max'` , que doit-on chercher dans le programme ?
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une fonction déclarée mais non définie
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
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 = %d`
   - ☐ `j = 5`
6. Si cette erreur apparaît à la compilation : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une fonction déclarée mais non définie
   - ☐ une fonction appelée avant sa déclaration
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
7. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
   - ☐ `x = mccarthy(n);`
   - ☐ `n = mccarthy(p, q);`
   - ☐ `int mccarthy(int 2);`
   - ☐ `n = mccarthy();`
8. 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
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=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)`
10. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
    - ☐ `void afficher_menu();`
    - ☐ `double afficher_menu();`
    - ☐ `int afficher_menu();`
    - ☐ `char afficher_menu(printf("menu"));`
    - ☐ `int afficher_menu(int char);`
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 3 2 1 0
    - ☐ 0 1 2 3 4
    - ☐ 4 3 2 1
12. 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

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

14. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 0
- ☐ 7
- ☐ 5

15. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=8, b=5`

16. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il ne compile pas
- ☐ il n'affiche rien
- ☐ il comporte une boucle infinie

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
- ☐ 16
- ☐ 8
- ☐ 4

18. Un enregistrement permet de grouper plusieurs valeurs dans :

- ☐ ses champs
- ☐ ses chants
- ☐ ses cases
- ☐ ses blocs

19. Le bus système sert à :

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

20. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 4
- ☐ 5
- ☐ 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 déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
   - ☐ `loop i;`
   - ☐ `int loop n;`
   - ☐ `int %d;`
   - ☐ `int k;`
2. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il n'affiche rien
   - ☐ il comporte une boucle infinie
   - ☐ il risque d'afficher bonjour à la place de coucou
   - ☐ il ne compile pas
3. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
   - ☐ `double afficher_menu();`
   - ☐ `int afficher_menu(int char);`
   - ☐ `int afficher_menu();`
   - ☐ `char afficher_menu(printf("menu"));`
   - ☐ `void afficher_menu();`
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 = 0`
   - ☐ `j = 5`
   - ☐ `j = %d`
   - ☐ `j = 4`
5. L'écriture 101 en binaire correspond au nombre naturel :
   - ☐ 101
   - ☐ 5
   - ☐ 3
   - ☐ 4
6. Si cette erreur apparaît à la compilation :
   `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une fonction déclarée mais non définie
7. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses chants
   - ☐ ses cases
   - ☐ ses champs
   - ☐ ses blocs
8. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

   Alors l'expression `f(1)` prendra la valeur :
   - ☐ 0
   - ☐ 5
   - ☐ 1
   - ☐ 4
9. 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
10. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

    Alors l'expression `f(0)` prendra la valeur :
    - ☐ 0
    - ☐ 4
    - ☐ 3
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   }
```

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

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 `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

- ☐ `n = factorielle();`
- ☐ `printf("%d", factorielle(n));`
- ☐ `n = factorielle(p, q);`
- ☐ `int factorielle(int 2);`

14. 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

15. Vous utilisez une boucle `while` quand :

- ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
- ☐ vous n'avez pas déclaré de fonction
- ☐ vous avez déjà fait un `for` dans le même programme principal
- ☐ l'incrément de la variable de boucle n'est pas 1

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. 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

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

19. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :

- ☐ 1
- ☐ 0
- ☐ 0.5
- ☐ 1.5

20. 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

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

2. 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

3. Le type des réels en C est :

- ☐ `double`
- ☐ `char`
- ☐ `int`
- ☐ `real`

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

5. Vous utilisez une boucle `while` quand :

- ☐ vous avez déjà fait un `for` dans le même programme principal
- ☐ vous n'avez pas déclaré de fonction
- ☐ l'incrément de la variable de boucle n'est pas 1
- ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance

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

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

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

8. Un enregistrement permet de grouper plusieurs valeurs dans :

- ☐ ses cases
- ☐ ses chants
- ☐ ses blocs
- ☐ ses champs

9. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `a = b`
- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a{n, x} == b{n, x}`
- ☐ `a == b`

10. 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`

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 1 2 3 4
- ☐ 0 1 2 3
- ☐ 4 3 2 1 0

12. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

- ☐ dans lequel vous avez déclaré ces fonction
- ☐ alphabétique
- ☐ dans lequel ces fonctions sont appelées dans le main
- ☐ un ordre quelconque

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
- ☐ 4
- ☐ 0
- ☐ 16

14. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 0
- ☐ 4
- ☐ 3

15. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une fonction déclarée mais non définie
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une directive préprocesseur `#include` manquante
16. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
    - ☐ `int factorielle();`
    - ☐ `int factorielle(int x);`
    - ☐ `int factorielle(double n);`
    - ☐ `struct int factorielle(int n);`
17. Si cette erreur apparaît à la compilation : `erreur: conflicting types for ‘max’` , que doit-on chercher dans le programme ?
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction déclarée mais non définie
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
18. Un programme en langage C doit comporter une et une seule définition de la fonction :
    - ☐ main
    - ☐ begin
    - ☐ init
    - ☐ include
19. Le code suivant :

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

affichera :

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

20. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :

- ☐ `struct toto_s toto;`
- ☐ `toto_s n, x;`
- ☐ `int toto.n = 3;`
- ☐ `int struct toto_s = {3, -1e10};`
- ☐ `toto_s struct z = {3, 0.5};`

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. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
   - ☐ `void afficher_date(date_s d);`
   - ☐ `void afficher_date(struct date_s d);`
   - ☐ `struct date_s afficher_date(struct date_s d);`
   - ☐ `int afficher_date(date_s d);`
2. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

   Alors l'affichage sera :
   - ☐ `cccccc`
   - ☐ `ABCDEF`
   - ☐ `i`
   - ☐ `A`
3. 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 = 5`
   - ☐ `j = 0`
   - ☐ `j = 4`
4. Vous utilisez une boucle `while` quand :
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ vous n'avez pas déclaré de fonction
   - ☐ vous avez déjà fait un `for` dans le même programme principal
5. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function 'max'` , que doit-on chercher dans le programme ?
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une fonction déclarée mais non définie
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une directive préprocesseur `#include` manquante
6. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il risque d'afficher bonjour à la place de coucou
   - ☐ il ne compile pas
   - ☐ il comporte une boucle infinie
   - ☐ il n'affiche rien
7. 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
8. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   - ☐ `f(3,5)=8, a=3, b=5`
   - ☐ `f(a,b)=13, a=8, b=5`
   - ☐ `f(a,b)=8, a=3, b=5`
   - ☐ `f(a,b)=8, a=8, b=5`
9. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
   - ☐ `x = racine(2/3);`
   - ☐ `x = racine(racine(x)*racine(x));`
   - ☐ `x - 1 = racine(x);`
   - ☐ `x = racine(x * x) - racine(x);`
10. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
    - ☐ `int n = carre();`
    - ☐ `n = carre(n);`
    - ☐ `n = carre(int n);`
    - ☐ `int carre(2);`

11. 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`

12. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :

- ☐ `void afficher_menu();`
- ☐ `int afficher_menu();`
- ☐ `double afficher_menu();`
- ☐ `int afficher_menu(int char);`
- ☐ `char afficher_menu(printf("menu"));`

13. 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 bus explose
- ☐ le terminal affiche 8

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

15. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

- ☐ `int char;`
- ☐ `char c;`
- ☐ `char 'c';`
- ☐ `char "c";`

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

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

17. Une variable booléenne est un variable :

- ☐ NaN (not a number, qui n'est pas un nombre)
- ☐ réelle positive
- ☐ à laquelle une valeur vient d'être affectée
- ☐ qui est vraie ou fausse
- ☐ jamais nulle

18. 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

19. L'écriture 101 en binaire correspond au nombre naturel :

- ☐ 4
- ☐ 5
- ☐ 3
- ☐ 101

20. 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

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

2. L'écriture 101 en binaire correspond au nombre naturel :
   - ☐ 4
   - ☐ 101
   - ☐ 3
   - ☐ 5

3. 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

4. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `a<=n<=b`
- ☐ `(a<=n) && (n<=b)`
- ☐ `(n<=a) && (n<=b)`
- ☐ `(a<n) || (n>b)`

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
- ☐ 0
- ☐ 6

6. Le code suivant :

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

affichera :

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

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

8. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
   - ☐ un débogueur
   - ☐ `scanf("%d", &n);`
   - ☐ `printf("Valeur de n ? %g\n", n);`
   - ☐ `printf("Valeur de n ? %d\n", n);`

9. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que `x` est une variable réelle définie et initialisée, il est incorrect d'écrire :
   - ☐ `x = racine(2/3);`
   - ☐ `x = racine(x * x) - racine(x);`
   - ☐ `x = racine(racine(x)*racine(x));`
   - ☐ `x - 1 = racine(x);`

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  }
```

- ☐ 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

11. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 1
- ☐ 5
- ☐ 0
- ☐ 4

12. Un enregistrement permet de grouper plusieurs valeurs dans :

- ☐ ses blocs
- ☐ ses chants
- ☐ ses cases
- ☐ ses champs

13. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :

- ☐ `int factorielle(double n);`
- ☐ `int factorielle();`
- ☐ `int factorielle(int x);`
- ☐ `struct int factorielle(int n);`

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. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il n'affiche rien
- ☐ il ne compile pas
- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il comporte une boucle infinie

16. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :

- ☐ `n = carre(int n);`
- ☐ `int carre(2);`
- ☐ `n = carre(n);`
- ☐ `int n = carre();`

17. Vous utilisez une boucle `while` quand :

- ☐ vous n'avez pas déclaré de fonction
- ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
- ☐ l'incrément de la variable de boucle n'est pas 1
- ☐ vous avez déjà fait un `for` dans le même programme principal

18. Le type des réels en C est :

- ☐ `double`
- ☐ `real`
- ☐ `char`
- ☐ `int`

19. 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
- ☐ une accolade manquante
- ☐ un point-virgule en trop

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

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 le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(3,5)=8, a=3, b=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 0 1 2 3`
- ☐ `0 1 2 0 1 2`
- ☐ `0 0 1 1 2 2 3`
- ☐ `0 1 2 3 0 1 2`

3. 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

4. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses blocs
   - ☐ ses chants
   - ☐ ses cases
   - ☐ ses champs

5. 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

6. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
   - ☐ `int mccarthy(int 2);`
   - ☐ `n = mccarthy(p, q);`
   - ☐ `x = mccarthy(n);`
   - ☐ `n = mccarthy();`

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

8. 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

9. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
   - ☐ un débogueur
   - ☐ `printf("Valeur de n ? %g\n", n);`
   - ☐ `scanf("%d", &n);`
   - ☐ `printf("Valeur de n ? %d\n", n);`

10. 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

11. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ dans lequel ces fonctions sont appelées dans le main
    - ☐ alphabétique
    - ☐ un ordre quelconque
    - ☐ dans lequel vous avez déclaré ces fonction

12. Le code suivant :

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

affichera :

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

13. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 3
- ☐ 0
- ☐ 4

14. 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

15. 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`

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 3 2 1 0
- ☐ 0 1 2 3 4
- ☐ 4 3 2 1

17. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

Alors l'affichage sera :

- ☐ `b`
- ☐ `B`
- ☐ `C`
- ☐ `A`

18. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `(a<n) || (n>b)`
- ☐ `(a<=n) && (n<=b)`
- ☐ `(n<=a) && (n<=b)`
- ☐ `a<=n<=b`

19. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

- ☐ `n = factorielle();`
- ☐ `printf("%d", factorielle(n));`
- ☐ `int factorielle(int 2);`
- ☐ `n = factorielle(p, q);`

20. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :

- ☐ 1
- ☐ 0.5
- ☐ 0
- ☐ 1.5

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. 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
2. 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 -Wall prog.exe -o prog.c`
   - ☐ `gcc prog.exe -Wall -o prog.c`
3. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il comporte une boucle infinie
   - ☐ il n'affiche rien
   - ☐ il ne compile pas
   - ☐ il risque d'afficher bonjour à la place de coucou
4. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   - ☐ `f(a,b)=8, a=8, b=5`
   - ☐ `f(a,b)=13, a=8, b=5`
   - ☐ `f(3,5)=8, a=3, b=5`
   - ☐ `f(a,b)=8, a=3, b=5`
5. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
   - ☐ un débogueur
   - ☐ `printf("Valeur de n ? %g\n", n);`
   - ☐ `printf("Valeur de n ? %d\n", n);`
   - ☐ `scanf("%d", &n);`
6. Sur unix (ou linux), la commande `mkdir` permet de :
   - ☐ changer de répertoire courant
   - ☐ créer un fichier texte
   - ☐ ouvrir un fichier texte
   - ☐ créer un répertoire
7. 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
8. Le langage C est un langage
   - ☐ lu, écrit, parlé
   - ☐ composé
   - ☐ interprété
   - ☐ compilé
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`
   - ☐ `0 1 0 1 0 1 0 1`
   - ☐ `0 0 0 1 1 1`
   - ☐ `1 2 1 2 3`
10. Le code suivant :

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

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

11. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `(a<=n) && (n<=b)`
- ☐ `a<=n<=b`
- ☐ `(a<n) || (n>b)`
- ☐ `(n<=a) && (n<=b)`

12. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

- ☐ `char "c";`
- ☐ `int char;`
- ☐ `char 'c';`
- ☐ `char c;`

13. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :

- ☐ `int pgcd(2);`
- ☐ `n = pgcd(int p, int q);`
- ☐ `n = pgcd(n, 3);`
- ☐ `int n = pgcd();`

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. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?

- ☐ `!(!A || B) == (A && !B)`
- ☐ `(!A || B)`
- ☐ `(A == TRUE) && (B == TRUE)`
- ☐ `A && B`

16. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :

- ☐ `struct toto_s toto;`
- ☐ `int toto.n = 3;`
- ☐ `toto_s n, x;`
- ☐ `toto_s struct z = {3, 0.5};`
- ☐ `int struct toto_s = {3, -1e10};`

17. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :

- ☐ `int saisie_utilisateur();`
- ☐ `saisie_utilisateur(scanf(%d));`
- ☐ `void saisie_utilisateur(int n);`
- ☐ `void saisie_utilisateur(char c);`

18. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 7
- ☐ 0
- ☐ 5

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 :

- ☐ 0
- ☐ 4
- ☐ 8
- ☐ 16

20. 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

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. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

☐ `ABCDEF`

☐ `cccccc`

☐ `i`

☐ `A`

2. Le code suivant :

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

affichera :

☐ `Mineur`

☐ rien

☐ `Majeur`

☐ `Mineur`
`Majeur`

3. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :

☐ 3

☐ 111

☐ 8

☐ 7

4. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

☐ 4

☐ 0

☐ 3

5. 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

6. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?

☐ `(A == TRUE) && (B == TRUE)`

☐ `!(!A || B) == (A && !B)`

☐ `A && B`

☐ `(!A || B)`

7. 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

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. 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. Vous utilisez une boucle `while` quand :

☐ l'incrément de la variable de boucle n'est pas 1

☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance

☐ vous avez déjà fait un `for` dans le même programme principal

☐ vous n'avez pas déclaré de fonction

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=%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);`

12. 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`
13. Quel est l'opérateur de différence en C :
    - ☐ `!`
    - ☐ `<>`
    - ☐ $\neq$
    - ☐ `!=`
14. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
    - ☐ `void saisie_utilisateur(char c);`
    - ☐ `saisie_utilisateur(scanf(%d));`
    - ☐ `void saisie_utilisateur(int n);`
    - ☐ `int saisie_utilisateur();`
15. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
    - ☐ 1
    - ☐ 0.5
    - ☐ 1.5
    - ☐ 0
16. Le langage C est un langage
    - ☐ lu, écrit, parlé
    - ☐ compilé
    - ☐ interprété
    - ☐ composé
17. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ dans lequel ces fonctions sont appelées dans le main
    - ☐ un ordre quelconque
    - ☐ alphabétique
    - ☐ dans lequel vous avez déclaré ces fonction
18. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :
    - ☐ `n = pgcd(n, 3);`
    - ☐ `n = pgcd(int p, int q);`
    - ☐ `int pgcd(2);`
    - ☐ `int n = pgcd();`
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  }
```

    - ☐ le programme affiche `****`
    - ☐ la variable x vaut $-\frac{1}{2}$
    - ☐ le programme affiche x
    - ☐ la variable x vaut 16
20. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

    Quelle est la condition `cond` :
    - ☐ `(a<=n) && (n<=b)`
    - ☐ `a<=n<=b`
    - ☐ `(a<n) || (n>b)`
    - ☐ `(n<=a) && (n<=b)`

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 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
2. Le type des réels en C est :
   - ☐ `double`
   - ☐ `char`
   - ☐ `int`
   - ☐ `real`
3. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
   - ☐ `void afficher_date(date_s d);`
   - ☐ `struct date_s afficher_date(struct date_s d);`
   - ☐ `void afficher_date(struct date_s d);`
   - ☐ `int afficher_date(date_s d);`
4. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse lexicale
   - ☐ analyse syntaxique
   - ☐ analyse sémantique
   - ☐ analyse harmonique
5. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :

- ☐ `toto_s n, x;`
- ☐ `struct toto_s toto;`
- ☐ `toto_s struct z = {3, 0.5};`
- ☐ `int toto.n = 3;`
- ☐ `int struct toto_s = {3, -1e10};`

6. 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>`
7. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
   - ☐ 3
   - ☐ 7
   - ☐ 111
   - ☐ 8
8. 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
9. 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
10. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `(n<=a) && (n<=b)`
- ☐ `(a<n) || (n>b)`
- ☐ `a<=n<=b`
- ☐ `(a<=n) && (n<=b)`

11. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ alphabétique
    - ☐ un ordre quelconque
    - ☐ dans lequel vous avez déclaré ces fonction
    - ☐ dans lequel ces fonctions sont appelées dans le main
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
- ☐ 0 1 2 3
- ☐ 4 3 2 1
- ☐ 4 3 2 1 0

13. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 3
- ☐ 0
- ☐ 4

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 = 0`
- ☐ `j = %d`

15. 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
16. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
    - ☐ `char c;`
    - ☐ `int char;`
    - ☐ `char "c";`
    - ☐ `char 'c';`
17. Une variable booléenne est un variable :
    - ☐ NaN (not a number, qui n'est pas un nombre)
    - ☐ réelle positive
    - ☐ à laquelle une valeur vient d'être affectée
    - ☐ jamais nulle
    - ☐ qui est vraie ou fausse
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. Sous unix (ou linux), la commande `cd` permet de :
    - ☐ changer de répertoire courant
    - ☐ ouvir un bureau partagé (common desktop)
    - ☐ récupérer un programme arrêté avec la commande `ab`
    - ☐ jouer de la musique
    - ☐ détruire un fichier
20. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `a{n, x} == b{n, x}`
- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a = b`
- ☐ `a == b`

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 enregistrement permet de grouper plusieurs valeurs dans :

   ☐ ses cases

   ☐ ses blocs

   ☐ ses champs

   ☐ ses chants

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 16

   ☐ la case mémoire 8 contiendra 0

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 y vaut 5

   ☐ le programme affiche "Faux"

   ☐ la variable x vaut 5 et la variable y vaut 0

   ☐ la variable x vaut 0

4. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

   Alors l'expression `f(1)` prendra la valeur :

   ☐ 5

   ☐ 1

   ☐ 4

   ☐ 0

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

   ☐ 8

   ☐ 4

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 1 2 0 1 2 3`

   ☐ `0 1 2 3 0 1 2`

   ☐ `0 0 1 1 2 2 3`

7. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

   ☐ `int char;`

   ☐ `char 'c';`

   ☐ `char c;`

   ☐ `char "c";`

8. 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`

9. 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 ?

   ☐ `**** **** **** ****`

   ☐ `** *** **** *****`

   ☐ `***** **** *** ***`

   ☐ `** ** ** ** ** **`

10. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ dans lequel ces fonctions sont appelées dans le main
    - ☐ alphabétique
    - ☐ dans lequel vous avez déclaré ces fonction
    - ☐ un ordre quelconque
11. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

    Alors l'expression `g(0)` prendra la valeur :
    - ☐ 7
    - ☐ 5
    - ☐ 0
12. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

    appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

    L'affichage dans le main est le suivant :
    - ☐ `f(a,b)=13, a=8, b=5`
    - ☐ `f(3,5)=8, a=3, b=5`
    - ☐ `f(a,b)=8, a=3, b=5`
    - ☐ `f(a,b)=8, a=8, b=5`
13. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

    Alors pour tester l'égalité de a et de b on utilise la condition :
    - ☐ `a{n, x} == b{n, x}`
    - ☐ `a == b`
    - ☐ `a = b`
    - ☐ `(a.n == b.n) && (a.x == b.x)`
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 sémantique
    - ☐ l'analyse des entrées clavier
    - ☐ l'édition de liens
    - ☐ l'analyse harmonique
15. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
    - ☐ `int saisie_utilisateur();`
    - ☐ `void saisie_utilisateur(int n);`
    - ☐ `void saisie_utilisateur(char c);`
    - ☐ `saisie_utilisateur(scanf(%d));`
16. Un programme en langage C doit comporter une et une seule définition de la fonction :
    - ☐ include
    - ☐ main
    - ☐ init
    - ☐ begin
17. L'écriture $\underline{101}$ en binaire correspond au nombre naturel :
    - ☐ 3
    - ☐ 5
    - ☐ 101
    - ☐ 4
18. 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
19. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

    Alors l'expression `f(0)` prendra la valeur :
    - ☐ 3
    - ☐ 0
    - ☐ 4
20. 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`

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 la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

☐ 0
☐ 3
☐ 4

2. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

☐ il n'affiche rien
☐ il comporte une boucle infinie
☐ il risque d'afficher bonjour à la place de coucou
☐ il ne compile pas

3. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :

☐ `int n = carre();`
☐ `n = carre(n);`
☐ `n = carre(int n);`
☐ `int carre(2);`

4. 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`

5. Le langage C est un langage

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

6. 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

7. Si cette erreur apparaît à la compilation :
`erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?

☐ une fonction déclarée mais non définie
☐ une fonction appelée avant sa déclaration
☐ une directive préprocesseur `#include` manquante
☐ un désaccord entre la déclaration et la définition d'une fonction

8. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?

☐ `(!A || B)`
☐ `(A == TRUE) && (B == TRUE)`
☐ `A && B`
☐ `!(!A || B) == (A && !B)`

9. Vous utilisez une boucle `while` quand :

☐ l'incrément de la variable de boucle n'est pas 1
☐ vous n'avez pas déclaré de fonction
☐ vous avez déjà fait un `for` dans le même programme principal
☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance

10. 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;`

11. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

☐ `a == b`
☐ `(a.n == b.n) && (a.x == b.x)`
☐ `a{n, x} == b{n, x}`
☐ `a = b`

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. L'écriture $\underline{101}$ en binaire correspond au nombre naturel :

☐ 3
☐ 4
☐ 101
☐ 5

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

15. 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
16. Avant de faire appel à une fonction il est nécessaire de :
    - ☐ l'avoir déclarée
    - ☐ l'avoir déclarée et définie
    - ☐ l'avoir définie
    - ☐ avoir défini une constante symbolique de la taille de cette fonction
17. 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
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
- ☐ 4 3 2 1
- ☐ 1 2 3 4
- ☐ 4 3 2 1 0

19. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ alphabétique
    - ☐ dans lequel vous avez déclaré ces fonction
    - ☐ dans lequel ces fonctions sont appelées dans le main
    - ☐ un ordre quelconque
20. 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>`

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 ?
   - ☐ 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

2. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
   - ☐ `A && B`
   - ☐ `!(!A || B) == (A && !B)`
   - ☐ `(!A || B)`
   - ☐ `(A == TRUE) && (B == TRUE)`

3. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
   - ☐ 1.5
   - ☐ 0.5
   - ☐ 1
   - ☐ 0

4. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
   - ☐ `char 'c';`
   - ☐ `int char;`
   - ☐ `char c;`
   - ☐ `char "c";`

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
   - ☐ 6
   - ☐ 15
   - ☐ 10

6. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
   - ☐ `void saisie_utilisateur(int n);`
   - ☐ `saisie_utilisateur(scanf(%d));`
   - ☐ `void saisie_utilisateur(char c);`
   - ☐ `int saisie_utilisateur();`

7. 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`

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

9. Le code suivant :

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

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

10. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

    appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

    L'affichage dans le main est le suivant :
    - ☐ `f(a,b)=8, a=8, b=5`
    - ☐ `f(a,b)=13, a=8, b=5`
    - ☐ `f(3,5)=8, a=3, b=5`
    - ☐ `f(a,b)=8, a=3, b=5`

11. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses cases
    - ☐ ses champs
    - ☐ ses chants
    - ☐ ses blocs

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

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

14. L'écriture 101 en binaire correspond au nombre naturel :
    - ☐ 5
    - ☐ 101
    - ☐ 4
    - ☐ 3

15. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :
    - ☐ `int toto.n = 3;`
    - ☐ `struct toto_s toto;`
    - ☐ `toto_s n, x;`
    - ☐ `int struct toto_s = {3, -1e10};`
    - ☐ `toto_s struct z = {3, 0.5};`

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

17. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
    - ☐ `printf("Valeur de n ? %g\n", n);`
    - ☐ un débogueur
    - ☐ `scanf("%d", &n);`
    - ☐ `printf("Valeur de n ? %d\n", n);`

18. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

    - ☐ il comporte une boucle infinie
    - ☐ il ne compile pas
    - ☐ il n'affiche rien
    - ☐ il risque d'afficher bonjour à la place de coucou

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

20. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :
    - ☐ `n = pgcd(n, 3);`
    - ☐ `n = pgcd(int p, int q);`
    - ☐ `int pgcd(2);`
    - ☐ `int n = pgcd();`

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 des entrées clavier
   - ☐ l'édition de liens
   - ☐ l'analyse harmonique
2. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   - ☐ 5
   - ☐ 0
   - ☐ 7
3. Le code suivant :

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

   affichera :
   - ☐ `Majeur`
   - ☐ `Mineur`
   - ☐ rien
   - ☐ `Mineur`
     `Majeur`
4. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
   - ☐ 1.5
   - ☐ 0.5
   - ☐ 1
   - ☐ 0
5. 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
   - ☐ une gamme de fréquence de fonctionnement du processeur
   - ☐ un composant qui contient la liste des fichiers du système
6. L'écriture 101 en binaire correspond au nombre naturel :
   - ☐ 5
   - ☐ 3
   - ☐ 101
   - ☐ 4
7. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   - ☐ `f(a,b)=8, a=3, b=5`
   - ☐ `f(a,b)=13, a=8, b=5`
   - ☐ `f(3,5)=8, a=3, b=5`
   - ☐ `f(a,b)=8, a=8, b=5`
8. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

   précède la fonction `main()`, alors on peut écrire en début de `main()` :
   - ☐ `int toto.n = 3;`
   - ☐ `struct toto_s toto;`
   - ☐ `toto_s n, x;`
   - ☐ `int struct toto_s = {3, -1e10};`
   - ☐ `toto_s struct z = {3, 0.5};`
9. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

   Alors l'expression `f(1)` prendra la valeur :
   - ☐ 5
   - ☐ 4
   - ☐ 1
   - ☐ 0
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=0;i<=5;i=i+1)`
    - ☐ `for(i=1;i<5;i=i+1)`
    - ☐ `for(i=1;i<=5;i=i+1)`

11. Un programme en langage C doit comporter une et une seule définition de la fonction :
    - ☐ init
    - ☐ include
    - ☐ main
    - ☐ begin
12. Quel est l'opérateur de différence en C :
    - ☐ `<>`
    - ☐ `!`
    - ☐ $\neq$
    - ☐ `!=`
13. Le langage C est un langage
    - ☐ interprété
    - ☐ composé
    - ☐ compilé
    - ☐ lu, écrit, parlé
14. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
    - ☐ `double afficher_menu();`
    - ☐ `char afficher_menu(printf("menu"));`
    - ☐ `int afficher_menu();`
    - ☐ `void afficher_menu();`
    - ☐ `int afficher_menu(int char);`
15. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
    - ☐ `(A == TRUE) && (B == TRUE)`
    - ☐ `!(!A || B) == (A && !B)`
    - ☐ `(!A || B)`
    - ☐ `A && B`
16. Une variable booléenne est un variable :
    - ☐ qui est vraie ou fausse
    - ☐ à laquelle une valeur vient d'être affectée
    - ☐ NaN (not a number, qui n'est pas un nombre)
    - ☐ jamais nulle
    - ☐ réelle positive
17. 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
18. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
    - ☐ `int pgcd(int x, y);`
    - ☐ `int pgcd(int x, int x);`
    - ☐ `void pgcd(int x, int y);`
    - ☐ `int pgcd(int y, int x);`
19. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ un ordre quelconque
    - ☐ dans lequel ces fonctions sont appelées dans le main
    - ☐ alphabétique
    - ☐ dans lequel vous avez déclaré ces fonction
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
    - ☐ répéter un bloc tant qu'une condition est vérifée
    - ☐ retourner un bloc
    - ☐ 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. Sur unix (ou linux), la commande `mkdir` permet de :
   - ☐ créer un fichier texte
   - ☐ changer de répertoire courant
   - ☐ ouvrir un fichier texte
   - ☐ créer un répertoire
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
   - ☐ 4 3 2 1
   - ☐ 0 1 2 3 4
3. Le code suivant :

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

   affichera :
   - ☐ `Mineur`
   - ☐ `Majeur`
   - ☐ `Mineur`
     `Majeur`
   - ☐ rien
4. Si cet avertissement apparaît à la compilation :
   `warning: implicit declaration of function ‘max’`
   , que doit-on chercher dans le programme ?
   - ☐ une fonction appelée avant sa déclaration
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une fonction déclarée mais non définie
   - ☐ une directive préprocesseur `#include` manquante
5. 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
   - ☐ 6
   - ☐ 42
6. Le type des réels en C est :
   - ☐ `real`
   - ☐ `int`
   - ☐ `char`
   - ☐ `double`
7. Si cette erreur apparaît à la compilation :
   `erreur: conflicting types for ‘max’` , que doit-on chercher dans le programme ?
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une fonction déclarée mais non définie
8. 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
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. 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
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 0
    - ☐ 4 3 2 1

12. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :

- ☐ `x = racine(x * x) - racine(x);`
- ☐ `x - 1 = racine(x);`
- ☐ `x = racine(racine(x)*racine(x));`
- ☐ `x = racine(2/3);`

13. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

- ☐ `A`
- ☐ `ABCDEF`
- ☐ `cccccc`
- ☐ `i`

14. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

- ☐ dans lequel vous avez déclaré ces fonction
- ☐ dans lequel ces fonctions sont appelées dans le main
- ☐ un ordre quelconque
- ☐ alphabétique

15. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :

- ☐ `printf("Valeur de n ? %d\n", n);`
- ☐ `scanf("%d", &n);`
- ☐ `printf("Valeur de n ? %g\n", n);`
- ☐ un débogueur

16. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :

- ☐ `void afficher_date(struct date_s d);`
- ☐ `int afficher_date(date_s d);`
- ☐ `void afficher_date(date_s d);`
- ☐ `struct date_s afficher_date(struct date_s d);`

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

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

18. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 5
- ☐ 0
- ☐ 1
- ☐ 4

19. Un enregistrement permet de grouper plusieurs valeurs dans :

- ☐ ses cases
- ☐ ses chants
- ☐ ses champs
- ☐ ses blocs

20. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

- ☐ `int char;`
- ☐ `char 'c';`
- ☐ `char c;`
- ☐ `char "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. 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

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 lancer un déboggueur
☐ qu'il faut indenter le fichier source

3. Si cette erreur apparaît à la compilation : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?

☐ une fonction déclarée mais non définie
☐ une directive préprocesseur `#include` manquante
☐ une fonction appelée avant sa déclaration
☐ un désaccord entre la déclaration et la définition d'une fonction

4. 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
☐ une accolade manquante
☐ un point-virgule en trop

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

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

6. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

Alors l'affichage sera :

☐ `b`
☐ `C`
☐ `A`
☐ `B`

7. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :

☐ `int struct toto_s = {3, -1e10};`
☐ `int toto.n = 3;`
☐ `toto_s n, x;`
☐ `struct toto_s toto;`
☐ `toto_s struct z = {3, 0.5};`

8. 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

9. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

☐ `char "c";`
☐ `int char;`
☐ `char c;`
☐ `char 'c';`

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. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

☐ `a<=n<=b`
☐ `(a<=n) && (n<=b)`
☐ `(n<=a) && (n<=b)`
☐ `(a<n) || (n>b)`

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

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

- ☐ `!=`
- ☐ `!`
- ☐ `<>`
- ☐ $\neq$

14. Une variable booléenne est un variable :

- ☐ NaN (not a number, qui n'est pas un nombre)
- ☐ à laquelle une valeur vient d'être affectée
- ☐ jamais nulle
- ☐ réelle positive
- ☐ qui est vraie ou fausse

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

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

16. Un enregistrement permet de grouper plusieurs valeurs dans :

- ☐ ses chants
- ☐ ses blocs
- ☐ ses champs
- ☐ ses cases

17. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `a = b`
- ☐ `a{n, x} == b{n, x}`
- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a == b`

18. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :

- ☐ `n = mccarthy();`
- ☐ `n = mccarthy(p, q);`
- ☐ `int mccarthy(int 2);`
- ☐ `x = mccarthy(n);`

19. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 4
- ☐ 0
- ☐ 3

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

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 une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
   - ☐ `int afficher_menu(int char);`
   - ☐ `double afficher_menu();`
   - ☐ `int afficher_menu();`
   - ☐ `void afficher_menu();`
   - ☐ `char afficher_menu(printf("menu"));`
2. 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)
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=%x et y=%y\n");`
   - ☐ `printf("x=%d et y=%d\n",x,y);`
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 = 0`
   - ☐ `j = %d`
   - ☐ `j = 5`
   - ☐ `j = 4`
5. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
   - ☐ `x = racine(racine(x)*racine(x));`
   - ☐ `x = racine(x * x) - racine(x);`
   - ☐ `x = racine(2/3);`
   - ☐ `x - 1 = racine(x);`
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 :
   - ☐ 16
   - ☐ 6
   - ☐ 20
   - ☐ 3
7. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   - ☐ `f(a,b)=8, a=3, b=5`
   - ☐ `f(a,b)=8, a=8, b=5`
   - ☐ `f(3,5)=8, a=3, b=5`
   - ☐ `f(a,b)=13, a=8, b=5`
8. L'écriture $\underline{101}$ en binaire correspond au nombre naturel :
   - ☐ 4
   - ☐ 101
   - ☐ 5
   - ☐ 3
9. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
   - ☐ `n = factorielle();`
   - ☐ `int factorielle(int 2);`
   - ☐ `printf("%d", factorielle(n));`
   - ☐ `n = factorielle(p, q);`
10. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :
    - ☐ `int pgcd(2);`
    - ☐ `n = pgcd(n, 3);`
    - ☐ `int n = pgcd();`
    - ☐ `n = pgcd(int p, int q);`
11. Sous unix (ou linux), la commande `cd` permet de :
    - ☐ jouer de la musique
    - ☐ changer de répertoire courant
    - ☐ récupérer un programme arrêté avec la commande `ab`
    - ☐ ouvir un bureau partagé (common desktop)
    - ☐ détruire un fichier
12. Le code suivant :

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

    affichera :

☐ `Majeur`

☐ rien

☐ `Mineur`
`Majeur`

☐ `Mineur`

13. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

☐ `char 'c';`

☐ `char c;`

☐ `int char;`

☐ `char "c";`

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 sémantique

☐ l'analyse des entrées clavier

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 terminal affiche 8

☐ la case mémoire 8 contiendra 16

☐ la case mémoire 8 contiendra 0

☐ le bus explose

16. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

☐ 7

☐ 5

☐ 0

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 `****`

☐ la variable x vaut 16

☐ la variable x vaut $-\frac{1}{2}$

☐ le programme affiche x

18. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

☐ il n'affiche rien

☐ il risque d'afficher bonjour à la place de coucou

☐ il comporte une boucle infinie

☐ il ne compile pas

19. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

☐ 0

☐ 3

☐ 4

20. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?

☐ `(A == TRUE) && (B == TRUE)`

☐ `A && B`

☐ `!(!A || B) == (A && !B)`

☐ `(!A || B)`

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 y vaut 5
   - ☐ la variable x vaut 5 et la variable y vaut 3
   - ☐ la variable x vaut 3

2. 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)

3. Le type des réels en C est :
   - ☐ `int`
   - ☐ `real`
   - ☐ `double`
   - ☐ `char`

4. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ dans lequel vous avez déclaré ces fonction
   - ☐ un ordre quelconque
   - ☐ alphabétique
   - ☐ dans lequel ces fonctions sont appelées dans le main

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

6. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
   - ☐ `struct int factorielle(int n);`
   - ☐ `int factorielle(double n);`
   - ☐ `int factorielle();`
   - ☐ `int factorielle(int x);`

7. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
   - ☐ 1.5
   - ☐ 0.5
   - ☐ 1
   - ☐ 0

8. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses cases
   - ☐ ses chants
   - ☐ ses blocs
   - ☐ ses champs

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

10. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

    Quelle est la condition `cond` :
    - ☐ `(a<n) || (n>b)`
    - ☐ `(a<=n) && (n<=b)`
    - ☐ `a<=n<=b`
    - ☐ `(n<=a) && (n<=b)`

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

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 :

- ☐ 20
- ☐ 16
- ☐ 3
- ☐ 6

14. Si cette erreur apparaît à la compilation :
`erreur: conflicting types for ‘max’` , que doit-on chercher dans le programme ?
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une fonction déclarée mais non définie

15. Le code suivant :

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

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

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

17. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :
    - ☐ `a == b`
    - ☐ `a = b`
    - ☐ `(a.n == b.n) && (a.x == b.x)`
    - ☐ `a{n, x} == b{n, x}`

18. 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

19. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :
    - ☐ 0
    - ☐ 7
    - ☐ 5

20. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :
    - ☐ `toto_s struct z = {3, 0.5};`
    - ☐ `toto_s n, x;`
    - ☐ `int struct toto_s = {3, -1e10};`
    - ☐ `int toto.n = 3;`
    - ☐ `struct toto_s toto;`

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 enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses chants
   - ☐ ses champs
   - ☐ ses blocs
   - ☐ ses cases

2. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ dans lequel ces fonctions sont appelées dans le main
   - ☐ un ordre quelconque
   - ☐ alphabétique
   - ☐ dans lequel vous avez déclaré ces fonction

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

4. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

   Alors l'affichage sera :
   - ☐ `B`
   - ☐ `b`
   - ☐ `C`
   - ☐ `A`

5. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

   Alors l'affichage sera :
   - ☐ `i`
   - ☐ `cccccc`
   - ☐ `A`
   - ☐ `ABCDEF`

6. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

   Quelle est la condition `cond` :
   - ☐ `a<=n<=b`
   - ☐ `(a<n) || (n>b)`
   - ☐ `(a<=n) && (n<=b)`
   - ☐ `(n<=a) && (n<=b)`

7. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
   - ☐ `void afficher_date(date_s d);`
   - ☐ `void afficher_date(struct date_s d);`
   - ☐ `int afficher_date(date_s d);`
   - ☐ `struct date_s afficher_date(struct date_s d);`

8. 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

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

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 x
   - ☐ la variable x vaut 16
   - ☐ le programme affiche `****`

11. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
   - ☐ `int afficher_menu();`
   - ☐ `int afficher_menu(int char);`
   - ☐ `void afficher_menu();`
   - ☐ `double afficher_menu();`
   - ☐ `char afficher_menu(printf("menu"));`

12. Un programme en langage C doit comporter une et une seule définition de la fonction :
    - ☐ include
    - ☐ main
    - ☐ init
    - ☐ begin
13. Avant de faire appel à une fonction il est nécessaire de :
    - ☐ l'avoir déclarée
    - ☐ avoir défini une constante symbolique de la taille de cette fonction
    - ☐ l'avoir déclarée et définie
    - ☐ l'avoir définie
14. Une variable booléenne est un variable :
    - ☐ à laquelle une valeur vient d'être affectée
    - ☐ réelle positive
    - ☐ jamais nulle
    - ☐ NaN (not a number, qui n'est pas un nombre)
    - ☐ qui est vraie ou fausse
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 = 5`
    - ☐ `j = %d`
    - ☐ `j = 4`
16. Vous utilisez une boucle `while` quand :
    - ☐ vous n'avez pas déclaré de fonction
    - ☐ l'incrément de la variable de boucle n'est pas 1
    - ☐ vous avez déjà fait un `for` dans le même programme principal
    - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
17. 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
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 x vaut 3
    - ☐ le programme affiche "Faux"
    - ☐ la variable y vaut 5
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
    - ☐ 0
    - ☐ 16
20. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
    - ☐ `void exposant(double x^n);`
    - ☐ `int exposant(double n, int x);`
    - ☐ `double exposant(double x, int n);`
    - ☐ `exposant(double x, int n, int r);`

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 a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `a == b`
- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a = b`
- ☐ `a{n, x} == b{n, x}`

2. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :

- ☐ `int factorielle(int x);`
- ☐ `int factorielle(double n);`
- ☐ `int factorielle();`
- ☐ `struct int factorielle(int n);`

3. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il ne compile pas
- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il comporte une boucle infinie
- ☐ il n'affiche rien

4. 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

5. 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
- ☐ 4 3 2 1 0

6. Si cette erreur apparaît à la compilation : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?

- ☐ une directive préprocesseur `#include` manquante
- ☐ un désaccord entre la déclaration et la définition d'une fonction
- ☐ une fonction appelée avant sa déclaration
- ☐ une fonction déclarée mais non définie

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

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

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

9. Le type des réels en C est :

- ☐ `char`
- ☐ `int`
- ☐ `real`
- ☐ `double`

10. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :

- ☐ `int toto.n = 3;`
- ☐ `toto_s struct z = {3, 0.5};`
- ☐ `struct toto_s toto;`
- ☐ `int struct toto_s = {3, -1e10};`
- ☐ `toto_s n, x;`

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 ?

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

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
- ☐ 8 6 4 2
- ☐ 0 2 4 6 8
- ☐ 8 2

13. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 1
- ☐ 4
- ☐ 0
- ☐ 5

14. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `a<=n<=b`
- ☐ `(n<=a) && (n<=b)`
- ☐ `(a<n) || (n>b)`
- ☐ `(a<=n) && (n<=b)`

15. Vous utilisez une boucle `while` quand :

- ☐ l'incrément de la variable de boucle n'est pas 1
- ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
- ☐ vous n'avez pas déclaré de fonction
- ☐ vous avez déjà fait un `for` dans le même programme principal

16. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

- ☐ `cccccc`
- ☐ `A`
- ☐ `ABCDEF`
- ☐ `i`

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. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 0
- ☐ 4
- ☐ 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=0;i<=5;i=i+1)`
- ☐ `for(i=1;i<=5;i=i+1)`

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  }
```

- ☐ 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

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 a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a == b`
- ☐ `a = b`
- ☐ `a{n, x} == b{n, x}`

2. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il ne compile pas
- ☐ il n'affiche rien
- ☐ il comporte une boucle infinie
- ☐ il risque d'afficher bonjour à la place de coucou

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 :

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

4. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :

- ☐ 0.5
- ☐ 1
- ☐ 1.5
- ☐ 0

5. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :

- ☐ `n = pgcd(int p, int q);`
- ☐ `n = pgcd(n, 3);`
- ☐ `int pgcd(2);`
- ☐ `int n = pgcd();`

6. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :

- ☐ `n = carre(int n);`
- ☐ `n = carre(n);`
- ☐ `int n = carre();`
- ☐ `int carre(2);`

7. 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`

8. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 3
- ☐ 0
- ☐ 4

9. Le langage C est un langage

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

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 0
- ☐ 8 6 4 2
- ☐ 0 2 4 6 8

11. Le code suivant :

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

affichera :

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

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

13. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?

- ☐ `A && B`
- ☐ `(A == TRUE) && (B == TRUE)`
- ☐ `!(!A || B) == (A && !B)`
- ☐ `(!A || B)`

14. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

- ☐ `char "c";`
- ☐ `char c;`
- ☐ `int char;`
- ☐ `char 'c';`

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. 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`

17. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :

- ☐ un débogueur
- ☐ `printf("Valeur de n ? %d\n", n);`
- ☐ `printf("Valeur de n ? %g\n", n);`
- ☐ `scanf("%d", &n);`

18. Un enregistrement permet de grouper plusieurs valeurs dans :

- ☐ ses champs
- ☐ ses blocs
- ☐ ses chants
- ☐ ses cases

19. 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

20. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

- ☐ `n = factorielle();`
- ☐ `printf("%d", factorielle(n));`
- ☐ `n = factorielle(p, q);`
- ☐ `int factorielle(int 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. 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

2. 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

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

4. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :
   - ☐ `int pgcd(2);`
   - ☐ `n = pgcd(n, 3);`
   - ☐ `n = pgcd(int p, int q);`
   - ☐ `int n = pgcd();`

5. Vous utilisez une boucle `while` quand :
   - ☐ vous avez déjà fait un `for` dans le même programme principal
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous n'avez pas déclaré de fonction
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance

6. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
   - ☐ `int saisie_utilisateur();`
   - ☐ `void saisie_utilisateur(char c);`
   - ☐ `saisie_utilisateur(scanf(%d));`
   - ☐ `void saisie_utilisateur(int n);`

7. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses chants
   - ☐ ses blocs
   - ☐ ses champs
   - ☐ ses cases

8. Le code suivant :

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

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

9. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

   précède la fonction `main()`, alors on peut écrire en début de `main()` :
   - ☐ `int toto.n = 3;`
   - ☐ `struct toto_s toto;`
   - ☐ `toto_s struct z = {3, 0.5};`
   - ☐ `int struct toto_s = {3, -1e10};`
   - ☐ `toto_s n, x;`

10. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
    - ☐ une fonction déclarée mais non définie
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction appelée avant sa déclaration

11. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
    - ☐ `char ’c’;`
    - ☐ `char c;`
    - ☐ `int char;`
    - ☐ `char "c";`

12. 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

13. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
    - ☐ `n = factorielle(p, q);`
    - ☐ `n = factorielle();`
    - ☐ `printf("%d", factorielle(n));`
    - ☐ `int factorielle(int 2);`

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

15. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
    - ☐ `x = racine(2/3);`
    - ☐ `x = racine(racine(x)*racine(x));`
    - ☐ `x - 1 = racine(x);`
    - ☐ `x = racine(x * x) - racine(x);`
16. Le code suivant :

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

affichera :

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

17. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
    - ☐ `int n = carre();`
    - ☐ `int carre(2);`
    - ☐ `n = carre(int n);`
    - ☐ `n = carre(n);`
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 :

- ☐ 8
- ☐ 4
- ☐ 0
- ☐ 16

19. Une variable booléenne est un variable :
    - ☐ à laquelle une valeur vient d'être affectée
    - ☐ jamais nulle
    - ☐ qui est vraie ou fausse
    - ☐ réelle positive
    - ☐ NaN (not a number, qui n'est pas un nombre)
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. Le langage C est un langage
   - ☐ compilé
   - ☐ interprété
   - ☐ composé
   - ☐ lu, écrit, parlé
2. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
   - ☐ `x = racine(2/3);`
   - ☐ `x = racine(racine(x)*racine(x));`
   - ☐ `x = racine(x * x) - racine(x);`
   - ☐ `x - 1 = racine(x);`
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
   - ☐ 8 6 4 2
   - ☐ 0 2 4 6 8
4. 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 -Wall prog.exe -o prog.c`
   - ☐ `gcc prog.c -o -Wall prog.exe`
5. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
   - ☐ `n = factorielle(p, q);`
   - ☐ `int factorielle(int 2);`
   - ☐ `printf("%d", factorielle(n));`
   - ☐ `n = factorielle();`
6. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

   Alors l'expression `f(1)` prendra la valeur :
   - ☐ 4
   - ☐ 5
   - ☐ 0
   - ☐ 1
7. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

   Alors pour tester l'égalité de a et de b on utilise la condition :
   - ☐ `(a.n == b.n) && (a.x == b.x)`
   - ☐ `a == b`
   - ☐ `a{n, x} == b{n, x}`
   - ☐ `a = b`
8. Si cet avertissement apparaît à la compilation :
   `warning: implicit declaration of function ‘max’`
   , que doit-on chercher dans le programme ?
   - ☐ une fonction appelée avant sa déclaration
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une fonction déclarée mais non définie
9. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

   précède la fonction `main()`, alors on peut écrire en début de `main()` :
   - ☐ `toto_s struct z = {3, 0.5};`
   - ☐ `int struct toto_s = {3, -1e10};`
   - ☐ `toto_s n, x;`
   - ☐ `int toto.n = 3;`
   - ☐ `struct toto_s toto;`
10. Une variable booléenne est un variable :
    - ☐ qui est vraie ou fausse
    - ☐ jamais nulle
    - ☐ réelle positive
    - ☐ NaN (not a number, qui n'est pas un nombre)
    - ☐ à laquelle une valeur vient d'être affectée
11. 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
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
    - ☐ 42
    - ☐ 1
    - ☐ 6

13. 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

14. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

- ☐ un ordre quelconque
- ☐ dans lequel ces fonctions sont appelées dans le main
- ☐ alphabétique
- ☐ dans lequel vous avez déclaré ces fonction

15. 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

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

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

17. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :

- ☐ `int exposant(double n, int x);`
- ☐ `exposant(double x, int n, int r);`
- ☐ `void exposant(double x^n);`
- ☐ `double exposant(double x, int n);`

18. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

Alors l'affichage sera :

- ☐ `A`
- ☐ `B`
- ☐ `C`
- ☐ `b`

19. Avant de faire appel à une fonction il est nécessaire de :

- ☐ l'avoir définie
- ☐ l'avoir déclarée
- ☐ avoir défini une constante symbolique de la taille de cette fonction
- ☐ l'avoir déclarée et définie

20. Un enregistrement permet de grouper plusieurs valeurs dans :

- ☐ ses champs
- ☐ ses chants
- ☐ ses cases
- ☐ ses blocs

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 n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
   - ☐ un débogueur
   - ☐ `printf("Valeur de n ? %d\n", n);`
   - ☐ `scanf("%d", &n);`
   - ☐ `printf("Valeur de n ? %g\n", n);`
2. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
   - ☐ `int afficher_date(date_s d);`
   - ☐ `void afficher_date(struct date_s d);`
   - ☐ `void afficher_date(date_s d);`
   - ☐ `struct date_s afficher_date(struct date_s d);`
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 5 et la variable y vaut 0
   - ☐ la variable x vaut 0
   - ☐ le programme affiche "Faux"
   - ☐ la variable y vaut 5
4. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :

   - ☐ `int struct toto_s = {3, -1e10};`
   - ☐ `toto_s struct z = {3, 0.5};`
   - ☐ `toto_s n, x;`
   - ☐ `int toto.n = 3;`
   - ☐ `struct toto_s toto;`
5. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

   - ☐ 4
   - ☐ 3
   - ☐ 0
6. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
   - ☐ `int k;`
   - ☐ `loop i;`
   - ☐ `int %d;`
   - ☐ `int loop n;`
7. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

   - ☐ `i`
   - ☐ `A`
   - ☐ `ABCDEF`
   - ☐ `cccccc`
8. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il risque d'afficher bonjour à la place de coucou
   - ☐ il ne compile pas
   - ☐ il n'affiche rien
   - ☐ il comporte une boucle infinie
9. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
   - ☐ `x - 1 = racine(x);`
   - ☐ `x = racine(x * x) - racine(x);`
   - ☐ `x = racine(racine(x)*racine(x));`
   - ☐ `x = racine(2/3);`
10. Le langage C est un langage
    - ☐ composé
    - ☐ lu, écrit, parlé
    - ☐ compilé
    - ☐ interprété
11. Un fichier source est :
    - ☐ 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 illisible pour les humains
    - ☐ un document de référence du système
12. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ ouvrir un fichier texte
    - ☐ créer un fichier texte
    - ☐ créer un répertoire
    - ☐ changer de répertoire courant

13. Le code suivant :

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

affichera :

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

14. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?

- ☐ `(A == TRUE) && (B == TRUE)`
- ☐ `A && B`
- ☐ `!(!A || B) == (A && !B)`
- ☐ `(!A || B)`

15. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :

- ☐ `int factorielle(int x);`
- ☐ `int factorielle(double n);`
- ☐ `struct int factorielle(int n);`
- ☐ `int factorielle();`

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

17. 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

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

19. 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`

20. Avant de faire appel à une fonction il est nécessaire de :

- ☐ l'avoir déclarée
- ☐ l'avoir définie
- ☐ avoir défini une constante symbolique de la taille de cette fonction
- ☐ l'avoir déclarée et définie

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 `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
   - ☐ `int n = carre();`
   - ☐ `n = carre(n);`
   - ☐ `n = carre(int n);`
   - ☐ `int carre(2);`
2. 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
3. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
   - ☐ `printf("%d", factorielle(n));`
   - ☐ `n = factorielle(p, q);`
   - ☐ `n = factorielle();`
   - ☐ `int factorielle(int 2);`
4. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse harmonique
   - ☐ analyse lexicale
   - ☐ analyse syntaxique
   - ☐ analyse sémantique
5. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il ne compile pas
   - ☐ il n'affiche rien
   - ☐ il comporte une boucle infinie
   - ☐ il risque d'afficher bonjour à la place de coucou
6. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :
   - ☐ `n = pgcd(n, 3);`
   - ☐ `int n = pgcd();`
   - ☐ `int pgcd(2);`
   - ☐ `n = pgcd(int p, int q);`
7. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
   - ☐ `char afficher_menu(printf("menu"));`
   - ☐ `void afficher_menu();`
   - ☐ `double afficher_menu();`
   - ☐ `int afficher_menu(int char);`
   - ☐ `int afficher_menu();`
8. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

   Alors l'affichage sera :
   - ☐ `ABCDEF`
   - ☐ `cccccc`
   - ☐ `i`
   - ☐ `A`
9. Avant de faire appel à une fonction il est nécessaire de :
   - ☐ avoir défini une constante symbolique de la taille de cette fonction
   - ☐ l'avoir définie
   - ☐ l'avoir déclarée et définie
   - ☐ l'avoir déclarée
10. Si cette erreur apparaît à la compilation : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une fonction déclarée mais non définie
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
11. 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
12. Vous utilisez une boucle `while` quand :
    - ☐ vous n'avez pas déclaré de fonction
    - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
    - ☐ vous avez déjà fait un `for` dans le même programme principal
    - ☐ l'incrément de la variable de boucle n'est pas 1

13. 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 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
14. Le bus système sert à :
    - ☐ Arriver à l'heure en cours
    - ☐ Transférer des données et intructions entre processeur et mémoire
    - ☐ transporter les processus du tourniquet au processeur
    - ☐ Écrire des données sur le dique dur
15. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ dans lequel ces fonctions sont appelées dans le main
    - ☐ dans lequel vous avez déclaré ces fonction
    - ☐ alphabétique
    - ☐ un ordre quelconque
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 :
    - ☐ 8
    - ☐ 16
    - ☐ 4
    - ☐ 0
17. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
    - ☐ 8
    - ☐ 3
    - ☐ 111
    - ☐ 7
18. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

    Alors l'expression `g(0)` prendra la valeur :
    - ☐ 5
    - ☐ 0
    - ☐ 7
19. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

    Alors l'expression `f(0)` prendra la valeur :
    - ☐ 0
    - ☐ 4
    - ☐ 3
20. 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

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. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ dans lequel vous avez déclaré ces fonction
   - ☐ un ordre quelconque
   - ☐ alphabétique
   - ☐ dans lequel ces fonctions sont appelées dans le main

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

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

5. 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

6. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
   - ☐ `void saisie_utilisateur(int n);`
   - ☐ `void saisie_utilisateur(char c);`
   - ☐ `int saisie_utilisateur();`
   - ☐ `saisie_utilisateur(scanf(%d));`

7. 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 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

8. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
   - ☐ `double afficher_menu();`
   - ☐ `int afficher_menu();`
   - ☐ `int afficher_menu(int char);`
   - ☐ `void afficher_menu();`
   - ☐ `char afficher_menu(printf("menu"));`

9. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
   - ☐ `exposant(double x, int n, int r);`
   - ☐ `int exposant(double n, int x);`
   - ☐ `void exposant(double x^n);`
   - ☐ `double exposant(double x, int n);`

10. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
    - ☐ 0.5
    - ☐ 0
    - ☐ 1
    - ☐ 1.5

11. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

    - ☐ il comporte une boucle infinie
    - ☐ il n'affiche rien
    - ☐ il ne compile pas
    - ☐ il risque d'afficher bonjour à la place de coucou

12. 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

13. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
    - ☐ `int n = carre();`
    - ☐ `n = carre(n);`
    - ☐ `n = carre(int n);`
    - ☐ `int carre(2);`

14. Une variable booléenne est un variable :

- ☐ qui est vraie ou fausse
- ☐ à laquelle une valeur vient d'être affectée
- ☐ NaN (not a number, qui n'est pas un nombre)
- ☐ réelle positive
- ☐ jamais nulle

15. Le type des réels en C est :

- ☐ `int`
- ☐ `real`
- ☐ `double`
- ☐ `char`

16. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

- ☐ `int char;`
- ☐ `char 'c';`
- ☐ `char "c";`
- ☐ `char c;`

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 :

- ☐ 0
- ☐ 4
- ☐ 8
- ☐ 16

18. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

- ☐ `n = factorielle(p, q);`
- ☐ `int factorielle(int 2);`
- ☐ `n = factorielle();`
- ☐ `printf("%d", factorielle(n));`

19. 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`
- ☐ `0 1 2 0 1 2 3`

20. 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

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");
}
printf("Majeur\n");
```

affichera :

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

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. 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
- ☐ 16
- ☐ 8

4. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :

- ☐ `exposant(double x, int n, int r);`
- ☐ `int exposant(double n, int x);`
- ☐ `double exposant(double x, int n);`
- ☐ `void exposant(double x^n);`

5. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il ne compile pas
- ☐ il n'affiche rien
- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il comporte une boucle infinie

6. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :

- ☐ `printf("Valeur de n ? %g\n", n);`
- ☐ `printf("Valeur de n ? %d\n", n);`
- ☐ `scanf("%d", &n);`
- ☐ un débogueur

7. Vous utilisez une boucle `while` quand :

- ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
- ☐ vous n'avez pas déclaré de fonction
- ☐ vous avez déjà fait un `for` dans le même programme principal
- ☐ l'incrément de la variable de boucle n'est pas 1

8. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 0
- ☐ 3
- ☐ 4

9. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=3, b=5`

10. 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

11. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :

- ☐ `saisie_utilisateur(scanf(%d));`
- ☐ `void saisie_utilisateur(char c);`
- ☐ `void saisie_utilisateur(int n);`
- ☐ `int saisie_utilisateur();`

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

13. 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
- ☐ 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

14. Une variable booléenne est un variable :

- ☐ réelle positive
- ☐ jamais nulle
- ☐ NaN (not a number, qui n'est pas un nombre)
- ☐ à laquelle une valeur vient d'être affectée
- ☐ qui est vraie ou fausse

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. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :

- ☐ `x = mccarthy(n);`
- ☐ `n = mccarthy(p, q);`
- ☐ `int mccarthy(int 2);`
- ☐ `n = mccarthy();`

17. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 0
- ☐ 7
- ☐ 5

18. 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

19. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :

- ☐ 0.5
- ☐ 0
- ☐ 1
- ☐ 1.5

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. 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

2. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   - ☐ 5
   - ☐ 7
   - ☐ 0

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

5. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
   - ☐ `void saisie_utilisateur(char c);`
   - ☐ `saisie_utilisateur(scanf(%d));`
   - ☐ `int saisie_utilisateur();`
   - ☐ `void saisie_utilisateur(int n);`

6. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
   - ☐ `!(!A || B) == (A && !B)`
   - ☐ `(!A || B)`
   - ☐ `(A == TRUE) && (B == TRUE)`
   - ☐ `A && B`

7. Vous utilisez une boucle `while` quand :
   - ☐ vous avez déjà fait un `for` dans le même programme principal
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous n'avez pas déclaré de fonction

8. Le code suivant :

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

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

9. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

   Alors l'affichage sera :
   - ☐ `b`
   - ☐ `C`
   - ☐ `A`
   - ☐ `B`

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. Le langage C est un langage

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

12. 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

13. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 4
- ☐ 5
- ☐ 1
- ☐ 0

14. 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

15. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a{n, x} == b{n, x}`
- ☐ `a == b`
- ☐ `a = b`

16. 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

17. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il comporte une boucle infinie
- ☐ il n'affiche rien
- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il ne compile pas

18. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :

- ☐ `x = mccarthy(n);`
- ☐ `n = mccarthy();`
- ☐ `int mccarthy(int 2);`
- ☐ `n = mccarthy(p, q);`

19. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 4
- ☐ 3
- ☐ 0

20. 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)

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 la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
   - ☐ `n = mccarthy();`
   - ☐ `int mccarthy(int 2);`
   - ☐ `x = mccarthy(n);`
   - ☐ `n = mccarthy(p, q);`

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

3. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ alphabétique
   - ☐ un ordre quelconque
   - ☐ dans lequel vous avez déclaré ces fonction
   - ☐ dans lequel ces fonctions sont appelées dans le main

4. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
   - ☐ `n = carre(int n);`
   - ☐ `n = carre(n);`
   - ☐ `int n = carre();`
   - ☐ `int carre(2);`

5. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

   Alors pour tester l'égalité de a et de b on utilise la condition :
   - ☐ `a == b`
   - ☐ `a = b`
   - ☐ `(a.n == b.n) && (a.x == b.x)`
   - ☐ `a{n, x} == b{n, x}`

6. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
   - ☐ `printf("Valeur de n ? %d\n", n);`
   - ☐ un débogueur
   - ☐ `printf("Valeur de n ? %g\n", n);`
   - ☐ `scanf("%d", &n);`

7. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses champs
   - ☐ ses chants
   - ☐ ses blocs
   - ☐ ses cases

8. 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

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 sémantique
   - ☐ l'édition de liens
   - ☐ l'analyse des entrées clavier
   - ☐ l'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 `x = 5`
    - ☐ le terminal affiche "Faux"
    - ☐ le terminal affiche `5`
    - ☐ le terminal affiche `x = 2`

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

12. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
    - ☐ `exposant(double x, int n, int r);`
    - ☐ `double exposant(double x, int n);`
    - ☐ `void exposant(double x^n);`
    - ☐ `int exposant(double n, int x);`

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 le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=3, b=5`

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=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)`

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
- ☐ 20
- ☐ 6
- ☐ 3

17. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :

- ☐ 8
- ☐ 3
- ☐ 111
- ☐ 7

18. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 0
- ☐ 5
- ☐ 7

19. 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

20. Le code suivant :

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

affichera :

- ☐ `Mineur`
- ☐ rien
- ☐ `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. 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
2. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
   - ☐ 0
   - ☐ 1.5
   - ☐ 1
   - ☐ 0.5
3. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une fonction déclarée mais non définie
4. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :
   - ☐ 3
   - ☐ 0
   - ☐ 4
5. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
   - ☐ `x = mccarthy(n);`
   - ☐ `n = mccarthy(p, q);`
   - ☐ `n = mccarthy();`
   - ☐ `int mccarthy(int 2);`
6. 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
7. 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`
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'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
9. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

   Alors l'affichage sera :
   - ☐ `b`
   - ☐ `A`
   - ☐ `B`
   - ☐ `C`
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
    - ☐ que l'on veut voir tous les avertissements
    - ☐ qu'il faut lancer un déboggueur
11. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

    Alors l'expression `g(0)` prendra la valeur :
    - ☐ 5
    - ☐ 7
    - ☐ 0
12. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ changer de répertoire courant
    - ☐ ouvrir un fichier texte
    - ☐ créer un répertoire
    - ☐ créer un fichier texte

13. 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`

14. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :

- ☐ `scanf("%d", &n);`
- ☐ un débogueur
- ☐ `printf("Valeur de n ? %d\n", n);`
- ☐ `printf("Valeur de n ? %g\n", n);`

15. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :

- ☐ `n = carre(int n);`
- ☐ `int n = carre();`
- ☐ `int carre(2);`
- ☐ `n = carre(n);`

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
- ☐ 16
- ☐ 4
- ☐ 8

17. Le langage C est un langage

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

18. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?

- ☐ `(!A || B)`
- ☐ `!(!A || B) == (A && !B)`
- ☐ `A && B`
- ☐ `(A == TRUE) && (B == TRUE)`

19. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `a = b`
- ☐ `a == b`
- ☐ `a{n, x} == b{n, x}`
- ☐ `(a.n == b.n) && (a.x == b.x)`

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. 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
2. 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 variable non déclarée
   - ☐ une faute de frappe dans un appel de fonction
   - ☐ une directive préprocesseur `#include` manquante
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 = 5`
   - ☐ le terminal affiche `5`
   - ☐ le terminal affiche "Faux"
   - ☐ le terminal affiche `x = 2`
4. Sur unix (ou linux), la commande `mkdir` permet de :
   - ☐ créer un répertoire
   - ☐ créer un fichier texte
   - ☐ ouvrir un fichier texte
   - ☐ changer de répertoire courant
5. Le code suivant :

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

   affichera :
   - ☐ `Majeur`
   - ☐ `Mineur`
`Majeur`
   - ☐ rien
   - ☐ `Mineur`
6. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

   Quelle est la condition `cond` :
   - ☐ `a<=n<=b`
   - ☐ `(n<=a) && (n<=b)`
   - ☐ `(a<=n) && (n<=b)`
   - ☐ `(a<n) || (n>b)`
7. 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
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
   - ☐ le programme affiche "Faux"
   - ☐ la variable x vaut 3
   - ☐ la variable y vaut 5
9. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
   - ☐ `x - 1 = racine(x);`
   - ☐ `x = racine(2/3);`
   - ☐ `x = racine(racine(x)*racine(x));`
   - ☐ `x = racine(x * x) - racine(x);`
10. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

    Alors l'expression `f(0)` prendra la valeur :
    - ☐ 4
    - ☐ 3
    - ☐ 0

11. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
    - ☐ `n = mccarthy();`
    - ☐ `n = mccarthy(p, q);`
    - ☐ `int mccarthy(int 2);`
    - ☐ `x = mccarthy(n);`
12. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
    - ☐ une fonction appelée avant sa déclaration
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction déclarée mais non définie
13. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

    appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

    L'affichage dans le main est le suivant :
    - ☐ `f(a,b)=8, a=8, b=5`
    - ☐ `f(a,b)=13, a=8, b=5`
    - ☐ `f(a,b)=8, a=3, b=5`
    - ☐ `f(3,5)=8, a=3, b=5`
14. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ dans lequel ces fonctions sont appelées dans le main
    - ☐ dans lequel vous avez déclaré ces fonction
    - ☐ alphabétique
    - ☐ un ordre quelconque
15. 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>`
16. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
    - ☐ `(A == TRUE) && (B == TRUE)`
    - ☐ `!(!A || B) == (A && !B)`
    - ☐ `(!A || B)`
    - ☐ `A && B`
17. Vous utilisez une boucle `while` quand :
    - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
    - ☐ vous n'avez pas déclaré de fonction
    - ☐ vous avez déjà fait un `for` dans le même programme principal
    - ☐ l'incrément de la variable de boucle n'est pas 1
18. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
    - ☐ `int toto[5];`
    - ☐ `char tableau[5];`
    - ☐ `int tab[] = 5;`
    - ☐ `int[] new tableau(5);`
    - ☐ `int toto[taille=5];`
19. Le code suivant :

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

    affichera :
    - ☐ `Mineur`
`Majeur`
    - ☐ rien
    - ☐ `Mineur`
    - ☐ `Majeur`
20. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
    - ☐ `printf("Valeur de n ? %g\n", n);`
    - ☐ un débogueur
    - ☐ `printf("Valeur de n ? %d\n", n);`
    - ☐ `scanf("%d", &n);`

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. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

- ☐ `A`
- ☐ `i`
- ☐ `cccccc`
- ☐ `ABCDEF`

2. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :

- ☐ `double afficher_menu();`
- ☐ `int afficher_menu(int char);`
- ☐ `int afficher_menu();`
- ☐ `void afficher_menu();`
- ☐ `char afficher_menu(printf("menu"));`

3. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :

- ☐ `n = mccarthy(p, q);`
- ☐ `int mccarthy(int 2);`
- ☐ `x = mccarthy(n);`
- ☐ `n = mccarthy();`

4. 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

5. 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

6. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :

- ☐ `x = racine(x * x) - racine(x);`
- ☐ `x = racine(racine(x)*racine(x));`
- ☐ `x - 1 = racine(x);`
- ☐ `x = racine(2/3);`

7. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :

- ☐ `n = carre(n);`
- ☐ `n = carre(int n);`
- ☐ `int carre(2);`
- ☐ `int n = carre();`

8. L'écriture $\underline{101}$ en binaire correspond au nombre naturel :

- ☐ 101
- ☐ 4
- ☐ 3
- ☐ 5

9. 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
- ☐ 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

10. 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

11. Le code suivant :

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

affichera :

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

12. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 3
- ☐ 4
- ☐ 0

13. 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
14. Le type des réels en C est :
    - ☐ `char`
    - ☐ `real`
    - ☐ `double`
    - ☐ `int`
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 `5`
    - ☐ le terminal affiche `x = 5`
16. Vous utilisez une boucle `while` quand :
    - ☐ vous avez déjà fait un `for` dans le même programme principal
    - ☐ vous n'avez pas déclaré de fonction
    - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
    - ☐ l'incrément de la variable de boucle n'est pas 1
17. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
    - ☐ `char "c";`
    - ☐ `char 'c';`
    - ☐ `char c;`
    - ☐ `int char;`
18. 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`
19. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
    - ☐ analyse lexicale
    - ☐ analyse sémantique
    - ☐ analyse harmonique
    - ☐ analyse syntaxique
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. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

☐ 3

☐ 0

☐ 4

2. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :

☐ `char afficher_menu(printf("menu"));`

☐ `double afficher_menu();`

☐ `int afficher_menu();`

☐ `int afficher_menu(int char);`

☐ `void afficher_menu();`

3. 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`

4. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

☐ `f(a,b)=13, a=8, b=5`

☐ `f(3,5)=8, a=3, b=5`

☐ `f(a,b)=8, a=8, b=5`

☐ `f(a,b)=8, a=3, b=5`

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
```

☐ la case mémoire 8 contiendra 16

☐ la case mémoire 8 contiendra 0

☐ le bus explose

☐ le terminal affiche 8

6. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :

☐ `int n = carre();`

☐ `n = carre(n);`

☐ `n = carre(int n);`

☐ `int carre(2);`

7. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :

☐ `x = racine(2/3);`

☐ `x = racine(racine(x)*racine(x));`

☐ `x - 1 = racine(x);`

☐ `x = racine(x * x) - racine(x);`

8. Vous utilisez une boucle `while` quand :

☐ vous n'avez pas déclaré de fonction

☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance

☐ vous avez déjà fait un `for` dans le même programme principal

☐ l'incrément de la variable de boucle n'est pas 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 8

☐ 0 2 4 6

10. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :

☐ `int factorielle(double n);`

☐ `int factorielle(int x);`

☐ `int factorielle();`

☐ `struct int factorielle(int n);`

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. 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

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 doubler la mémoire disponible
- ☐ de ne pas perdre de temps avec la commutation de contexte

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

15. Le langage C est un langage

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

16. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

- ☐ `int char;`
- ☐ `char 'c';`
- ☐ `char "c";`
- ☐ `char c;`

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 = 5`
- ☐ `j = 4`
- ☐ `j = 0`

18. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :

- ☐ `void saisie_utilisateur(int n);`
- ☐ `int saisie_utilisateur();`
- ☐ `saisie_utilisateur(scanf(%d));`
- ☐ `void saisie_utilisateur(char c);`

19. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

- ☐ `n = factorielle();`
- ☐ `n = factorielle(p, q);`
- ☐ `int factorielle(int 2);`
- ☐ `printf("%d", factorielle(n));`

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 0 1 2 3`
- ☐ `0 1 2 3 0 1 2`
- ☐ `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. Une variable booléenne est un variable :
   - ☐ jamais nulle
   - ☐ à laquelle une valeur vient d'être affectée
   - ☐ NaN (not a number, qui n'est pas un nombre)
   - ☐ qui est vraie ou fausse
   - ☐ réelle positive
2. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il risque d'afficher bonjour à la place de coucou
   - ☐ il comporte une boucle infinie
   - ☐ il n'affiche rien
   - ☐ il ne compile pas
3. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :
   - ☐ 0
   - ☐ 3
   - ☐ 4
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 des entrées clavier
   - ☐ l'édition de liens
   - ☐ l'analyse sémantique
5. 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
6. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
   - ☐ `void afficher_date(struct date_s d);`
   - ☐ `int afficher_date(date_s d);`
   - ☐ `void afficher_date(date_s d);`
   - ☐ `struct date_s afficher_date(struct date_s d);`
7. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   - ☐ `f(3,5)=8, a=3, b=5`
   - ☐ `f(a,b)=8, a=8, b=5`
   - ☐ `f(a,b)=13, a=8, b=5`
   - ☐ `f(a,b)=8, a=3, b=5`
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 terminal affiche 8
   - ☐ la case mémoire 8 contiendra 0
   - ☐ le bus explose
   - ☐ la case mémoire 8 contiendra 16
9. 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
10. Si cette erreur apparaît à la compilation : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction déclarée mais non définie
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction appelée avant sa déclaration
11. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function 'max'` , que doit-on chercher dans le programme ?
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une fonction déclarée mais non définie
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une directive préprocesseur `#include` manquante

12. Le code suivant :

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

affichera :

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

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

14. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
    - ☐ `char 'c';`
    - ☐ `int char;`
    - ☐ `char "c";`
    - ☐ `char c;`

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. 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

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 :

- ☐ 3
- ☐ 16
- ☐ 6
- ☐ 20

18. 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`

19. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
    - ☐ 3
    - ☐ 111
    - ☐ 7
    - ☐ 8

20. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a == b`
- ☐ `a = b`
- ☐ `a{n, x} == b{n, x}`

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

2. 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

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. 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

5. 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

6. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
   - ☐ `int factorielle();`
   - ☐ `int factorielle(int x);`
   - ☐ `int factorielle(double n);`
   - ☐ `struct int factorielle(int n);`

7. 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`

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

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

10. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

    Alors l'affichage sera :
    - ☐ `cccccc`
    - ☐ `ABCDEF`
    - ☐ `A`
    - ☐ `i`

11. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

    appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(a,b)=13, a=8, b=5`

12. L'écriture 111 en binaire correspond au nombre naturel :
    - ☐ 7
    - ☐ 3
    - ☐ 8
    - ☐ 111
13. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ dans lequel vous avez déclaré ces fonction
    - ☐ un ordre quelconque
    - ☐ alphabétique
    - ☐ dans lequel ces fonctions sont appelées dans le main
14. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a{n, x} == b{n, x}`
- ☐ `a == b`
- ☐ `a = b`

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.c -o prog.exe`
    - ☐ `gcc -Wall prog.exe -o prog.c`
16. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il ne compile pas
- ☐ il comporte une boucle infinie
- ☐ il n'affiche rien
- ☐ il risque d'afficher bonjour à la place de coucou

17. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
    - ☐ `(A == TRUE) && (B == TRUE)`
    - ☐ `(!A || B)`
    - ☐ `A && B`
    - ☐ `!(!A || B) == (A && !B)`
18. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
    - ☐ `int pgcd(int x, int x);`
    - ☐ `int pgcd(int y, int x);`
    - ☐ `void pgcd(int x, int y);`
    - ☐ `int pgcd(int x, y);`
19. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 1
- ☐ 0
- ☐ 5
- ☐ 4

20. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 5
- ☐ 0
- ☐ 7

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'écriture 101 en binaire correspond au nombre naturel :
   - ☐ 101
   - ☐ 4
   - ☐ 3
   - ☐ 5
2. 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
3. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
   - ☐ `void afficher_date(struct date_s d);`
   - ☐ `void afficher_date(date_s d);`
   - ☐ `struct date_s afficher_date(struct date_s d);`
   - ☐ `int afficher_date(date_s d);`
4. Le code suivant :

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

   affichera :
   - ☐ `Mineur`
   - ☐ rien
   - ☐ `Majeur`
   - ☐ `Mineur`
     `Majeur`
5. 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
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. Si cette erreur apparaît à la compilation :
   `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une fonction déclarée mais non définie
8. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   - ☐ 7
   - ☐ 5
   - ☐ 0
9. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
   - ☐ `saisie_utilisateur(scanf(%d));`
   - ☐ `void saisie_utilisateur(int n);`
   - ☐ `void saisie_utilisateur(char c);`
   - ☐ `int saisie_utilisateur();`
10. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
    - ☐ `int pgcd(int y, int x);`
    - ☐ `int pgcd(int x, int x);`
    - ☐ `int pgcd(int x, y);`
    - ☐ `void pgcd(int x, int y);`
11. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses chants
    - ☐ ses champs
    - ☐ ses blocs
    - ☐ ses cases
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 :
    - ☐ 4
    - ☐ 8
    - ☐ 16
    - ☐ 0
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. Vous utilisez une boucle `while` quand :
    - ☐ vous avez déjà fait un `for` dans le même programme principal
    - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
    - ☐ l'incrément de la variable de boucle n'est pas 1
    - ☐ vous n'avez pas déclaré de fonction
15. Quel est l'opérateur de différence en C :
    - ☐ `!`
    - ☐ `<>`
    - ☐ `!=`
    - ☐ $\neq$
16. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
    - ☐ `x = mccarthy(n);`
    - ☐ `n = mccarthy();`
    - ☐ `n = mccarthy(p, q);`
    - ☐ `int mccarthy(int 2);`
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 y vaut 5
    - ☐ le programme affiche "Faux"
    - ☐ la variable x vaut 5 et la variable y vaut 0
18. Le code suivant :

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

    affichera :
    - ☐ `Mineur`
`Majeur`
    - ☐ `Majeur`
    - ☐ rien
    - ☐ `Mineur`
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 a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

    Alors pour tester l'égalité de a et de b on utilise la condition :
    - ☐ `a = b`
    - ☐ `(a.n == b.n) && (a.x == b.x)`
    - ☐ `a == b`
    - ☐ `a{n, x} == b{n, x}`

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 la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 5
- ☐ 1
- ☐ 0
- ☐ 4

2. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses champs
   - ☐ ses blocs
   - ☐ ses chants
   - ☐ ses cases
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'édition de liens
   - ☐ l'analyse des entrées clavier
   - ☐ l'analyse sémantique
   - ☐ l'analyse harmonique
4. Vous utilisez une boucle `while` quand :
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous avez déjà fait un `for` dans le même programme principal
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ vous n'avez pas déclaré de fonction
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. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=13, a=8, b=5`

7. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
   - ☐ 0.5
   - ☐ 1.5
   - ☐ 1
   - ☐ 0
8. Le code suivant :

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

affichera :

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

9. 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
10. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
    - ☐ `printf("Valeur de n ? %d\n", n);`
    - ☐ un débogueur
    - ☐ `printf("Valeur de n ? %g\n", n);`
    - ☐ `scanf("%d", &n);`
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
- ☐ 0
- ☐ 16

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

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. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :

- ☐ `struct toto_s toto;`
- ☐ `int toto.n = 3;`
- ☐ `toto_s n, x;`
- ☐ `toto_s struct z = {3, 0.5};`
- ☐ `int struct toto_s = {3, -1e10};`

15. L'écriture 111 en binaire correspond au nombre naturel :
    - ☐ 7
    - ☐ 3
    - ☐ 111
    - ☐ 8
16. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 3
- ☐ 4
- ☐ 0

17. 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 illisible pour les humains
    - ☐ un document qui doit être protégé
18. 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`
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
- ☐ 4 3 2 1
- ☐ 1 2 3 4

20. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
    - ☐ `int pgcd(int x, int x);`
    - ☐ `int pgcd(int y, int x);`
    - ☐ `void pgcd(int x, int y);`
    - ☐ `int pgcd(int x, y);`

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 ?
   - ☐ une accolade manquante
   - ☐ une accolade en trop
   - ☐ un point-virgule manquant
   - ☐ un point-virgule en trop
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. Vous utilisez une boucle `while` quand :
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous avez déjà fait un `for` dans le même programme principal
   - ☐ vous n'avez pas déclaré de fonction
4. L'écriture 111 en binaire correspond au nombre naturel :
   - ☐ 3
   - ☐ 8
   - ☐ 7
   - ☐ 111
5. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il comporte une boucle infinie
   - ☐ il ne compile pas
   - ☐ il risque d'afficher bonjour à la place de coucou
   - ☐ il n'affiche rien
6. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :
   - ☐ 4
   - ☐ 3
   - ☐ 0
7. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
   - ☐ `int char;`
   - ☐ `char "c";`
   - ☐ `char c;`
   - ☐ `char 'c';`
8. Sur unix (ou linux), la commande `mkdir` permet de :
   - ☐ changer de répertoire courant
   - ☐ ouvrir un fichier texte
   - ☐ créer un répertoire
   - ☐ créer un fichier texte
9. 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
10. 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
11. 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 2 4 6
    - ☐ 0 1 2 3 4 5 6
12. Si cette erreur apparaît à la compilation :
    `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction appelée avant sa déclaration
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction déclarée mais non définie
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 `****`
    - ☐ la variable x vaut $-\frac{1}{2}$
    - ☐ la variable x vaut 16
    - ☐ le programme affiche x

14. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ alphabétique
    - ☐ un ordre quelconque
    - ☐ dans lequel vous avez déclaré ces fonction
    - ☐ dans lequel ces fonctions sont appelées dans le main
15. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses champs
    - ☐ ses chants
    - ☐ ses blocs
    - ☐ ses cases
16. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 4
- ☐ 5
- ☐ 1
- ☐ 0

17. Si cet avertissement apparaît à la compilation :
`warning: implicit declaration of function ‘max’`
, que doit-on chercher dans le programme ?
    - ☐ une fonction déclarée mais non définie
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction appelée avant sa déclaration
18. Avant de faire appel à une fonction il est nécessaire de :
    - ☐ l'avoir déclarée et définie
    - ☐ l'avoir déclarée
    - ☐ l'avoir définie
    - ☐ avoir défini une constante symbolique de la taille de cette fonction
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 ?

- ☐ `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`

20. 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);`

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. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
   - ☐ `printf("%d", factorielle(n));`
   - ☐ `int factorielle(int 2);`
   - ☐ `n = factorielle();`
   - ☐ `n = factorielle(p, q);`
3. 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
4. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

   Alors l'affichage sera :
   - ☐ `A`
   - ☐ `cccccc`
   - ☐ `i`
   - ☐ `ABCDEF`
5. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :
   - ☐ 0
   - ☐ 4
   - ☐ 3
6. 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
7. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   - ☐ `f(3,5)=8, a=3, b=5`
   - ☐ `f(a,b)=8, a=3, b=5`
   - ☐ `f(a,b)=13, a=8, b=5`
   - ☐ `f(a,b)=8, a=8, b=5`
8. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

   Quelle est la condition `cond` :
   - ☐ `(a<=n) && (n<=b)`
   - ☐ `(a<n) || (n>b)`
   - ☐ `(n<=a) && (n<=b)`
   - ☐ `a<=n<=b`
9. 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
   - ☐ chacun son tour, après que le processus précédent a terminé
   - ☐ tous ensemble
10. Le type des réels en C est :
    - ☐ `real`
    - ☐ `int`
    - ☐ `char`
    - ☐ `double`
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.c -o prog.exe`
    - ☐ `gcc -Wall prog.exe -o prog.c`

12. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :
    - ☐ `int n = pgcd();`
    - ☐ `n = pgcd(int p, int q);`
    - ☐ `n = pgcd(n, 3);`
    - ☐ `int pgcd(2);`
13. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une fonction déclarée mais non définie
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
14. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
    - ☐ `char afficher_menu(printf("menu"));`
    - ☐ `void afficher_menu();`
    - ☐ `double afficher_menu();`
    - ☐ `int afficher_menu(int char);`
    - ☐ `int afficher_menu();`
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=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)`
16. Vous utilisez une boucle `while` quand :
    - ☐ l'incrément de la variable de boucle n'est pas 1
    - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
    - ☐ vous avez déjà fait un `for` dans le même programme principal
    - ☐ vous n'avez pas déclaré de fonction
17. Avant de faire appel à une fonction il est nécessaire de :
    - ☐ l'avoir définie
    - ☐ l'avoir déclarée et définie
    - ☐ l'avoir déclarée
    - ☐ avoir défini une constante symbolique de la taille de cette fonction
18. Si a et b sont deux variables de type :

    ```
    struct toto_s
    {
      int n;
      double x;
    };
    ```

    Alors pour tester l'égalité de a et de b on utilise la condition :
    - ☐ `(a.n == b.n) && (a.x == b.x)`
    - ☐ `a = b`
    - ☐ `a == b`
    - ☐ `a{n, x} == b{n, x}`
19. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
    - ☐ `int carre(2);`
    - ☐ `int n = carre();`
    - ☐ `n = carre(n);`
    - ☐ `n = carre(int n);`
20. 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

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. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

Alors l'affichage sera :

☐ `A`

☐ `b`

☐ `C`

☐ `B`

2. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :

☐ `void afficher_menu();`

☐ `int afficher_menu();`

☐ `char afficher_menu(printf("menu"));`

☐ `int afficher_menu(int char);`

☐ `double afficher_menu();`

3. 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

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

☐ `#appart <stdlib.h>`

☐ `#include <studio.h>`

☐ `#include <studlib.h>`

☐ `#include <stdio.h>`

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. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :

☐ `n = pgcd(int p, int q);`

☐ `int n = pgcd();`

☐ `n = pgcd(n, 3);`

☐ `int pgcd(2);`

7. 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

8. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?

☐ `(!A || B)`

☐ `(A == TRUE) && (B == TRUE)`

☐ `A && B`

☐ `!(!A || B) == (A && !B)`

9. 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

10. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

☐ il ne compile pas

☐ il n'affiche rien

☐ il risque d'afficher bonjour à la place de coucou

☐ il comporte une boucle infinie

11. 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

12. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :

☐ `x = racine(racine(x)*racine(x));`

☐ `x = racine(2/3);`

☐ `x - 1 = racine(x);`

☐ `x = racine(x * x) - racine(x);`

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

14. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ dans lequel ces fonctions sont appelées dans le main
    - ☐ dans lequel vous avez déclaré ces fonction
    - ☐ un ordre quelconque
    - ☐ alphabétique

15. Le code suivant :

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

affichera :

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

16. 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

17. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
    - ☐ 0.5
    - ☐ 0
    - ☐ 1.5
    - ☐ 1

18. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=3, b=5`

19. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
    - ☐ `void saisie_utilisateur(int n);`
    - ☐ `int saisie_utilisateur();`
    - ☐ `void saisie_utilisateur(char c);`
    - ☐ `saisie_utilisateur(scanf(%d));`

20. 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

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 une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
   - ☐ `int afficher_menu(int char);`
   - ☐ `double afficher_menu();`
   - ☐ `char afficher_menu(printf("menu"));`
   - ☐ `void afficher_menu();`
   - ☐ `int afficher_menu();`
2. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
   - ☐ un débogueur
   - ☐ `printf("Valeur de n ? %g\n", n);`
   - ☐ `scanf("%d", &n);`
   - ☐ `printf("Valeur de n ? %d\n", n);`
3. 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
4. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   - ☐ `f(a,b)=8, a=3, b=5`
   - ☐ `f(3,5)=8, a=3, b=5`
   - ☐ `f(a,b)=8, a=8, b=5`
   - ☐ `f(a,b)=13, a=8, b=5`
5. 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
6. L'écriture $\underline{101}$ en binaire correspond au nombre naturel :
   - ☐ 101
   - ☐ 5
   - ☐ 4
   - ☐ 3
7. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
   - ☐ `saisie_utilisateur(scanf(%d));`
   - ☐ `void saisie_utilisateur(char c);`
   - ☐ `void saisie_utilisateur(int n);`
   - ☐ `int saisie_utilisateur();`
8. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   - ☐ 5
   - ☐ 0
   - ☐ 7
9. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
   - ☐ `char 'c';`
   - ☐ `char "c";`
   - ☐ `int char;`
   - ☐ `char c;`
10. Le type des réels en C est :
    - ☐ `real`
    - ☐ `double`
    - ☐ `char`
    - ☐ `int`
11. Si cette erreur apparaît à la compilation : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une fonction déclarée mais non définie
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une directive préprocesseur `#include` manquante
12. 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

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

14. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
    - ☐ `struct int factorielle(int n);`
    - ☐ `int factorielle(int x);`
    - ☐ `int factorielle(double n);`
    - ☐ `int factorielle();`

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

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

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

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

20. 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

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)
   - ☐ un battement d'horloge processeur
   - ☐ l'instruction qui met fin à un programme
   - ☐ la longueur d'un mot mémoire
2. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
   - ☐ `void afficher_date(struct date_s d);`
   - ☐ `int afficher_date(date_s d);`
   - ☐ `void afficher_date(date_s d);`
   - ☐ `struct date_s afficher_date(struct date_s d);`
3. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
   - ☐ `int saisie_utilisateur();`
   - ☐ `saisie_utilisateur(scanf(%d));`
   - ☐ `void saisie_utilisateur(char c);`
   - ☐ `void saisie_utilisateur(int n);`
4. Si cette erreur apparaît à la compilation : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
   - ☐ une fonction appelée avant sa déclaration
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une fonction déclarée mais non définie
5. Un fichier source est :
   - ☐ 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
   - ☐ un document de référence du système
6. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
   - ☐ `int pgcd(int x, y);`
   - ☐ `int pgcd(int y, int x);`
   - ☐ `int pgcd(int x, int x);`
   - ☐ `void pgcd(int x, int y);`
7. Le code suivant :

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

   affichera :
   - ☐ `Mineur`
`Majeur`
   - ☐ `Mineur`
   - ☐ rien
   - ☐ `Majeur`
8. Le type des réels en C est :
   - ☐ `double`
   - ☐ `int`
   - ☐ `char`
   - ☐ `real`
9. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
   - ☐ 111
   - ☐ 7
   - ☐ 8
   - ☐ 3
10. 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
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
    - ☐ 6
    - ☐ 0
    - ☐ 15
12. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
    - ☐ `struct int factorielle(int n);`
    - ☐ `int factorielle(int x);`
    - ☐ `int factorielle(double n);`
    - ☐ `int factorielle();`
13. 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>`

14. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 7
- ☐ 0
- ☐ 5

15. 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

16. 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`

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 :

- ☐ 8
- ☐ 4
- ☐ 0
- ☐ 16

18. 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

19. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il n'affiche rien
- ☐ il ne compile pas
- ☐ il comporte une boucle infinie

20. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

- ☐ `int char;`
- ☐ `char 'c';`
- ☐ `char "c";`
- ☐ `char 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 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 fichier texte qui sera traduit en instructions processeur
   - ☐ un document illisible pour les humains
2. Le langage C est un langage
   - ☐ interprété
   - ☐ lu, écrit, parlé
   - ☐ composé
   - ☐ compilé
3. 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)`
4. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
   - ☐ `void saisie_utilisateur(int n);`
   - ☐ `int saisie_utilisateur();`
   - ☐ `void saisie_utilisateur(char c);`
   - ☐ `saisie_utilisateur(scanf(%d));`
5. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ alphabétique
   - ☐ dans lequel vous avez déclaré ces fonction
   - ☐ dans lequel ces fonctions sont appelées dans le main
   - ☐ un ordre quelconque
6. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
   - ☐ `double exposant(double x, int n);`
   - ☐ `exposant(double x, int n, int r);`
   - ☐ `int exposant(double n, int x);`
   - ☐ `void exposant(double x^n);`
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
   - ☐ 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
8. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   - ☐ 7
   - ☐ 5
   - ☐ 0
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
   - ☐ une unité de calcul spécialisée de l'ordinateur
   - ☐ un composant qui contient la liste des fichiers du système
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 :
    - ☐ 6
    - ☐ 0
    - ☐ 10
    - ☐ 15
11. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :
    - ☐ `int n = pgcd();`
    - ☐ `n = pgcd(n, 3);`
    - ☐ `n = pgcd(int p, int q);`
    - ☐ `int pgcd(2);`
12. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
    - ☐ `int afficher_menu();`
    - ☐ `char afficher_menu(printf("menu"));`
    - ☐ `int afficher_menu(int char);`
    - ☐ `void afficher_menu();`
    - ☐ `double afficher_menu();`
13. Vous utilisez une boucle `while` quand :
    - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
    - ☐ vous n'avez pas déclaré de fonction
    - ☐ vous avez déjà fait un `for` dans le même programme principal
    - ☐ l'incrément de la variable de boucle n'est pas 1

14. Le code suivant :

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

affichera :

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

15. Une variable booléenne est un variable :
    - ☐ NaN (not a number, qui n'est pas un nombre)
    - ☐ réelle positive
    - ☐ à laquelle une valeur vient d'être affectée
    - ☐ jamais nulle
    - ☐ qui est vraie ou fausse

16. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
    - ☐ `char 'c';`
    - ☐ `int char;`
    - ☐ `char "c";`
    - ☐ `char c;`

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

18. Avant de faire appel à une fonction il est nécessaire de :
    - ☐ l'avoir déclarée
    - ☐ l'avoir définie
    - ☐ avoir défini une constante symbolique de la taille de cette fonction
    - ☐ l'avoir déclarée et définie

19. 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

20. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function 'max'` , que doit-on chercher dans le programme ?
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction déclarée mais non définie

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=1;i<5;i=i+1)`
   - ☐ `for(i=0;i<=5;i=i+1)`
   - ☐ `for(i=0;i<5;i=i+1)`
2. L'écriture 111 en binaire correspond au nombre naturel :
   - ☐ 111
   - ☐ 8
   - ☐ 3
   - ☐ 7
3. Une variable booléenne est un variable :
   - ☐ réelle positive
   - ☐ qui est vraie ou fausse
   - ☐ NaN (not a number, qui n'est pas un nombre)
   - ☐ jamais nulle
   - ☐ à laquelle une valeur vient d'être affectée
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. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses chants
   - ☐ ses champs
   - ☐ ses blocs
   - ☐ ses cases
6. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   - ☐ 5
   - ☐ 7
   - ☐ 0
7. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
   - ☐ `int char;`
   - ☐ `char c;`
   - ☐ `char 'c';`
   - ☐ `char "c";`
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 des entrées clavier
   - ☐ l'analyse sémantique
   - ☐ l'analyse harmonique
   - ☐ l'édition de liens
9. 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>`
10. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

    Quelle est la condition `cond` :
    - ☐ `(n<=a) && (n<=b)`
    - ☐ `(a<=n) && (n<=b)`
    - ☐ `a<=n<=b`
    - ☐ `(a<n) || (n>b)`
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 :
    - ☐ 15
    - ☐ 10
    - ☐ 6
    - ☐ 0
12. Le type des réels en C est :
    - ☐ `char`
    - ☐ `double`
    - ☐ `real`
    - ☐ `int`

13. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a{n, x} == b{n, x}`
- ☐ `a = b`
- ☐ `a == b`

14. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 3
- ☐ 0
- ☐ 4

15. Le code suivant :

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

affichera :

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

16. Le code suivant :

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

affichera :

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

17. 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 = 0`
- ☐ `j = 4`

18. 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`

19. 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

20. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :

- ☐ `void afficher_menu();`
- ☐ `char afficher_menu(printf("menu"));`
- ☐ `int afficher_menu();`
- ☐ `double afficher_menu();`
- ☐ `int afficher_menu(int char);`

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. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :

☐ `int mccarthy(int 2);`
☐ `x = mccarthy(n);`
☐ `n = mccarthy();`
☐ `n = mccarthy(p, q);`

3. 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

4. Vous utilisez une boucle `while` quand :

☐ vous avez déjà fait un `for` dans le même programme principal
☐ l'incrément de la variable de boucle n'est pas 1
☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
☐ vous n'avez pas déclaré de fonction

5. 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`

6. Le type des réels en C est :

☐ `char`
☐ `real`
☐ `int`
☐ `double`

7. 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

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

9. Le code suivant :

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

affichera :

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

10. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :

☐ `int factorielle(double n);`
☐ `int factorielle(int x);`
☐ `struct int factorielle(int n);`
☐ `int factorielle();`

11. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

☐ dans lequel ces fonctions sont appelées dans le main
☐ alphabétique
☐ dans lequel vous avez déclaré ces fonction
☐ un ordre quelconque

12. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que `x` est une variable réelle définie et initialisée, il est incorrect d'écrire :

☐ `x = racine(racine(x)*racine(x));`
☐ `x = racine(2/3);`
☐ `x = racine(x * x) - racine(x);`
☐ `x - 1 = racine(x);`

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 0 1 1 2 2`
- ☐ `0 1 0 1 0 1`

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. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `(a<=n) && (n<=b)`
- ☐ `a<=n<=b`
- ☐ `(n<=a) && (n<=b)`
- ☐ `(a<n) || (n>b)`

16. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :

- ☐ `int exposant(double n, int x);`
- ☐ `void exposant(double x^n);`
- ☐ `double exposant(double x, int n);`
- ☐ `exposant(double x, int n, int r);`

17. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :

- ☐ `printf("Valeur de n ? %g\n", n);`
- ☐ `scanf("%d", &n);`
- ☐ `printf("Valeur de n ? %d\n", n);`
- ☐ un débogueur

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

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

19. 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

20. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 0
- ☐ 5
- ☐ 7

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 a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `a = b`
- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a{n, x} == b{n, x}`
- ☐ `a == b`

2. 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

3. 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`

4. Le langage C est un langage

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

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
- ☐ 16
- ☐ 0
- ☐ 8

6. Le code suivant :

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

affichera :

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

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
- ☐ 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

8. Vous utilisez une boucle `while` quand :

- ☐ vous avez déjà fait un `for` dans le même programme principal
- ☐ l'incrément de la variable de boucle n'est pas 1
- ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
- ☐ vous n'avez pas déclaré de fonction

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. 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
- ☐ 3
- ☐ 6

11. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

- ☐ `int factorielle(int 2);`
- ☐ `n = factorielle();`
- ☐ `printf("%d", factorielle(n));`
- ☐ `n = factorielle(p, q);`

12. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que `x` est une variable réelle définie et initialisée, il est incorrect d'écrire :

- ☐ `x = racine(x * x) - racine(x);`
- ☐ `x - 1 = racine(x);`
- ☐ `x = racine(racine(x)*racine(x));`
- ☐ `x = racine(2/3);`

13. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 0
- ☐ 5
- ☐ 7

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

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

15. Si cet avertissement apparaît à la compilation :
`warning: implicit declaration of function ‘max’`
, que doit-on chercher dans le programme ?

- ☐ une directive préprocesseur `#include` manquante
- ☐ un désaccord entre la déclaration et la définition d'une fonction
- ☐ une fonction appelée avant sa déclaration
- ☐ une fonction déclarée mais non définie

16. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :

- ☐ `int pgcd(2);`
- ☐ `n = pgcd(int p, int q);`
- ☐ `n = pgcd(n, 3);`
- ☐ `int n = pgcd();`

17. L'ordonnancement par tourniquet permet :

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

18. Le type des réels en C est :

- ☐ `real`
- ☐ `double`
- ☐ `int`
- ☐ `char`

19. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il comporte une boucle infinie
- ☐ il n'affiche rien
- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il ne compile pas

20. L'écriture $\underline{101}$ en binaire correspond au nombre naturel :

- ☐ 4
- ☐ 5
- ☐ 101
- ☐ 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. Avant de faire appel à une fonction il est nécessaire de :
   - ☐ l'avoir déclarée
   - ☐ l'avoir déclarée et définie
   - ☐ avoir défini une constante symbolique de la taille de cette fonction
   - ☐ l'avoir définie
2. Vous utilisez une boucle `while` quand :
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ vous n'avez pas déclaré de fonction
   - ☐ vous avez déjà fait un `for` dans le même programme principal
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 :
   - ☐ certaines données de la mémoire de travail
   - ☐ en temps d'accès
   - ☐ les fichiers du disque
   - ☐ des processus
4. 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`
5. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   - ☐ 7
   - ☐ 0
   - ☐ 5
6. 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
7. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
   - ☐ `struct date_s afficher_date(struct date_s d);`
   - ☐ `void afficher_date(struct date_s d);`
   - ☐ `int afficher_date(date_s d);`
   - ☐ `void afficher_date(date_s d);`
8. Sous unix (ou linux), la commande `cd` permet de :
   - ☐ changer de répertoire courant
   - ☐ ouvir un bureau partagé (common desktop)
   - ☐ détruire un fichier
   - ☐ jouer de la musique
   - ☐ récupérer un programme arrêté avec la commande `ab`
9. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
   - ☐ `x - 1 = racine(x);`
   - ☐ `x = racine(racine(x)*racine(x));`
   - ☐ `x = racine(x * x) - racine(x);`
   - ☐ `x = racine(2/3);`
10. L'écriture $\underline{101}$ en binaire correspond au nombre naturel :
    - ☐ 4
    - ☐ 5
    - ☐ 101
    - ☐ 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 :
    - ☐ 8
    - ☐ 4
    - ☐ 0
    - ☐ 16
12. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
    - ☐ une fonction déclarée mais non définie
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction appelée avant sa déclaration
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
13. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

    - ☐ il risque d'afficher bonjour à la place de coucou
    - ☐ il n'affiche rien
    - ☐ il ne compile pas
    - ☐ il comporte une boucle infinie

14. 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`

15. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :

- ☐ `void saisie_utilisateur(char c);`
- ☐ `saisie_utilisateur(scanf(%d));`
- ☐ `int saisie_utilisateur();`
- ☐ `void saisie_utilisateur(int n);`

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
- ☐ 3
- ☐ 16

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. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

- ☐ `n = factorielle();`
- ☐ `n = factorielle(p, q);`
- ☐ `printf("%d", factorielle(n));`
- ☐ `int factorielle(int 2);`

19. Le code suivant :

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

affichera :

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

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. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses blocs
   - ☐ ses cases
   - ☐ ses champs
   - ☐ ses chants
2. Le bus système sert à :
   - ☐ Arriver à l'heure en cours
   - ☐ Écrire des données sur le dique dur
   - ☐ Transférer des données et intructions entre processeur et mémoire
   - ☐ transporter les processus du tourniquet au processeur
3. Avant de faire appel à une fonction il est nécessaire de :
   - ☐ avoir défini une constante symbolique de la taille de cette fonction
   - ☐ l'avoir définie
   - ☐ l'avoir déclarée
   - ☐ l'avoir déclarée et définie
4. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
   - ☐ `printf("Valeur de n ? %g\n", n);`
   - ☐ `printf("Valeur de n ? %d\n", n);`
   - ☐ `scanf("%d", &n);`
   - ☐ un débogueur
5. 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
6. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
   - ☐ `x - 1 = racine(x);`
   - ☐ `x = racine(x * x) - racine(x);`
   - ☐ `x = racine(2/3);`
   - ☐ `x = racine(racine(x)*racine(x));`
7. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
   - ☐ 0.5
   - ☐ 1
   - ☐ 1.5
   - ☐ 0
8. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
   - ☐ `int mccarthy(int 2);`
   - ☐ `x = mccarthy(n);`
   - ☐ `n = mccarthy();`
   - ☐ `n = mccarthy(p, q);`
9. 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.c -o prog.exe`
   - ☐ `gcc -Wall prog.exe -o prog.c`
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 des entrées clavier
    - ☐ l'analyse sémantique
    - ☐ l'édition de liens
11. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ un ordre quelconque
    - ☐ dans lequel ces fonctions sont appelées dans le main
    - ☐ alphabétique
    - ☐ dans lequel vous avez déclaré ces fonction
12. L'écriture 101 en binaire correspond au nombre naturel :
    - ☐ 4
    - ☐ 5
    - ☐ 3
    - ☐ 101
13. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
    - ☐ `n = factorielle(p, q);`
    - ☐ `int factorielle(int 2);`
    - ☐ `printf("%d", factorielle(n));`
    - ☐ `n = factorielle();`
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
    - ☐ le programme affiche "Faux"
    - ☐ la variable x vaut 5 et la variable y vaut 3
    - ☐ la variable y vaut 5

15. 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;`

16. L'écriture 111 en binaire correspond au nombre naturel :

- ☐ 111
- ☐ 8
- ☐ 7
- ☐ 3

17. 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 ?

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

18. 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

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

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

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. 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`
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. 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
4. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

   Alors pour tester l'égalité de a et de b on utilise la condition :
   - ☐ `a = b`
   - ☐ `a{n, x} == b{n, x}`
   - ☐ `(a.n == b.n) && (a.x == b.x)`
   - ☐ `a == b`
5. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
   - ☐ `int mccarthy(int 2);`
   - ☐ `n = mccarthy();`
   - ☐ `x = mccarthy(n);`
   - ☐ `n = mccarthy(p, q);`
6. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   - ☐ `f(a,b)=13, a=8, b=5`
   - ☐ `f(a,b)=8, a=8, b=5`
   - ☐ `f(3,5)=8, a=3, b=5`
   - ☐ `f(a,b)=8, a=3, b=5`
7. Sur unix (ou linux), la commande `mkdir` permet de :
   - ☐ créer un répertoire
   - ☐ changer de répertoire courant
   - ☐ ouvrir un fichier texte
   - ☐ créer un fichier texte
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. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse sémantique
   - ☐ analyse harmonique
   - ☐ analyse lexicale
   - ☐ analyse syntaxique
10. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
    - ☐ `double exposant(double x, int n);`
    - ☐ `void exposant(double x^n);`
    - ☐ `int exposant(double n, int x);`
    - ☐ `exposant(double x, int n, int r);`
11. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

    précède la fonction `main()`, alors on peut écrire en début de `main()` :
    - ☐ `struct toto_s toto;`
    - ☐ `toto_s struct z = {3, 0.5};`
    - ☐ `int toto.n = 3;`
    - ☐ `toto_s n, x;`
    - ☐ `int struct toto_s = {3, -1e10};`

12. Un enregistrement permet de grouper plusieurs valeurs dans :

- ☐ ses champs
- ☐ ses chants
- ☐ ses cases
- ☐ ses blocs

13. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

- ☐ alphabétique
- ☐ dans lequel vous avez déclaré ces fonction
- ☐ un ordre quelconque
- ☐ dans lequel ces fonctions sont appelées dans le main

14. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

- ☐ `n = factorielle(p, q);`
- ☐ `printf("%d", factorielle(n));`
- ☐ `int factorielle(int 2);`
- ☐ `n = factorielle();`

15. Le langage C est un langage

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

16. 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

17. Une variable booléenne est un variable :

- ☐ à laquelle une valeur vient d'être affectée
- ☐ réelle positive
- ☐ jamais nulle
- ☐ qui est vraie ou fausse
- ☐ NaN (not a number, qui n'est pas un nombre)

18. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :

- ☐ `n = pgcd(int p, int q);`
- ☐ `int n = pgcd();`
- ☐ `int pgcd(2);`
- ☐ `n = pgcd(n, 3);`

19. 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 ?

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

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

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. 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
2. Vous utilisez une boucle `while` quand :
   - ☐ vous n'avez pas déclaré de fonction
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous avez déjà fait un `for` dans le même programme principal
3. 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`
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. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ alphabétique
   - ☐ dans lequel ces fonctions sont appelées dans le main
   - ☐ un ordre quelconque
   - ☐ dans lequel vous avez déclaré ces fonction
6. Sur unix (ou linux), la commande `mkdir` permet de :
   - ☐ ouvrir un fichier texte
   - ☐ créer un fichier texte
   - ☐ changer de répertoire courant
   - ☐ créer un répertoire
7. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
   - ☐ `void saisie_utilisateur(int n);`
   - ☐ `int saisie_utilisateur();`
   - ☐ `saisie_utilisateur(scanf(%d));`
   - ☐ `void saisie_utilisateur(char c);`
8. 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`
9. 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
10. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
    - ☐ `printf("Valeur de n ? %d\n", n);`
    - ☐ `printf("Valeur de n ? %g\n", n);`
    - ☐ `scanf("%d", &n);`
    - ☐ un débogueur
11. 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
12. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
    - ☐ `int afficher_date(date_s d);`
    - ☐ `void afficher_date(struct date_s d);`
    - ☐ `struct date_s afficher_date(struct date_s d);`
    - ☐ `void afficher_date(date_s d);`
13. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

    - ☐ il risque d'afficher bonjour à la place de coucou
    - ☐ il comporte une boucle infinie
    - ☐ il n'affiche rien
    - ☐ il ne compile pas
14. 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`

15. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
    - ☐ `int afficher_menu(int char);`
    - ☐ `char afficher_menu(printf("menu"));`
    - ☐ `void afficher_menu();`
    - ☐ `double afficher_menu();`
    - ☐ `int afficher_menu();`
16. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses champs
    - ☐ ses cases
    - ☐ ses chants
    - ☐ ses blocs
17. Si cette erreur apparaît à la compilation :
    `erreur: conflicting types for ‘max’` , que doit-on chercher dans le programme ?
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction déclarée mais non définie
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
18. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

    appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

    L'affichage dans le main est le suivant :
    - ☐ `f(a,b)=8, a=8, b=5`
    - ☐ `f(3,5)=8, a=3, b=5`
    - ☐ `f(a,b)=13, a=8, b=5`
    - ☐ `f(a,b)=8, a=3, b=5`
19. 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
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 = %d`
    - ☐ `j = 4`
    - ☐ `j = 0`
    - ☐ `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. 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

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

3. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

   Alors l'affichage sera :
   - ☐ `A`
   - ☐ `B`
   - ☐ `C`
   - ☐ `b`

4. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses blocs
   - ☐ ses champs
   - ☐ ses chants
   - ☐ ses cases

5. Le code suivant :

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

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

6. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
   - ☐ 0
   - ☐ 1.5
   - ☐ 1
   - ☐ 0.5

7. Avant de faire appel à une fonction il est nécessaire de :
   - ☐ l'avoir déclarée et définie
   - ☐ avoir défini une constante symbolique de la taille de cette fonction
   - ☐ l'avoir déclarée
   - ☐ l'avoir définie

8. 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`

9. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

   précède la fonction `main()`, alors on peut écrire en début de `main()` :
   - ☐ `int struct toto_s = {3, -1e10};`
   - ☐ `toto_s struct z = {3, 0.5};`
   - ☐ `toto_s n, x;`
   - ☐ `struct toto_s toto;`
   - ☐ `int toto.n = 3;`

10. 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

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

12. 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

13. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(a,b)=13, a=8, b=5`

14. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

- ☐ `ABCDEF`
- ☐ `cccccc`
- ☐ `A`
- ☐ `i`

15. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il ne compile pas
- ☐ il comporte une boucle infinie
- ☐ il n'affiche rien
- ☐ il risque d'afficher bonjour à la place de coucou

16. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?

- ☐ `(A == TRUE) && (B == TRUE)`
- ☐ `A && B`
- ☐ `(!A || B)`
- ☐ `!(!A || B) == (A && !B)`

17. 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`

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 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`

20. Vous utilisez une boucle `while` quand :

- ☐ l'incrément de la variable de boucle n'est pas 1
- ☐ vous n'avez pas déclaré de fonction
- ☐ vous avez déjà fait un `for` dans le même programme principal
- ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance

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

affichera :

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

2. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
   - ☐ `char "c";`
   - ☐ `char 'c';`
   - ☐ `char c;`
   - ☐ `int char;`
3. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses chants
   - ☐ ses blocs
   - ☐ ses champs
   - ☐ ses cases
4. 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 tab[] = 5;`
   - ☐ `int toto[taille=5];`
5. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ un ordre quelconque
   - ☐ alphabétique
   - ☐ dans lequel ces fonctions sont appelées dans le main
   - ☐ dans lequel vous avez déclaré ces fonction
6. Vous utilisez une boucle `while` quand :
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous n'avez pas déclaré de fonction
   - ☐ vous avez déjà fait un `for` dans le même programme principal
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
7. Le langage C est un langage
   - ☐ lu, écrit, parlé
   - ☐ compilé
   - ☐ composé
   - ☐ interprété
8. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `(n<=a) && (n<=b)`
- ☐ `(a<n) || (n>b)`
- ☐ `a<=n<=b`
- ☐ `(a<=n) && (n<=b)`

9. 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
10. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=8, b=5`

11. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
    - ☐ `printf("Valeur de n ? %g\n", n);`
    - ☐ `printf("Valeur de n ? %d\n", n);`
    - ☐ `scanf("%d", &n);`
    - ☐ un débogueur
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
- ☐ 0 1 2 3 4
- ☐ 1 2 3 4

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 :

- ☐ 15
- ☐ 0
- ☐ 10
- ☐ 6

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

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

15. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :

- ☐ `int exposant(double n, int x);`
- ☐ `double exposant(double x, int n);`
- ☐ `exposant(double x, int n, int r);`
- ☐ `void exposant(double x^n);`

16. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

Alors l'affichage sera :

- ☐ `A`
- ☐ `b`
- ☐ `B`
- ☐ `C`

17. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

- ☐ `n = factorielle();`
- ☐ `int factorielle(int 2);`
- ☐ `printf("%d", factorielle(n));`
- ☐ `n = factorielle(p, q);`

18. 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

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

20. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :

- ☐ `x = racine(x * x) - racine(x);`
- ☐ `x - 1 = racine(x);`
- ☐ `x = racine(racine(x)*racine(x));`
- ☐ `x = racine(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. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

Alors l'affichage sera :

- ☐ `b`
- ☐ `A`
- ☐ `B`
- ☐ `C`

2. 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

3. Vous utilisez une boucle `while` quand :

- ☐ l'incrément de la variable de boucle n'est pas 1
- ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
- ☐ vous avez déjà fait un `for` dans le même programme principal
- ☐ vous n'avez pas déclaré de fonction

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

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

5. Un enregistrement permet de grouper plusieurs valeurs dans :

- ☐ ses blocs
- ☐ ses chants
- ☐ ses cases
- ☐ ses champs

6. Le code suivant :

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

affichera :

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

7. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :

- ☐ `int afficher_date(date_s d);`
- ☐ `struct date_s afficher_date(struct date_s d);`
- ☐ `void afficher_date(date_s d);`
- ☐ `void afficher_date(struct date_s d);`

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

9. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :

- ☐ `int n = pgcd();`
- ☐ `int pgcd(2);`
- ☐ `n = pgcd(int p, int q);`
- ☐ `n = pgcd(n, 3);`

10. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

- ☐ `int char;`
- ☐ `char 'c';`
- ☐ `char c;`
- ☐ `char "c";`

11. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que `x` est une variable réelle définie et initialisée, il est incorrect d'écrire :

- ☐ `x = racine(racine(x)*racine(x));`
- ☐ `x = racine(x * x) - racine(x);`
- ☐ `x = racine(2/3);`
- ☐ `x - 1 = racine(x);`

12. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

- ☐ `int factorielle(int 2);`
- ☐ `n = factorielle(p, q);`
- ☐ `printf("%d", factorielle(n));`
- ☐ `n = factorielle();`

13. 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

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
- ☐ 0
- ☐ 4
- ☐ 8

15. 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

16. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :

- ☐ `int factorielle(double n);`
- ☐ `int factorielle(int x);`
- ☐ `int factorielle();`
- ☐ `struct int factorielle(int n);`

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. Si cette erreur apparaît à la compilation :
`erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?

- ☐ une fonction déclarée mais non définie
- ☐ une directive préprocesseur `#include` manquante
- ☐ un désaccord entre la déclaration et la définition d'une fonction
- ☐ une fonction appelée avant sa déclaration

19. Le code suivant :

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

affichera :

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

20. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 0
- ☐ 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. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
   - ☐ `x = racine(x * x) - racine(x);`
   - ☐ `x = racine(2/3);`
   - ☐ `x = racine(racine(x)*racine(x));`
   - ☐ `x - 1 = racine(x);`
2. 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
3. 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[5];`
   - ☐ `int toto[taille=5];`
4. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

   Alors l'affichage sera :
   - ☐ `A`
   - ☐ `b`
   - ☐ `B`
   - ☐ `C`
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 :
   - ☐ 16
   - ☐ 8
   - ☐ 4
   - ☐ 0
6. 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
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 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.c -o prog.exe`
   - ☐ `gcc -Wall prog.exe -o prog.c`
9. Sur unix (ou linux), la commande `mkdir` permet de :
   - ☐ changer de répertoire courant
   - ☐ créer un fichier texte
   - ☐ créer un répertoire
   - ☐ ouvrir un fichier texte
10. Une variable booléenne est un variable :
    - ☐ à laquelle une valeur vient d'être affectée
    - ☐ réelle positive
    - ☐ NaN (not a number, qui n'est pas un nombre)
    - ☐ qui est vraie ou fausse
    - ☐ jamais nulle
11. 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
12. Vous utilisez une boucle `while` quand :
    - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
    - ☐ vous n'avez pas déclaré de fonction
    - ☐ l'incrément de la variable de boucle n'est pas 1
    - ☐ vous avez déjà fait un `for` dans le même programme principal
13. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
    - ☐ 7
    - ☐ 8
    - ☐ 3
    - ☐ 111

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 1 2 3 4 5 6 7
- ☐ 0 2 4 6

15. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :

- ☐ `int saisie_utilisateur();`
- ☐ `saisie_utilisateur(scanf(%d));`
- ☐ `void saisie_utilisateur(char c);`
- ☐ `void saisie_utilisateur(int n);`

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
- ☐ 16
- ☐ 4
- ☐ 8

17. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :

- ☐ `int pgcd(int y, int x);`
- ☐ `void pgcd(int x, int y);`
- ☐ `int pgcd(int x, int x);`
- ☐ `int pgcd(int x, y);`

18. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

- ☐ dans lequel vous avez déclaré ces fonction
- ☐ un ordre quelconque
- ☐ alphabétique
- ☐ dans lequel ces fonctions sont appelées dans le main

19. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=3, b=5`

20. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :

- ☐ `n = pgcd(n, 3);`
- ☐ `int n = pgcd();`
- ☐ `n = pgcd(int p, int q);`
- ☐ `int pgcd(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. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

- ☐ `ABCDEF`
- ☐ `A`
- ☐ `cccccc`
- ☐ `i`

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

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

3. 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`

4. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il ne compile pas
- ☐ il n'affiche rien
- ☐ il comporte une boucle infinie

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 qui doit être protégé
- ☐ un document de référence du système
- ☐ un document illisible pour les humains

6. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :

- ☐ 3
- ☐ 111
- ☐ 8
- ☐ 7

7. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

- ☐ un ordre quelconque
- ☐ dans lequel ces fonctions sont appelées dans le main
- ☐ alphabétique
- ☐ dans lequel vous avez déclaré ces fonction

8. Le code suivant :

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

affichera :

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

9. 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

10. 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 ?

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

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

12. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 5
- ☐ 0
- ☐ 7

13. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :

- ☐ `exposant(double x, int n, int r);`
- ☐ `double exposant(double x, int n);`
- ☐ `int exposant(double n, int x);`
- ☐ `void exposant(double x^n);`

14. Le code suivant :

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

affichera :

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

15. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :

- ☐ `int factorielle(int x);`
- ☐ `int factorielle(double n);`
- ☐ `struct int factorielle(int n);`
- ☐ `int factorielle();`

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

17. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

- ☐ `char 'c';`
- ☐ `int char;`
- ☐ `char c;`
- ☐ `char "c";`

18. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `(a<=n) && (n<=b)`
- ☐ `(n<=a) && (n<=b)`
- ☐ `a<=n<=b`
- ☐ `(a<n) || (n>b)`

19. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?

- ☐ `!(!A || B) == (A && !B)`
- ☐ `(A == TRUE) && (B == TRUE)`
- ☐ `(!A || B)`
- ☐ `A && B`

20. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :

- ☐ `void afficher_date(struct date_s d);`
- ☐ `int afficher_date(date_s d);`
- ☐ `void afficher_date(date_s d);`
- ☐ `struct date_s afficher_date(struct date_s 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. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses cases
   - ☐ ses champs
   - ☐ ses blocs
   - ☐ ses chants

2. 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

3. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
   - ☐ `int factorielle(int x);`
   - ☐ `int factorielle();`
   - ☐ `int factorielle(double n);`
   - ☐ `struct int factorielle(int n);`

4. Si cette erreur apparaît à la compilation : `erreur: conflicting types for ‘max’` , que doit-on chercher dans le programme ?
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une fonction déclarée mais non définie
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une directive préprocesseur `#include` manquante

5. Le code suivant :

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

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

6. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
   - ☐ `char "c";`
   - ☐ `char 'c';`
   - ☐ `char c;`
   - ☐ `int char;`

7. 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

8. 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

9. Vous utilisez une boucle `while` quand :
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous n'avez pas déclaré de fonction
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ vous avez déjà fait un `for` dans le même programme principal

10. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
    - ☐ 3
    - ☐ 111
    - ☐ 7
    - ☐ 8

11. 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>`

12. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ dans lequel vous avez déclaré ces fonction
    - ☐ un ordre quelconque
    - ☐ alphabétique
    - ☐ dans lequel ces fonctions sont appelées dans le main

13. Le code suivant :

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

affichera :

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

14. 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`
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=%d et y=%d\n,x,y");`
    - ☐ `printf("x=%x et y=%y\n");`
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 1 0 1 0 1`
- ☐ `1 2 3 1 2`
- ☐ `0 1 2 0 1 2`
- ☐ `0 0 1 1 2 2`

17. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `(a<=n) && (n<=b)`
- ☐ `(n<=a) && (n<=b)`
- ☐ `(a<n) || (n>b)`
- ☐ `a<=n<=b`

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 :

- ☐ 3
- ☐ 20
- ☐ 6
- ☐ 16

19. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
    - ☐ `void afficher_date(struct date_s d);`
    - ☐ `void afficher_date(date_s d);`
    - ☐ `int afficher_date(date_s d);`
    - ☐ `struct date_s afficher_date(struct date_s d);`
20. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
    - ☐ `exposant(double x, int n, int r);`
    - ☐ `double exposant(double x, int n);`
    - ☐ `void exposant(double x^n);`
    - ☐ `int exposant(double n, int x);`

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
   - ☐ 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

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 16
   - ☐ la case mémoire 8 contiendra 0

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

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

5. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
   - ☐ `A && B`
   - ☐ `(A == TRUE) && (B == TRUE)`
   - ☐ `(!A || B)`
   - ☐ `!(!A || B) == (A && !B)`

6. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
   - ☐ `void saisie_utilisateur(char c);`
   - ☐ `void saisie_utilisateur(int n);`
   - ☐ `saisie_utilisateur(scanf(%d));`
   - ☐ `int saisie_utilisateur();`

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

8. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :
   - ☐ 0
   - ☐ 4
   - ☐ 3

9. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   - ☐ `f(a,b)=8, a=8, b=5`
   - ☐ `f(a,b)=13, a=8, b=5`
   - ☐ `f(3,5)=8, a=3, b=5`
   - ☐ `f(a,b)=8, a=3, b=5`

10. 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
    - ☐ 20
    - ☐ 3

11. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

- ☐ `ABCDEF`
- ☐ `cccccc`
- ☐ `i`
- ☐ `A`

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

13. 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

14. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

Alors l'affichage sera :

- ☐ `b`
- ☐ `A`
- ☐ `B`
- ☐ `C`

15. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il n'affiche rien
- ☐ il ne compile pas
- ☐ il comporte une boucle infinie
- ☐ il risque d'afficher bonjour à la place de coucou

16. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 7
- ☐ 5
- ☐ 0

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 tab[] = 5;`
- ☐ `int toto[5];`

18. Vous utilisez une boucle `while` quand :

- ☐ l'incrément de la variable de boucle n'est pas 1
- ☐ vous avez déjà fait un `for` dans le même programme principal
- ☐ vous n'avez pas déclaré de fonction
- ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance

19. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :

- ☐ 111
- ☐ 7
- ☐ 8
- ☐ 3

20. Si cette erreur apparaît à la compilation :
`erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?

- ☐ une fonction appelée avant sa déclaration
- ☐ un désaccord entre la déclaration et la définition d'une fonction
- ☐ une directive préprocesseur `#include` manquante
- ☐ une fonction déclarée mais non définie

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 :

☐ 0
☐ 6
☐ 1
☐ 42

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 = 5`
☐ `j = %d`
☐ `j = 0`

3. Un enregistrement permet de grouper plusieurs valeurs dans :

☐ ses chants
☐ ses champs
☐ ses cases
☐ ses blocs

4. 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)`

5. 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

6. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :

☐ 111
☐ 3
☐ 8
☐ 7

7. 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

8. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :

☐ `void exposant(double x^n);`
☐ `double exposant(double x, int n);`
☐ `exposant(double x, int n, int r);`
☐ `int exposant(double n, int x);`

9. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

☐ `int char;`
☐ `char "c";`
☐ `char 'c';`
☐ `char c;`

10. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

☐ 4
☐ 5
☐ 1
☐ 0

11. Le code suivant :

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

affichera :

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

12. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :

- ☐ `x = mccarthy(n);`
- ☐ `int mccarthy(int 2);`
- ☐ `n = mccarthy(p, q);`
- ☐ `n = mccarthy();`

13. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

- ☐ dans lequel vous avez déclaré ces fonction
- ☐ alphabétique
- ☐ un ordre quelconque
- ☐ dans lequel ces fonctions sont appelées dans le main

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

15. Avant de faire appel à une fonction il est nécessaire de :

- ☐ l'avoir définie
- ☐ l'avoir déclarée
- ☐ l'avoir déclarée et définie
- ☐ avoir défini une constante symbolique de la taille de cette fonction

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

17. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il n'affiche rien
- ☐ il comporte une boucle infinie
- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il ne compile pas

18. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `a<=n<=b`
- ☐ `(a<=n) && (n<=b)`
- ☐ `(n<=a) && (n<=b)`
- ☐ `(a<n) || (n>b)`

19. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?

- ☐ `(A == TRUE) && (B == TRUE)`
- ☐ `(!A || B)`
- ☐ `!(!A || B) == (A && !B)`
- ☐ `A && B`

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 2 4 6
- ☐ 0 1 2 3 4 5 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. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

☐ 0
☐ 3
☐ 4

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 = %d`
☐ `j = 5`

3. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

☐ `a<=n<=b`
☐ `(n<=a) && (n<=b)`
☐ `(a<n) || (n>b)`
☐ `(a<=n) && (n<=b)`

4. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

☐ il comporte une boucle infinie
☐ il n'affiche rien
☐ il ne compile pas
☐ il risque d'afficher bonjour à la place de coucou

5. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :

☐ `int n = pgcd();`
☐ `n = pgcd(int p, int q);`
☐ `n = pgcd(n, 3);`
☐ `int pgcd(2);`

6. 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

7. Le langage C est un langage

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

8. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que `x` est une variable réelle définie et initialisée, il est incorrect d'écrire :

☐ `x - 1 = racine(x);`
☐ `x = racine(racine(x)*racine(x));`
☐ `x = racine(2/3);`
☐ `x = racine(x * x) - racine(x);`

9. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

☐ `f(a,b)=13, a=8, b=5`
☐ `f(a,b)=8, a=8, b=5`
☐ `f(3,5)=8, a=3, b=5`
☐ `f(a,b)=8, a=3, b=5`

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 :

☐ 15
☐ 10
☐ 0
☐ 6

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 :

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

12. 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

13. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 5
- ☐ 1
- ☐ 4
- ☐ 0

14. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :

- ☐ 1
- ☐ 0
- ☐ 0.5
- ☐ 1.5

15. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 7
- ☐ 5
- ☐ 0

16. 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

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

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

19. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :

- ☐ `int afficher_menu();`
- ☐ `double afficher_menu();`
- ☐ `char afficher_menu(printf("menu"));`
- ☐ `void afficher_menu();`
- ☐ `int afficher_menu(int char);`

20. Un enregistrement permet de grouper plusieurs valeurs dans :

- ☐ ses blocs
- ☐ ses cases
- ☐ ses chants
- ☐ ses champs

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 a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `a = b`
- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a{n, x} == b{n, x}`
- ☐ `a == b`

2. L'écriture 101 en binaire correspond au nombre naturel :

- ☐ 5
- ☐ 3
- ☐ 4
- ☐ 101

3. Le type des réels en C est :

- ☐ `real`
- ☐ `double`
- ☐ `char`
- ☐ `int`

4. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :

- ☐ 0
- ☐ 1
- ☐ 1.5
- ☐ 0.5

5. Le code suivant :

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

affichera :

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

6. Un fichier source est :

- ☐ un fichier texte qui sera traduit en instructions processeur
- ☐ 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

7. 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

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
- ☐ tous ensemble
- ☐ tour à tour, un petit peu à chaque fois

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 :

- ☐ 6
- ☐ 10
- ☐ 0
- ☐ 15

10. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=8, b=5`

11. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 7
- ☐ 0
- ☐ 5

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
    - ☐ 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
13. Le langage C est un langage
    - ☐ interprété
    - ☐ compilé
    - ☐ composé
    - ☐ lu, écrit, parlé
14. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :

- ☐ `toto_s struct z = {3, 0.5};`
- ☐ `int struct toto_s = {3, -1e10};`
- ☐ `struct toto_s toto;`
- ☐ `int toto.n = 3;`
- ☐ `toto_s n, x;`

15. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
    - ☐ `int mccarthy(int 2);`
    - ☐ `n = mccarthy(p, q);`
    - ☐ `n = mccarthy();`
    - ☐ `x = mccarthy(n);`
16. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
    - ☐ `char c;`
    - ☐ `int char;`
    - ☐ `char "c";`
    - ☐ `char 'c';`
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
    - ☐ les fichiers du disque
    - ☐ en temps d'accès
18. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
    - ☐ `(!A || B)`
    - ☐ `!(!A || B) == (A && !B)`
    - ☐ `A && B`
    - ☐ `(A == TRUE) && (B == TRUE)`
19. Le code suivant :

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

affichera :

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

- ☐ 20
- ☐ 6
- ☐ 3
- ☐ 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. Si cet avertissement apparaît à la compilation :
   `warning: implicit declaration of function ‘max’`
   , que doit-on chercher dans le programme ?
   - ☐ une fonction déclarée mais non définie
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une directive préprocesseur `#include` manquante

2. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :
   - ☐ 4
   - ☐ 3
   - ☐ 0

3. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
   - ☐ `printf("Valeur de n ? %g\n", n);`
   - ☐ `printf("Valeur de n ? %d\n", n);`
   - ☐ un débogueur
   - ☐ `scanf("%d", &n);`

4. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
   - ☐ `int carre(2);`
   - ☐ `int n = carre();`
   - ☐ `n = carre(int n);`
   - ☐ `n = carre(n);`

5. Le code suivant :

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

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

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. 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`

8. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

   Alors pour tester l'égalité de a et de b on utilise la condition :
   - ☐ `a{n, x} == b{n, x}`
   - ☐ `a = b`
   - ☐ `(a.n == b.n) && (a.x == b.x)`
   - ☐ `a == b`

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

10. L'écriture $\underline{101}$ en binaire correspond au nombre naturel :
    - ☐ 101
    - ☐ 5
    - ☐ 4
    - ☐ 3

11. Si cette erreur apparaît à la compilation :
    `erreur: conflicting types for ‘max’` , que doit-on chercher dans le programme ?
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une fonction déclarée mais non définie
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ un désaccord entre la déclaration et la définition d'une fonction

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
- ☐ 16
- ☐ 4

13. 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 de référence du système
    - ☐ un document illisible pour les humains
    - ☐ un document qui doit être protégé
14. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
    - ☐ `exposant(double x, int n, int r);`
    - ☐ `int exposant(double n, int x);`
    - ☐ `void exposant(double x^n);`
    - ☐ `double exposant(double x, int n);`
15. 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`
16. Lorsqu'un programme utilise `printf` ou `scanf` il faut qu'il contienne l'instruction préprocesseur :
    - ☐ `#appart <stdlib.h>`
    - ☐ `#include <studio.h>`
    - ☐ `#include <studlib.h>`
    - ☐ `#include <stdio.h>`
17. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il ne compile pas
- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il comporte une boucle infinie
- ☐ il n'affiche rien

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=%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);`
19. 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)
20. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 7
- ☐ 0
- ☐ 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. 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

2. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 7
- ☐ 0
- ☐ 5

3. 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
- ☐ 1
- ☐ 42

4. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=3, b=5`

5. Si cette erreur apparaît à la compilation :
`erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?

- ☐ un désaccord entre la déclaration et la définition d'une fonction
- ☐ une fonction déclarée mais non définie
- ☐ une fonction appelée avant sa déclaration
- ☐ une directive préprocesseur `#include` manquante

6. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :

- ☐ 3
- ☐ 111
- ☐ 8
- ☐ 7

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 2
- ☐ 8 6 4 2 0

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=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)`

9. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :

- ☐ `n = pgcd(int p, int q);`
- ☐ `n = pgcd(n, 3);`
- ☐ `int pgcd(2);`
- ☐ `int n = pgcd();`

10. Le langage C est un langage

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

11. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :

- ☐ `double exposant(double x, int n);`
- ☐ `int exposant(double n, int x);`
- ☐ `exposant(double x, int n, int r);`
- ☐ `void exposant(double x^n);`

12. Le code suivant :

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

affichera :

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

13. 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
14. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
    - ☐ `n = mccarthy();`
    - ☐ `n = mccarthy(p, q);`
    - ☐ `x = mccarthy(n);`
    - ☐ `int mccarthy(int 2);`
15. Si cette erreur apparaît à la compilation :
    `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
    - ☐ un point-virgule manquant
    - ☐ une accolade manquante
    - ☐ une accolade en trop
    - ☐ un point-virgule en trop
16. Une variable booléenne est un variable :
    - ☐ jamais nulle
    - ☐ à laquelle une valeur vient d'être affectée
    - ☐ qui est vraie ou fausse
    - ☐ réelle positive
    - ☐ NaN (not a number, qui n'est pas un nombre)
17. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
    - ☐ `!(!A || B) == (A && !B)`
    - ☐ `(A == TRUE) && (B == TRUE)`
    - ☐ `(!A || B)`
    - ☐ `A && B`
18. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
    - ☐ `int char;`
    - ☐ `char c;`
    - ☐ `char "c";`
    - ☐ `char 'c';`
19. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
    - ☐ `int n = carre();`
    - ☐ `n = carre(int n);`
    - ☐ `int carre(2);`
    - ☐ `n = carre(n);`
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
- ☐ 20
- ☐ 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. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
   - ☐ `(!A || B)`
   - ☐ `(A == TRUE) && (B == TRUE)`
   - ☐ `!(!A || B) == (A && !B)`
   - ☐ `A && B`
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 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
4. 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>`
5. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
   - ☐ `int factorielle(int 2);`
   - ☐ `printf("%d", factorielle(n));`
   - ☐ `n = factorielle();`
   - ☐ `n = factorielle(p, q);`
6. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
   - ☐ `int n = carre();`
   - ☐ `n = carre(int n);`
   - ☐ `int carre(2);`
   - ☐ `n = carre(n);`
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
   - ☐ compiler un programme
   - ☐ voir des clips musicaux
8. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il risque d'afficher bonjour à la place de coucou
   - ☐ il ne compile pas
   - ☐ il n'affiche rien
   - ☐ il comporte une boucle infinie
9. Si cette erreur apparaît à la compilation : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
   - ☐ une fonction appelée avant sa déclaration
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une fonction déclarée mais non définie
10. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

    Alors pour tester l'égalité de a et de b on utilise la condition :
    - ☐ `(a.n == b.n) && (a.x == b.x)`
    - ☐ `a = b`
    - ☐ `a{n, x} == b{n, x}`
    - ☐ `a == b`
11. 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`
12. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ alphabétique
    - ☐ un ordre quelconque
    - ☐ dans lequel vous avez déclaré ces fonction
    - ☐ dans lequel ces fonctions sont appelées dans le main

13. Un fichier source est :
    - ☐ un document qui doit être protégé
    - ☐ 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 illisible pour les humains
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. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
    - ☐ `int afficher_date(date_s d);`
    - ☐ `void afficher_date(date_s d);`
    - ☐ `void afficher_date(struct date_s d);`
    - ☐ `struct date_s afficher_date(struct date_s 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
    - ☐ 1 2 3 4
    - ☐ 4 3 2 1 0
    - ☐ 0 1 2 3 4
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=0;i<5;i=i+1)`
    - ☐ `for(i=1;i<5;i=i+1)`
18. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
    - ☐ `char 'c';`
    - ☐ `int char;`
    - ☐ `char c;`
    - ☐ `char "c";`
19. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses champs
    - ☐ ses chants
    - ☐ ses blocs
    - ☐ ses cases
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. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ dans lequel ces fonctions sont appelées dans le main
   - ☐ alphabétique
   - ☐ un ordre quelconque
   - ☐ dans lequel vous avez déclaré ces fonction

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

3. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il comporte une boucle infinie
   - ☐ il ne compile pas
   - ☐ il risque d'afficher bonjour à la place de coucou
   - ☐ il n'affiche rien

4. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
   - ☐ `x = racine(2/3);`
   - ☐ `x = racine(x * x) - racine(x);`
   - ☐ `x - 1 = racine(x);`
   - ☐ `x = racine(racine(x)*racine(x));`

5. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
   - ☐ `void saisie_utilisateur(int n);`
   - ☐ `int saisie_utilisateur();`
   - ☐ `void saisie_utilisateur(char c);`
   - ☐ `saisie_utilisateur(scanf(%d));`

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
   - ☐ qu'il faut lancer un déboggueur
   - ☐ que l'on veut voir tous les avertissements

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. Avant de faire appel à une fonction il est nécessaire de :
   - ☐ l'avoir déclarée
   - ☐ l'avoir définie
   - ☐ avoir défini une constante symbolique de la taille de cette fonction
   - ☐ l'avoir déclarée et définie

9. 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`

10. Le type des réels en C est :
    - ☐ `char`
    - ☐ `int`
    - ☐ `real`
    - ☐ `double`

11. Le code suivant :

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

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

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

13. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `(a<=n) && (n<=b)`
- ☐ `(n<=a) && (n<=b)`
- ☐ `a<=n<=b`
- ☐ `(a<n) || (n>b)`

14. 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>`

15. Un enregistrement permet de grouper plusieurs valeurs dans :

- ☐ ses blocs
- ☐ ses champs
- ☐ ses cases
- ☐ ses chants

16. 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

17. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :

- ☐ `struct date_s afficher_date(struct date_s d);`
- ☐ `int afficher_date(date_s d);`
- ☐ `void afficher_date(date_s d);`
- ☐ `void afficher_date(struct date_s d);`

18. Si cette erreur apparaît à la compilation :
`erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?

- ☐ une fonction déclarée mais non définie
- ☐ une directive préprocesseur `#include` manquante
- ☐ un désaccord entre la déclaration et la définition d'une fonction
- ☐ une fonction appelée avant sa déclaration

19. 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");`

20. 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

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 :
   - ☐ ouvrir un fichier texte
   - ☐ créer un fichier texte
   - ☐ changer de répertoire courant
   - ☐ créer un répertoire
2. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ dans lequel ces fonctions sont appelées dans le main
   - ☐ alphabétique
   - ☐ dans lequel vous avez déclaré ces fonction
   - ☐ un ordre quelconque
3. 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
4. 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`
5. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
   - ☐ `printf("Valeur de n ? %d\n", n);`
   - ☐ `scanf("%d", &n);`
   - ☐ `printf("Valeur de n ? %g\n", n);`
   - ☐ un débogueur
6. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
   - ☐ 7
   - ☐ 111
   - ☐ 8
   - ☐ 3
7. 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
8. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :
   - ☐ `int pgcd(2);`
   - ☐ `int n = pgcd();`
   - ☐ `n = pgcd(n, 3);`
   - ☐ `n = pgcd(int p, int q);`
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 manquant
   - ☐ un point-virgule en trop
   - ☐ une accolade en trop
10. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
    - ☐ `n = mccarthy();`
    - ☐ `int mccarthy(int 2);`
    - ☐ `n = mccarthy(p, q);`
    - ☐ `x = mccarthy(n);`
11. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

    Alors l'affichage sera :
    - ☐ `b`
    - ☐ `A`
    - ☐ `C`
    - ☐ `B`
12. Le code suivant :

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

    affichera :
    - ☐ `Majeur`
    - ☐ `Mineur`
      `Majeur`
    - ☐ `Mineur`
    - ☐ rien
13. 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

14. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
    - ☐ `saisie_utilisateur(scanf(%d));`
    - ☐ `void saisie_utilisateur(char c);`
    - ☐ `void saisie_utilisateur(int n);`
    - ☐ `int saisie_utilisateur();`
15. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
    - ☐ `!(!A || B) == (A && !B)`
    - ☐ `A && B`
    - ☐ `(!A || B)`
    - ☐ `(A == TRUE) && (B == TRUE)`
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=%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);`
17. Avant de faire appel à une fonction il est nécessaire de :
    - ☐ l'avoir déclarée et définie
    - ☐ avoir défini une constante symbolique de la taille de cette fonction
    - ☐ l'avoir définie
    - ☐ l'avoir déclarée
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 = 5`
    - ☐ `j = 4`
    - ☐ `j = 0`
    - ☐ `j = %d`
19. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
    - ☐ `int afficher_menu(int char);`
    - ☐ `void afficher_menu();`
    - ☐ `char afficher_menu(printf("menu"));`
    - ☐ `int afficher_menu();`
    - ☐ `double afficher_menu();`
20. 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

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 une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
   - ☐ `exposant(double x, int n, int r);`
   - ☐ `int exposant(double n, int x);`
   - ☐ `void exposant(double x^n);`
   - ☐ `double exposant(double x, int n);`
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
   - ☐ de doubler la mémoire disponible
   - ☐ d'entretenir l'illusion que les processus tournent en parallèle
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
   - ☐ 4 3 2 1 0
   - ☐ 4 3 2 1
   - ☐ 1 2 3 4
4. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
   - ☐ `int afficher_menu();`
   - ☐ `void afficher_menu();`
   - ☐ `int afficher_menu(int char);`
   - ☐ `char afficher_menu(printf("menu"));`
   - ☐ `double afficher_menu();`
5. Vous utilisez une boucle `while` quand :
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ vous n'avez pas déclaré de fonction
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous avez déjà fait un `for` dans le même programme principal
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
   - ☐ 8
   - ☐ 0
   - ☐ 4
7. Le langage C est un langage
   - ☐ lu, écrit, parlé
   - ☐ compilé
   - ☐ interprété
   - ☐ composé
8. Une variable booléenne est un variable :
   - ☐ qui est vraie ou fausse
   - ☐ réelle positive
   - ☐ à laquelle une valeur vient d'être affectée
   - ☐ jamais nulle
   - ☐ NaN (not a number, qui n'est pas un nombre)
9. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   - ☐ `f(a,b)=8, a=3, b=5`
   - ☐ `f(a,b)=8, a=8, b=5`
   - ☐ `f(3,5)=8, a=3, b=5`
   - ☐ `f(a,b)=13, a=8, b=5`
10. Avant de faire appel à une fonction il est nécessaire de :
    - ☐ l'avoir déclarée et définie
    - ☐ l'avoir déclarée
    - ☐ l'avoir définie
    - ☐ avoir défini une constante symbolique de la taille de cette fonction
11. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
    - ☐ `x = racine(x * x) - racine(x);`
    - ☐ `x - 1 = racine(x);`
    - ☐ `x = racine(2/3);`
    - ☐ `x = racine(racine(x)*racine(x));`
12. Un fichier source est :
    - ☐ un document qui doit être protégé
    - ☐ un document illisible pour les humains
    - ☐ un fichier texte qui sera traduit en instructions processeur
    - ☐ un document de référence du système
    - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur

13. 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

14. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
    - ☐ analyse syntaxique
    - ☐ analyse lexicale
    - ☐ analyse sémantique
    - ☐ analyse harmonique
15. 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
16. Le type des réels en C est :
    - ☐ `real`
    - ☐ `int`
    - ☐ `double`
    - ☐ `char`
17. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :

- ☐ `struct toto_s toto;`
- ☐ `int toto.n = 3;`
- ☐ `toto_s struct z = {3, 0.5};`
- ☐ `toto_s n, x;`
- ☐ `int struct toto_s = {3, -1e10};`

18. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
    - ☐ `(A == TRUE) && (B == TRUE)`
    - ☐ `A && B`
    - ☐ `(!A || B)`
    - ☐ `!(!A || B) == (A && !B)`
19. 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
20. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `a = b`
- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a == b`
- ☐ `a{n, x} == b{n, x}`

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. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ dans lequel ces fonctions sont appelées dans le main
   - ☐ dans lequel vous avez déclaré ces fonction
   - ☐ alphabétique
   - ☐ un ordre quelconque
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 lancer un déboggueur
   - ☐ qu'il faut indenter le fichier source
3. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

   Alors l'affichage sera :
   - ☐ `ABCDEF`
   - ☐ `i`
   - ☐ `cccccc`
   - ☐ `A`
4. 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
5. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :
   - ☐ 3
   - ☐ 0
   - ☐ 4
6. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

   Alors l'affichage sera :
   - ☐ `A`
   - ☐ `b`
   - ☐ `C`
   - ☐ `B`
7. 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
8. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
   - ☐ 3
   - ☐ 7
   - ☐ 8
   - ☐ 111
9. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
   - ☐ `n = mccarthy();`
   - ☐ `x = mccarthy(n);`
   - ☐ `int mccarthy(int 2);`
   - ☐ `n = mccarthy(p, q);`
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 `x = 5`
    - ☐ le terminal affiche "Faux"
    - ☐ le terminal affiche `5`
11. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

    appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

    L'affichage dans le main est le suivant :
    - ☐ `f(a,b)=8, a=8, b=5`
    - ☐ `f(a,b)=8, a=3, b=5`
    - ☐ `f(3,5)=8, a=3, b=5`
    - ☐ `f(a,b)=13, a=8, b=5`

12. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `a = b`
- ☐ `a == b`
- ☐ `a{n, x} == b{n, x}`
- ☐ `(a.n == b.n) && (a.x == b.x)`

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 faute de frappe dans un appel de fonction
    - ☐ un caractère interdit en C
    - ☐ une variable non déclarée
14. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
    - ☐ analyse sémantique
    - ☐ analyse harmonique
    - ☐ analyse lexicale
    - ☐ analyse syntaxique
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 ?

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

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. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il comporte une boucle infinie
- ☐ il n'affiche rien
- ☐ il ne compile pas

18. 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

19. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :

- ☐ `int carre(2);`
- ☐ `n = carre(int n);`
- ☐ `n = carre(n);`
- ☐ `int n = carre();`

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. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

   Quelle est la condition `cond` :
   - ☐ `(n<=a) && (n<=b)`
   - ☐ `(a<n) || (n>b)`
   - ☐ `a<=n<=b`
   - ☐ `(a<=n) && (n<=b)`

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

3. Vous utilisez une boucle `while` quand :
   - ☐ vous n'avez pas déclaré de fonction
   - ☐ vous avez déjà fait un `for` dans le même programme principal
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance

4. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
   - ☐ `struct date_s afficher_date(struct date_s d);`
   - ☐ `int afficher_date(date_s d);`
   - ☐ `void afficher_date(date_s d);`
   - ☐ `void afficher_date(struct date_s d);`

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=%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");`

6. Une variable booléenne est un variable :
   - ☐ jamais nulle
   - ☐ NaN (not a number, qui n'est pas un nombre)
   - ☐ qui est vraie ou fausse
   - ☐ à laquelle une valeur vient d'être affectée
   - ☐ réelle positive

7. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
   - ☐ `int factorielle(int x);`
   - ☐ `int factorielle();`
   - ☐ `int factorielle(double n);`
   - ☐ `struct int factorielle(int n);`

8. 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`

9. 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
   - ☐ 0 1 2 3 4
   - ☐ 4 3 2 1

10. 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

11. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
    - ☐ `int saisie_utilisateur();`
    - ☐ `saisie_utilisateur(scanf(%d));`
    - ☐ `void saisie_utilisateur(char c);`
    - ☐ `void saisie_utilisateur(int n);`

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

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. 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)`
15. Un programme en langage C doit comporter une et une seule définition de la fonction :
    - ☐ main
    - ☐ init
    - ☐ include
    - ☐ begin
16. L'écriture 101 en binaire correspond au nombre naturel :
    - ☐ 101
    - ☐ 3
    - ☐ 4
    - ☐ 5
17. 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 qui doit être protégé
    - ☐ un document de référence du système
    - ☐ un document illisible pour les humains
18. L'écriture 111 en binaire correspond au nombre naturel :
    - ☐ 3
    - ☐ 7
    - ☐ 8
    - ☐ 111
19. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
    - ☐ `char c;`
    - ☐ `char "c";`
    - ☐ `char 'c';`
    - ☐ `int char;`
20. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 0
- ☐ 1
- ☐ 5
- ☐ 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. 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 qui doit être protégé
   - ☐ un document illisible pour les humains
   - ☐ un document de référence du système

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

3. Une variable booléenne est un variable :
   - ☐ qui est vraie ou fausse
   - ☐ à laquelle une valeur vient d'être affectée
   - ☐ NaN (not a number, qui n'est pas un nombre)
   - ☐ réelle positive
   - ☐ jamais nulle

4. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   - ☐ `f(3,5)=8, a=3, b=5`
   - ☐ `f(a,b)=13, a=8, b=5`
   - ☐ `f(a,b)=8, a=8, b=5`
   - ☐ `f(a,b)=8, a=3, b=5`

5. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
   - ☐ `struct date_s afficher_date(struct date_s d);`
   - ☐ `int afficher_date(date_s d);`
   - ☐ `void afficher_date(struct date_s d);`
   - ☐ `void afficher_date(date_s d);`

6. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
   - ☐ 3
   - ☐ 7
   - ☐ 8
   - ☐ 111

7. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
   - ☐ `char c;`
   - ☐ `int char;`
   - ☐ `char 'c';`
   - ☐ `char "c";`

8. L'écriture $\underline{101}$ en binaire correspond au nombre naturel :
   - ☐ 4
   - ☐ 3
   - ☐ 101
   - ☐ 5

9. 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`

10. Vous utilisez une boucle `while` quand :
    - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
    - ☐ vous avez déjà fait un `for` dans le même programme principal
    - ☐ l'incrément de la variable de boucle n'est pas 1
    - ☐ vous n'avez pas déclaré de fonction

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
    - ☐ un caractère interdit en C
    - ☐ une variable non déclarée
    - ☐ une directive préprocesseur `#include` manquante

12. Sur un ordinateur avec un seul processeur, habituellement les processus sont exécutés :
    - ☐ tous ensemble
    - ☐ 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

13. 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

14. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ créer un fichier texte
    - ☐ créer un répertoire
    - ☐ changer de répertoire courant
    - ☐ ouvrir un fichier texte
15. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

    Alors l'expression `f(0)` prendra la valeur :
    - ☐ 3
    - ☐ 4
    - ☐ 0
16. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :
    - ☐ `int n = pgcd();`
    - ☐ `n = pgcd(int p, int q);`
    - ☐ `int pgcd(2);`
    - ☐ `n = pgcd(n, 3);`
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 1 0 1 0 1`
    - ☐ `0 0 1 1 2 2`
    - ☐ `0 1 2 0 1 2`
18. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
    - ☐ `n = carre(int n);`
    - ☐ `int carre(2);`
    - ☐ `n = carre(n);`
    - ☐ `int n = carre();`
19. Le type des réels en C est :
    - ☐ `real`
    - ☐ `int`
    - ☐ `double`
    - ☐ `char`
20. 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;`
    - ☐ `int toto[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. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
   - ☐ `double afficher_menu();`
   - ☐ `char afficher_menu(printf("menu"));`
   - ☐ `int afficher_menu(int char);`
   - ☐ `void afficher_menu();`
   - ☐ `int afficher_menu();`
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. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :
   - ☐ `n = pgcd(int p, int q);`
   - ☐ `n = pgcd(n, 3);`
   - ☐ `int pgcd(2);`
   - ☐ `int n = pgcd();`
4. Le code suivant :

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

   affichera :
   - ☐ `Mineur`
   - ☐ `Mineur`
`Majeur`
   - ☐ rien
   - ☐ `Majeur`
5. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :
   - ☐ 4
   - ☐ 3
   - ☐ 0
6. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse harmonique
   - ☐ analyse lexicale
   - ☐ analyse sémantique
   - ☐ analyse syntaxique
7. 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
8. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
   - ☐ `char c;`
   - ☐ `int char;`
   - ☐ `char "c";`
   - ☐ `char 'c';`
9. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

   Alors pour tester l'égalité de a et de b on utilise la condition :
   - ☐ `(a.n == b.n) && (a.x == b.x)`
   - ☐ `a{n, x} == b{n, x}`
   - ☐ `a == b`
   - ☐ `a = b`
10. 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
    - ☐ une accolade manquante
    - ☐ un point-virgule en trop
11. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
    - ☐ 8
    - ☐ 7
    - ☐ 111
    - ☐ 3
12. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

    Alors l'affichage sera :
    - ☐ `ABCDEF`
    - ☐ `cccccc`
    - ☐ `i`
    - ☐ `A`

13. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?

- ☐ une directive préprocesseur `#include` manquante
- ☐ une fonction déclarée mais non définie
- ☐ une fonction appelée avant sa déclaration
- ☐ un désaccord entre la déclaration et la définition d'une fonction

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

15. 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>`

16. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 4
- ☐ 1
- ☐ 0
- ☐ 5

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 3 2 1
- ☐ 4 3 2 1 0
- ☐ 0 1 2 3 4

18. Un enregistrement permet de grouper plusieurs valeurs dans :

- ☐ ses chants
- ☐ ses champs
- ☐ ses blocs
- ☐ ses cases

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
- ☐ 0 1 2 3
- ☐ 4 3 2 1

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  }
```

- ☐ 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"

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 la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

☐ 4
☐ 0
☐ 5
☐ 1

2. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

☐ 5
☐ 0
☐ 7

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. 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

5. 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

6. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :

☐ un débogueur
☐ `scanf("%d", &n);`
☐ `printf("Valeur de n ? %d\n", n);`
☐ `printf("Valeur de n ? %g\n", n);`

7. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

☐ `printf("%d", factorielle(n));`
☐ `n = factorielle(p, q);`
☐ `int factorielle(int 2);`
☐ `n = factorielle();`

8. Si `x` est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :

☐ 0.5
☐ 1
☐ 0
☐ 1.5

9. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :

☐ `void afficher_date(struct date_s d);`
☐ `struct date_s afficher_date(struct date_s d);`
☐ `int afficher_date(date_s d);`
☐ `void afficher_date(date_s d);`

10. Vous utilisez une boucle `while` quand :

☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
☐ l'incrément de la variable de boucle n'est pas 1
☐ vous avez déjà fait un `for` dans le même programme principal
☐ vous n'avez pas déclaré de fonction

11. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

☐ `char "c";`
☐ `char 'c';`
☐ `int char;`
☐ `char c;`

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
☐ 20
☐ 16
☐ 3

13. 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

14. Si cet avertissement apparaît à la compilation :
    `warning: implicit declaration of function ‘max’`
    , que doit-on chercher dans le programme ?
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une fonction déclarée mais non définie
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ un désaccord entre la déclaration et la définition d'une fonction

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 5 et la variable y vaut 3
    - ☐ la variable x vaut 3
    - ☐ la variable y vaut 5

16. 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

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

18. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses champs
    - ☐ ses chants
    - ☐ ses cases
    - ☐ ses blocs

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

20. Un fichier source est :
    - ☐ 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
    - ☐ un document illisible pour les humains

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

2. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :

- ☐ `n = pgcd(n, 3);`
- ☐ `int pgcd(2);`
- ☐ `n = pgcd(int p, int q);`
- ☐ `int n = pgcd();`

3. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 5
- ☐ 0
- ☐ 7

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

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

5. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :

- ☐ `int toto.n = 3;`
- ☐ `toto_s struct z = {3, 0.5};`
- ☐ `struct toto_s toto;`
- ☐ `toto_s n, x;`
- ☐ `int struct toto_s = {3, -1e10};`

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. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

- ☐ un ordre quelconque
- ☐ dans lequel vous avez déclaré ces fonction
- ☐ alphabétique
- ☐ dans lequel ces fonctions sont appelées dans le main

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. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 0
- ☐ 4
- ☐ 1
- ☐ 5

10. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

- ☐ `n = factorielle();`
- ☐ `printf("%d", factorielle(n));`
- ☐ `n = factorielle(p, q);`
- ☐ `int factorielle(int 2);`

11. 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

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.c -o -Wall prog.exe`
    - ☐ `gcc prog.exe -Wall -o prog.c`
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
    - ☐ 4 3 2 1
    - ☐ 4 3 2 1 0
    - ☐ 0 1 2 3
14. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
    - ☐ `n = carre(n);`
    - ☐ `int carre(2);`
    - ☐ `int n = carre();`
    - ☐ `n = carre(int n);`
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 3 0 1 2`
    - ☐ `0 1 2 0 1 2 3`
    - ☐ `0 1 2 0 1 2`
    - ☐ `0 0 1 1 2 2 3`
16. Si cette erreur apparaît à la compilation : `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
    - ☐ un point-virgule manquant
    - ☐ une accolade manquante
    - ☐ une accolade en trop
    - ☐ un point-virgule en trop
17. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
    - ☐ `scanf("%d", &n);`
    - ☐ `printf("Valeur de n ? %d\n", n);`
    - ☐ `printf("Valeur de n ? %g\n", n);`
    - ☐ un débogueur
18. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
    - ☐ `struct date_s afficher_date(struct date_s d);`
    - ☐ `void afficher_date(struct date_s d);`
    - ☐ `void afficher_date(date_s d);`
    - ☐ `int afficher_date(date_s d);`
19. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

    Alors pour tester l'égalité de a et de b on utilise la condition :
    - ☐ `(a.n == b.n) && (a.x == b.x)`
    - ☐ `a = b`
    - ☐ `a{n, x} == b{n, x}`
    - ☐ `a == b`
20. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses cases
    - ☐ ses champs
    - ☐ ses blocs
    - ☐ ses chants

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. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

☐ `i`
☐ `A`
☐ `ABCDEF`
☐ `cccccc`

2. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

☐ `int char;`
☐ `char c;`
☐ `char "c";`
☐ `char 'c';`

3. L'écriture 101 en binaire correspond au nombre naturel :

☐ 5
☐ 3
☐ 101
☐ 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=%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. 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;`

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 :

☐ 8
☐ 4
☐ 16
☐ 0

7. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

☐ il comporte une boucle infinie
☐ il n'affiche rien
☐ il risque d'afficher bonjour à la place de coucou
☐ il ne compile pas

8. 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

9. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

☐ 0
☐ 4
☐ 3

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 :

☐ 6
☐ 10
☐ 15
☐ 0

11. L'écriture 111 en binaire correspond au nombre naturel :

☐ 7
☐ 8
☐ 3
☐ 111

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
☐ 0 1 2 3 4 5 6
☐ 0 2 4 6 8

13. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `a == b`
- ☐ `a{n, x} == b{n, x}`
- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a = b`

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

15. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 0
- ☐ 5
- ☐ 7

16. 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>`

17. Avant de faire appel à une fonction il est nécessaire de :

- ☐ l'avoir déclarée
- ☐ avoir défini une constante symbolique de la taille de cette fonction
- ☐ l'avoir définie
- ☐ l'avoir déclarée et définie

18. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :

- ☐ un débogueur
- ☐ `printf("Valeur de n ? %g\n", n);`
- ☐ `scanf("%d", &n);`
- ☐ `printf("Valeur de n ? %d\n", n);`

19. 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

20. Vous utilisez une boucle `while` quand :

- ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
- ☐ vous avez déjà fait un `for` dans le même programme principal
- ☐ l'incrément de la variable de boucle n'est pas 1
- ☐ vous n'avez pas déclaré de fonction

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 enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses chants
   - ☐ ses blocs
   - ☐ ses champs
   - ☐ ses cases

2. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

   Alors l'expression `f(1)` prendra la valeur :
   - ☐ 0
   - ☐ 5
   - ☐ 4
   - ☐ 1

3. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
   - ☐ `n = carre(n);`
   - ☐ `n = carre(int n);`
   - ☐ `int carre(2);`
   - ☐ `int n = carre();`

4. 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

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

6. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

   Alors l'affichage sera :
   - ☐ `ABCDEF`
   - ☐ `A`
   - ☐ `ccccc`
   - ☐ `i`

7. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il risque d'afficher bonjour à la place de coucou
   - ☐ il n'affiche rien
   - ☐ il comporte une boucle infinie
   - ☐ il ne compile pas

8. 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

9. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
   - ☐ `x = racine(racine(x)*racine(x));`
   - ☐ `x = racine(x * x) - racine(x);`
   - ☐ `x = racine(2/3);`
   - ☐ `x - 1 = racine(x);`

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

11. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

- ☐ `char c;`
- ☐ `int char;`
- ☐ `char 'c';`
- ☐ `char "c";`

12. 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;`

13. 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

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. 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 variable non déclarée
- ☐ une faute de frappe dans un appel de fonction
- ☐ une directive préprocesseur `#include` manquante

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 ?

- ☐ `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`

17. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :

- ☐ `int afficher_date(date_s d);`
- ☐ `void afficher_date(date_s d);`
- ☐ `struct date_s afficher_date(struct date_s d);`
- ☐ `void afficher_date(struct date_s d);`

18. Avant de faire appel à une fonction il est nécessaire de :

- ☐ l'avoir déclarée et définie
- ☐ l'avoir déclarée
- ☐ avoir défini une constante symbolique de la taille de cette fonction
- ☐ l'avoir définie

19. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

- ☐ alphabétique
- ☐ un ordre quelconque
- ☐ dans lequel vous avez déclaré ces fonction
- ☐ dans lequel ces fonctions sont appelées dans le main

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 :

- ☐ les fichiers du disque
- ☐ certaines données de la mémoire de travail
- ☐ en temps d'accès
- ☐ 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. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :

- ☐ `struct toto_s toto;`
- ☐ `int struct toto_s = {3, -1e10};`
- ☐ `toto_s struct z = {3, 0.5};`
- ☐ `int toto.n = 3;`
- ☐ `toto_s n, x;`

2. Vous utilisez une boucle `while` quand :
   - ☐ vous n'avez pas déclaré de fonction
   - ☐ vous avez déjà fait un `for` dans le même programme principal
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance

3. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=3, b=5`

4. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ un ordre quelconque
   - ☐ alphabétique
   - ☐ dans lequel vous avez déclaré ces fonction
   - ☐ dans lequel ces fonctions sont appelées dans le main

5. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses chants
   - ☐ ses blocs
   - ☐ ses cases
   - ☐ ses champs

6. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
   - ☐ 0.5
   - ☐ 1
   - ☐ 1.5
   - ☐ 0

7. 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`

8. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
   - ☐ `int pgcd(int x, int x);`
   - ☐ `void pgcd(int x, int y);`
   - ☐ `int pgcd(int x, y);`
   - ☐ `int pgcd(int y, int x);`

9. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :
   - ☐ `A`
   - ☐ `ABCDEF`
   - ☐ `ccccc`
   - ☐ `i`

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

11. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
   - ☐ `void exposant(double x^n);`
   - ☐ `double exposant(double x, int n);`
   - ☐ `exposant(double x, int n, int r);`
   - ☐ `int exposant(double n, int x);`

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=%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);`

13. 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)

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. 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`

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
- ☐ 6
- ☐ 16
- ☐ 20

17. Une variable booléenne est un variable :

- ☐ jamais nulle
- ☐ à laquelle une valeur vient d'être affectée
- ☐ qui est vraie ou fausse
- ☐ réelle positive
- ☐ NaN (not a number, qui n'est pas un nombre)

18. 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

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

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. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

☐ 5

☐ 7

☐ 0

2. Le langage C est un langage

☐ interprété

☐ compilé

☐ composé

☐ lu, écrit, parlé

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

☐ le programme affiche `****`

☐ la variable x vaut 16

☐ la variable x vaut $-\frac{1}{2}$

4. 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)`

5. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

☐ `(a.n == b.n) && (a.x == b.x)`

☐ `a == b`

☐ `a = b`

☐ `a{n, x} == b{n, x}`

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

☐ qu'il faut indenter le fichier source

☐ qu'il faut lancer un déboggueur

☐ que l'on veut voir tous les avertissements

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

7. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

☐ 5

☐ 1

☐ 4

☐ 0

8. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

☐ 3

☐ 4

☐ 0

9. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :

☐ `int carre(2);`

☐ `int n = carre();`

☐ `n = carre(int n);`

☐ `n = carre(n);`

10. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

☐ `ABCDEF`

☐ `A`

☐ `i`

☐ `cccccc`

11. Avant de faire appel à une fonction il est nécessaire de :

☐ l'avoir déclarée

☐ l'avoir déclarée et définie

☐ avoir défini une constante symbolique de la taille de cette fonction

☐ l'avoir définie

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

13. Le type des réels en C est :
    - ☐ `real`
    - ☐ `char`
    - ☐ `double`
    - ☐ `int`

14. 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

15. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
    - ☐ `int factorielle(int x);`
    - ☐ `struct int factorielle(int n);`
    - ☐ `int factorielle(double n);`
    - ☐ `int factorielle();`

16. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
    - ☐ `A && B`
    - ☐ `(!A || B)`
    - ☐ `(A == TRUE) && (B == TRUE)`
    - ☐ `!(!A || B) == (A && !B)`

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  }
```

- ☐ 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

18. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=8, b=5`

19. Un fichier source est :
    - ☐ 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
    - ☐ un fichier texte qui sera traduit en instructions processeur
    - ☐ un document illisible pour les humains

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
- ☐ 3
- ☐ 16
- ☐ 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. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `(a<n) || (n>b)`
- ☐ `a<=n<=b`
- ☐ `(a<=n) && (n<=b)`
- ☐ `(n<=a) && (n<=b)`

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

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

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

4. 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

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
- ☐ 8 2
- ☐ 8 6 4 2 0

6. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il comporte une boucle infinie
- ☐ il ne compile pas
- ☐ il n'affiche rien
- ☐ il risque d'afficher bonjour à la place de coucou

7. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :

- ☐ `struct int factorielle(int n);`
- ☐ `int factorielle(int x);`
- ☐ `int factorielle(double n);`
- ☐ `int factorielle();`

8. Vous utilisez une boucle `while` quand :

- ☐ l'incrément de la variable de boucle n'est pas 1
- ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
- ☐ vous avez déjà fait un `for` dans le même programme principal
- ☐ vous n'avez pas déclaré de fonction

9. 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;`

10. L'écriture $\underline{101}$ en binaire correspond au nombre naturel :

- ☐ 5
- ☐ 4
- ☐ 3
- ☐ 101

11. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

- ☐ `n = factorielle();`
- ☐ `n = factorielle(p, q);`
- ☐ `int factorielle(int 2);`
- ☐ `printf("%d", factorielle(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 :

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

13. Le code suivant :

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

affichera :

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

14. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 4
- ☐ 3
- ☐ 0

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 :

- ☐ 0
- ☐ 4
- ☐ 8
- ☐ 16

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

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

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

18. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?

- ☐ `(A == TRUE) && (B == TRUE)`
- ☐ `(!A || B)`
- ☐ `!(!A || B) == (A && !B)`
- ☐ `A && B`

19. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(3,5)=8, a=3, b=5`

20. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :

- ☐ `int exposant(double n, int x);`
- ☐ `double exposant(double x, int n);`
- ☐ `void exposant(double x^n);`
- ☐ `exposant(double x, int n, int r);`

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 a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a == b`
- ☐ `a = b`
- ☐ `a{n, x} == b{n, x}`

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

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

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

4. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :

- ☐ `x = racine(x * x) - racine(x);`
- ☐ `x = racine(2/3);`
- ☐ `x - 1 = racine(x);`
- ☐ `x = racine(racine(x)*racine(x));`

5. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :

- ☐ `void saisie_utilisateur(int n);`
- ☐ `saisie_utilisateur(scanf(%d));`
- ☐ `int saisie_utilisateur();`
- ☐ `void saisie_utilisateur(char c);`

6. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :

- ☐ `n = carre(int n);`
- ☐ `int carre(2);`
- ☐ `n = carre(n);`
- ☐ `int n = carre();`

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

- ☐ `!`
- ☐ `<>`
- ☐ `!=`
- ☐ $\neq$

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

9. 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

10. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=8, b=5`

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

12. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :

- ☐ 7
- ☐ 3
- ☐ 111
- ☐ 8

13. L'écriture 101 en binaire correspond au nombre naturel :

- ☐ 4
- ☐ 5
- ☐ 101
- ☐ 3

14. Vous utilisez une boucle `while` quand :

- ☐ vous avez déjà fait un `for` dans le même programme principal
- ☐ vous n'avez pas déclaré de fonction
- ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
- ☐ l'incrément de la variable de boucle n'est pas 1

15. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?

- ☐ une directive préprocesseur `#include` manquante
- ☐ un désaccord entre la déclaration et la définition d'une fonction
- ☐ une fonction appelée avant sa déclaration
- ☐ une fonction déclarée mais non définie

16. Un enregistrement permet de grouper plusieurs valeurs dans :

- ☐ ses blocs
- ☐ ses chants
- ☐ ses champs
- ☐ ses cases

17. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :

- ☐ 0
- ☐ 0.5
- ☐ 1.5
- ☐ 1

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`
- ☐ `1 2 3 1 2`
- ☐ `0 0 1 1 2 2`

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. 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

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 cet avertissement apparaît à la compilation :
   `warning: implicit declaration of function ‘max’`
   , que doit-on chercher dans le programme ?
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une fonction déclarée mais non définie
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une directive préprocesseur `#include` manquante
2. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
   - ☐ `void pgcd(int x, int y);`
   - ☐ `int pgcd(int y, int x);`
   - ☐ `int pgcd(int x, int x);`
   - ☐ `int pgcd(int x, y);`
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
   - ☐ 8 6 4 2 0
   - ☐ 0 2 4 6 8
4. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
   - ☐ 0
   - ☐ 1
   - ☐ 0.5
   - ☐ 1.5
5. Avant de faire appel à une fonction il est nécessaire de :
   - ☐ avoir défini une constante symbolique de la taille de cette fonction
   - ☐ l'avoir définie
   - ☐ l'avoir déclarée
   - ☐ l'avoir déclarée et définie
6. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il comporte une boucle infinie
   - ☐ il n'affiche rien
   - ☐ il risque d'afficher bonjour à la place de coucou
   - ☐ il ne compile pas
7. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ init
   - ☐ main
   - ☐ begin
   - ☐ include
8. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

   précède la fonction `main()`, alors on peut écrire en début de `main()` :
   - ☐ `toto_s n, x;`
   - ☐ `struct toto_s toto;`
   - ☐ `int struct toto_s = {3, -1e10};`
   - ☐ `int toto.n = 3;`
   - ☐ `toto_s struct z = {3, 0.5};`
9. Vous utilisez une boucle `while` quand :
   - ☐ vous avez déjà fait un `for` dans le même programme principal
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ vous n'avez pas déclaré de fonction
   - ☐ l'incrément de la variable de boucle n'est pas 1
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. 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`
12. 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>`

13. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=3, b=5`

14. 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

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

16. Si cette erreur apparaît à la compilation : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction déclarée mais non définie

17. 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

18. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
    - ☐ `int n = carre();`
    - ☐ `int carre(2);`
    - ☐ `n = carre(n);`
    - ☐ `n = carre(int n);`

19. 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

20. 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);`

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 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

2. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
   ☐ `!(!A || B) == (A && !B)`
   ☐ `A && B`
   ☐ `(A == TRUE) && (B == TRUE)`
   ☐ `(!A || B)`

3. 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`

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

5. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

   Alors l'affichage sera :
   ☐ `i`
   ☐ `cccccc`
   ☐ `A`
   ☐ `ABCDEF`

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

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

8. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   ☐ 5
   ☐ 0
   ☐ 7

9. Vous utilisez une boucle `while` quand :
   ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   ☐ vous n'avez pas déclaré de fonction
   ☐ l'incrément de la variable de boucle n'est pas 1
   ☐ vous avez déjà fait un `for` dans le même programme principal

10. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
    ☐ 3
    ☐ 7
    ☐ 111
    ☐ 8

11. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
    ☐ `int pgcd(int x, int x);`
    ☐ `int pgcd(int y, int x);`
    ☐ `int pgcd(int x, y);`
    ☐ `void pgcd(int x, int y);`

12. 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

13. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 4
- ☐ 3
- ☐ 0

14. 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;`

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 = 5`
- ☐ `j = 4`
- ☐ `j = %d`

16. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :

- ☐ `n = carre(int n);`
- ☐ `n = carre(n);`
- ☐ `int n = carre();`
- ☐ `int carre(2);`

17. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 5
- ☐ 4
- ☐ 0
- ☐ 1

18. Si cette erreur apparaît à la compilation :
`erreur: conflicting types for ‘max’` , que doit-on chercher dans le programme ?

- ☐ une fonction déclarée mais non définie
- ☐ une directive préprocesseur `#include` manquante
- ☐ un désaccord entre la déclaration et la définition d'une fonction
- ☐ une fonction appelée avant sa déclaration

19. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il comporte une boucle infinie
- ☐ il n'affiche rien
- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il ne compile pas

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

- ☐ analyse syntaxique
- ☐ analyse lexicale
- ☐ analyse sémantique
- ☐ 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. Le langage C est un langage
   - ☐ lu, écrit, parlé
   - ☐ interprété
   - ☐ composé
   - ☐ compilé

2. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
   - ☐ `void afficher_date(date_s d);`
   - ☐ `int afficher_date(date_s d);`
   - ☐ `void afficher_date(struct date_s d);`
   - ☐ `struct date_s afficher_date(struct date_s 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 1 2 3 4 5 6
   - ☐ 0 2 4 6 8

4. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
   - ☐ `int n = carre();`
   - ☐ `n = carre(int n);`
   - ☐ `n = carre(n);`
   - ☐ `int carre(2);`

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
   - ☐ 1 2 3 4
   - ☐ 0 1 2 3 4

6. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
   - ☐ `printf("%d", factorielle(n));`
   - ☐ `n = factorielle(p, q);`
   - ☐ `int factorielle(int 2);`
   - ☐ `n = factorielle();`

7. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
   - ☐ `int exposant(double n, int x);`
   - ☐ `void exposant(double x^n);`
   - ☐ `double exposant(double x, int n);`
   - ☐ `exposant(double x, int n, int r);`

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=%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");`

9. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   - ☐ `f(3,5)=8, a=3, b=5`
   - ☐ `f(a,b)=13, a=8, b=5`
   - ☐ `f(a,b)=8, a=3, b=5`
   - ☐ `f(a,b)=8, a=8, b=5`

10. 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 variable non déclarée
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une faute de frappe dans un appel de fonction

11. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

    Alors l'affichage sera :
    - ☐ `cccccc`
    - ☐ `i`
    - ☐ `A`
    - ☐ `ABCDEF`

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
- ☐ 15
- ☐ 10
- ☐ 6

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. 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`

15. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 0
- ☐ 4
- ☐ 3

16. 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

17. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

- ☐ `char "c";`
- ☐ `char c;`
- ☐ `int char;`
- ☐ `char 'c';`

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

19. Un enregistrement permet de grouper plusieurs valeurs dans :

- ☐ ses champs
- ☐ ses blocs
- ☐ ses cases
- ☐ ses chants

20. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `a<=n<=b`
- ☐ `(n<=a) && (n<=b)`
- ☐ `(a<=n) && (n<=b)`
- ☐ `(a<n) || (n>b)`

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. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ dans lequel vous avez déclaré ces fonction
   - ☐ dans lequel ces fonctions sont appelées dans le main
   - ☐ alphabétique
   - ☐ un ordre quelconque
2. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

   Quelle est la condition `cond` :
   - ☐ `(a<=n) && (n<=b)`
   - ☐ `a<=n<=b`
   - ☐ `(n<=a) && (n<=b)`
   - ☐ `(a<n) || (n>b)`
3. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

   précède la fonction `main()`, alors on peut écrire en début de `main()` :
   - ☐ `toto_s struct z = {3, 0.5};`
   - ☐ `int toto.n = 3;`
   - ☐ `int struct toto_s = {3, -1e10};`
   - ☐ `struct toto_s toto;`
   - ☐ `toto_s n, x;`
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`
   - ☐ `0 1 2 0 1 2`
   - ☐ `1 2 1 2 3`
   - ☐ `0 1 0 1 0 1 0 1`
5. 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>`
6. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il comporte une boucle infinie
   - ☐ il n'affiche rien
   - ☐ il risque d'afficher bonjour à la place de coucou
   - ☐ il ne compile pas
7. Le langage C est un langage
   - ☐ compilé
   - ☐ composé
   - ☐ interprété
   - ☐ lu, écrit, parlé
8. 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`
9. 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
10. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction déclarée mais non définie
11. 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
12. L'écriture $\underline{101}$ en binaire correspond au nombre naturel :
    - ☐ 5
    - ☐ 4
    - ☐ 101
    - ☐ 3

13. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 7
- ☐ 0
- ☐ 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 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

15. 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

16. Une variable booléenne est un variable :
    - ☐ NaN (not a number, qui n'est pas un nombre)
    - ☐ jamais nulle
    - ☐ à laquelle une valeur vient d'être affectée
    - ☐ réelle positive
    - ☐ qui est vraie ou fausse

17. 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`

18. 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

19. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 0
- ☐ 3
- ☐ 4

20. Avant de faire appel à une fonction il est nécessaire de :
    - ☐ l'avoir déclarée et définie
    - ☐ avoir défini une constante symbolique de la taille de cette fonction
    - ☐ l'avoir déclarée
    - ☐ l'avoir définie

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 x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
   - ☐ 0
   - ☐ 0.5
   - ☐ 1
   - ☐ 1.5

2. Une variable booléenne est un variable :
   - ☐ à laquelle une valeur vient d'être affectée
   - ☐ réelle positive
   - ☐ qui est vraie ou fausse
   - ☐ NaN (not a number, qui n'est pas un nombre)
   - ☐ jamais nulle

3. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   - ☐ `f(a,b)=8, a=8, b=5`
   - ☐ `f(a,b)=8, a=3, b=5`
   - ☐ `f(3,5)=8, a=3, b=5`
   - ☐ `f(a,b)=13, a=8, b=5`

4. Si cet avertissement apparaît à la compilation :
   `warning: implicit declaration of function ‘max’`
   , que doit-on chercher dans le programme ?
   - ☐ une fonction appelée avant sa déclaration
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une fonction déclarée mais non définie

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
   - ☐ 4 3 2 1
   - ☐ 0 1 2 3

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 :
   - ☐ 0
   - ☐ 15
   - ☐ 10
   - ☐ 6

7. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ alphabétique
   - ☐ dans lequel vous avez déclaré ces fonction
   - ☐ un ordre quelconque
   - ☐ dans lequel ces fonctions sont appelées dans le main

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

9. 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

10. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses chants
    - ☐ ses champs
    - ☐ ses blocs
    - ☐ ses cases

11. 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

12. Le code suivant :

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

affichera :

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

13. L'écriture 111 en binaire correspond au nombre naturel :
    - ☐ 111
    - ☐ 7
    - ☐ 3
    - ☐ 8

14. Vous utilisez une boucle `while` quand :
    - ☐ vous avez déjà fait un `for` dans le même programme principal
    - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
    - ☐ vous n'avez pas déclaré de fonction
    - ☐ l'incrément de la variable de boucle n'est pas 1

15. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
    - ☐ `char afficher_menu(printf("menu"));`
    - ☐ `void afficher_menu();`
    - ☐ `double afficher_menu();`
    - ☐ `int afficher_menu();`
    - ☐ `int afficher_menu(int char);`

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

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

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 la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 4
- ☐ 3
- ☐ 0

20. 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
    - ☐ compiler un programme
    - ☐ 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. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
   - ☐ `int factorielle(int x);`
   - ☐ `int factorielle();`
   - ☐ `int factorielle(double n);`
   - ☐ `struct int factorielle(int n);`
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
   - ☐ un caractère interdit en C
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une faute de frappe dans un appel de fonction
3. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

   Alors l'expression `f(1)` prendra la valeur :
   - ☐ 4
   - ☐ 1
   - ☐ 0
   - ☐ 5
4. Le code suivant :

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

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

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

   affichera :
   - ☐ `Mineur`
   - ☐ `Majeur`
   - ☐ rien
   - ☐ `Mineur` `Majeur`
6. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
   - ☐ `n = carre(n);`
   - ☐ `int n = carre();`
   - ☐ `int carre(2);`
   - ☐ `n = carre(int n);`
7. Si cette erreur apparaît à la compilation : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une fonction déclarée mais non définie
8. 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
9. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses champs
   - ☐ ses chants
   - ☐ ses cases
   - ☐ ses blocs
10. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
    - ☐ 111
    - ☐ 7
    - ☐ 3
    - ☐ 8
11. 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>`
12. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ ouvrir un fichier texte
    - ☐ changer de répertoire courant
    - ☐ créer un fichier texte
    - ☐ créer un répertoire
13. Le type des réels en C est :
    - ☐ `char`
    - ☐ `int`
    - ☐ `double`
    - ☐ `real`

14. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :

- ☐ 0
- ☐ 0.5
- ☐ 1.5
- ☐ 1

15. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :

- ☐ `n = mccarthy();`
- ☐ `x = mccarthy(n);`
- ☐ `int mccarthy(int 2);`
- ☐ `n = mccarthy(p, q);`

16. 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

17. Vous utilisez une boucle `while` quand :

- ☐ vous n'avez pas déclaré de fonction
- ☐ l'incrément de la variable de boucle n'est pas 1
- ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
- ☐ vous avez déjà fait un `for` dans le même programme principal

18. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :

- ☐ `int afficher_date(date_s d);`
- ☐ `struct date_s afficher_date(struct date_s d);`
- ☐ `void afficher_date(struct date_s d);`
- ☐ `void afficher_date(date_s d);`

19. 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 0 1 2`
- ☐ `0 1 2 3 0 1 2`

20. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 0
- ☐ 4
- ☐ 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. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
   - ☐ `int saisie_utilisateur();`
   - ☐ `saisie_utilisateur(scanf(%d));`
   - ☐ `void saisie_utilisateur(int n);`
   - ☐ `void saisie_utilisateur(char c);`
2. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
   - ☐ `n = carre(int n);`
   - ☐ `int n = carre();`
   - ☐ `n = carre(n);`
   - ☐ `int carre(2);`
3. Le langage C est un langage
   - ☐ lu, écrit, parlé
   - ☐ interprété
   - ☐ composé
   - ☐ compilé
4. Vous utilisez une boucle `while` quand :
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous avez déjà fait un `for` dans le même programme principal
   - ☐ vous n'avez pas déclaré de fonction
5. 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
6. 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
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. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   - ☐ `f(a,b)=8, a=8, b=5`
   - ☐ `f(3,5)=8, a=3, b=5`
   - ☐ `f(a,b)=8, a=3, b=5`
   - ☐ `f(a,b)=13, a=8, b=5`
9. Une variable booléenne est un variable :
   - ☐ réelle positive
   - ☐ NaN (not a number, qui n'est pas un nombre)
   - ☐ jamais nulle
   - ☐ à laquelle une valeur vient d'être affectée
   - ☐ qui est vraie ou fausse
10. 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
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
    - ☐ mettre les blocs en séquence les uns à la suite des autres
    - ☐ retourner un bloc
    - ☐ répéter un bloc tant qu'une condition est vérifée
12. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

    - ☐ il risque d'afficher bonjour à la place de coucou
    - ☐ il comporte une boucle infinie
    - ☐ il n'affiche rien
    - ☐ il ne compile pas

13. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :

- ☐ `struct date_s afficher_date(struct date_s d);`
- ☐ `void afficher_date(struct date_s d);`
- ☐ `void afficher_date(date_s d);`
- ☐ `int afficher_date(date_s d);`

14. L'écriture 111 en binaire correspond au nombre naturel :

- ☐ 8
- ☐ 111
- ☐ 7
- ☐ 3

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 :

- ☐ 4
- ☐ 0
- ☐ 8
- ☐ 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 2 0 1 2`
- ☐ `1 2 3 1 2`
- ☐ `0 1 0 1 0 1`

17. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

- ☐ `int char;`
- ☐ `char 'c';`
- ☐ `char "c";`
- ☐ `char c;`

18. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `(a<=n) && (n<=b)`
- ☐ `(n<=a) && (n<=b)`
- ☐ `(a<n) || (n>b)`
- ☐ `a<=n<=b`

19. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 4
- ☐ 3
- ☐ 0

20. Si cette erreur apparaît à la compilation :
`erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?

- ☐ une fonction appelée avant sa déclaration
- ☐ une fonction déclarée mais non définie
- ☐ une directive préprocesseur `#include` manquante
- ☐ un désaccord entre la déclaration et la définition d'une fonction

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 `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :
   - ☐ `int n = pgcd();`
   - ☐ `n = pgcd(n, 3);`
   - ☐ `n = pgcd(int p, int q);`
   - ☐ `int pgcd(2);`
2. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   - ☐ 0
   - ☐ 7
   - ☐ 5
3. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

   précède la fonction `main()`, alors on peut écrire en début de `main()` :
   - ☐ `int struct toto_s = {3, -1e10};`
   - ☐ `struct toto_s toto;`
   - ☐ `int toto.n = 3;`
   - ☐ `toto_s struct z = {3, 0.5};`
   - ☐ `toto_s n, x;`
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 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
5. 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
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 = 0`
   - ☐ `j = %d`
   - ☐ `j = 4`
   - ☐ `j = 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 :
   - ☐ 4
   - ☐ 8
   - ☐ 0
   - ☐ 16
8. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses blocs
   - ☐ ses cases
   - ☐ ses chants
   - ☐ ses champs
9. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

   Quelle est la condition `cond` :
   - ☐ `(a<n) || (n>b)`
   - ☐ `a<=n<=b`
   - ☐ `(a<=n) && (n<=b)`
   - ☐ `(n<=a) && (n<=b)`
10. Le type des réels en C est :
    - ☐ `int`
    - ☐ `double`
    - ☐ `char`
    - ☐ `real`

11. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :

- ☐ `char afficher_menu(printf("menu"));`
- ☐ `void afficher_menu();`
- ☐ `int afficher_menu(int char);`
- ☐ `double afficher_menu();`
- ☐ `int afficher_menu();`

12. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :

- ☐ `void afficher_date(date_s d);`
- ☐ `int afficher_date(date_s d);`
- ☐ `void afficher_date(struct date_s d);`
- ☐ `struct date_s afficher_date(struct date_s d);`

13. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il ne compile pas
- ☐ il comporte une boucle infinie
- ☐ il n'affiche rien

14. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 1
- ☐ 4
- ☐ 5
- ☐ 0

15. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

Alors l'affichage sera :

- ☐ `A`
- ☐ `B`
- ☐ `b`
- ☐ `C`

16. Un fichier source est :

- ☐ un document qui doit être protégé
- ☐ 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 fichier texte qui sera traduit en instructions processeur

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 langage C est un langage

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

19. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :

- ☐ `void pgcd(int x, int y);`
- ☐ `int pgcd(int y, int x);`
- ☐ `int pgcd(int x, int x);`
- ☐ `int pgcd(int x, y);`

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

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

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 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

2. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :

   - ☐ 0
   - ☐ 4
   - ☐ 3

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. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

   Alors l'affichage sera :

   - ☐ `C`
   - ☐ `A`
   - ☐ `b`
   - ☐ `B`

5. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

   - ☐ dans lequel vous avez déclaré ces fonction
   - ☐ dans lequel ces fonctions sont appelées dans le main
   - ☐ alphabétique
   - ☐ un ordre quelconque

6. Une variable booléenne est un variable :

   - ☐ réelle positive
   - ☐ jamais nulle
   - ☐ qui est vraie ou fausse
   - ☐ NaN (not a number, qui n'est pas un nombre)
   - ☐ à laquelle une valeur vient d'être affectée

7. 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`

8. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

   - ☐ `char "c";`
   - ☐ `char 'c';`
   - ☐ `int char;`
   - ☐ `char c;`

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 unité de calcul spécialisée de l'ordinateur
   - ☐ une case mémoire interne au processeur qui sera manipulée directement lors des calculs

10. 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

11. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

    - ☐ `printf("%d", factorielle(n));`
    - ☐ `n = factorielle(p, q);`
    - ☐ `int factorielle(int 2);`
    - ☐ `n = factorielle();`

12. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 7
- ☐ 5
- ☐ 0

13. 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

14. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a == b`
- ☐ `a{n, x} == b{n, x}`
- ☐ `a = b`

15. Le code suivant :

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

affichera :

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

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

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

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
- ☐ 0 1 2 3 4
- ☐ 1 2 3 4

18. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?

- ☐ une fonction déclarée mais non définie
- ☐ une fonction appelée avant sa déclaration
- ☐ une directive préprocesseur `#include` manquante
- ☐ un désaccord entre la déclaration et la définition d'une fonction

19. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :

- ☐ 1.5
- ☐ 1
- ☐ 0.5
- ☐ 0

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 :

- ☐ certaines données de la mémoire de travail
- ☐ les fichiers du disque
- ☐ en temps d'accès
- ☐ 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 fichier source est :
   - ☐ 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 de référence du système
   - ☐ un document illisible pour les humains
2. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ begin
   - ☐ init
   - ☐ main
   - ☐ include
3. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   - ☐ 5
   - ☐ 7
   - ☐ 0
4. 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`
5. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
   - ☐ `int char;`
   - ☐ `char "c";`
   - ☐ `char 'c';`
   - ☐ `char c;`
6. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
   - ☐ `n = carre(n);`
   - ☐ `int carre(2);`
   - ☐ `int n = carre();`
   - ☐ `n = carre(int n);`
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 1 2 3
   - ☐ 0 1 2 3 4
   - ☐ 4 3 2 1 0
8. Le type des réels en C est :
   - ☐ `char`
   - ☐ `real`
   - ☐ `double`
   - ☐ `int`
9. Une variable booléenne est un variable :
   - ☐ NaN (not a number, qui n'est pas un nombre)
   - ☐ qui est vraie ou fausse
   - ☐ jamais nulle
   - ☐ à laquelle une valeur vient d'être affectée
   - ☐ réelle positive
10. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

    - ☐ il risque d'afficher bonjour à la place de coucou
    - ☐ il ne compile pas
    - ☐ il comporte une boucle infinie
    - ☐ il n'affiche rien
11. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

    Alors pour tester l'égalité de a et de b on utilise la condition :
    - ☐ `(a.n == b.n) && (a.x == b.x)`
    - ☐ `a == b`
    - ☐ `a = b`
    - ☐ `a{n, x} == b{n, x}`

12. 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
13. Sous unix (ou linux), la commande `cd` permet de :
    - ☐ changer de répertoire courant
    - ☐ jouer de la musique
    - ☐ récupérer un programme arrêté avec la commande `ab`
    - ☐ ouvir un bureau partagé (common desktop)
    - ☐ détruire un fichier
14. Si cette erreur apparaît à la compilation : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une fonction déclarée mais non définie
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
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=1;i<5;i=i+1)`
    - ☐ `for(i=0;i<=5;i=i+1)`
16. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ dans lequel ces fonctions sont appelées dans le main
    - ☐ alphabétique
    - ☐ un ordre quelconque
    - ☐ dans lequel vous avez déclaré ces fonction
17. 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`
18. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
    - ☐ `struct date_s afficher_date(struct date_s d);`
    - ☐ `void afficher_date(date_s d);`
    - ☐ `int afficher_date(date_s d);`
    - ☐ `void afficher_date(struct date_s d);`
19. 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
20. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
    - ☐ `(!A || B)`
    - ☐ `(A == TRUE) && (B == TRUE)`
    - ☐ `!(!A || B) == (A && !B)`
    - ☐ `A && B`

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'écriture 111 en binaire correspond au nombre naturel :
   - ☐ 3
   - ☐ 7
   - ☐ 8
   - ☐ 111

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

3. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

   Alors pour tester l'égalité de a et de b on utilise la condition :
   - ☐ `a = b`
   - ☐ `a == b`
   - ☐ `a{n, x} == b{n, x}`
   - ☐ `(a.n == b.n) && (a.x == b.x)`

4. 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>`

5. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
   - ☐ `x = racine(x * x) - racine(x);`
   - ☐ `x - 1 = racine(x);`
   - ☐ `x = racine(racine(x)*racine(x));`
   - ☐ `x = racine(2/3);`

6. Le type des réels en C est :
   - ☐ `real`
   - ☐ `char`
   - ☐ `int`
   - ☐ `double`

7. Si cette erreur apparaît à la compilation :
   `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
   - ☐ une fonction appelée avant sa déclaration
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une fonction déclarée mais non définie
   - ☐ une directive préprocesseur `#include` manquante

8. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

   Quelle est la condition `cond` :
   - ☐ `(a<n) || (n>b)`
   - ☐ `(n<=a) && (n<=b)`
   - ☐ `a<=n<=b`
   - ☐ `(a<=n) && (n<=b)`

9. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

   Alors l'expression `f(1)` prendra la valeur :
   - ☐ 0
   - ☐ 1
   - ☐ 5
   - ☐ 4

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

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
    - ☐ 1 2 3 4
    - ☐ 4 3 2 1 0

12. Vous utilisez une boucle `while` quand :
    - ☐ l'incrément de la variable de boucle n'est pas 1
    - ☐ vous n'avez pas déclaré de fonction
    - ☐ vous avez déjà fait un `for` dans le même programme principal
    - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance

13. 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

14. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
    - ☐ `char 'c';`
    - ☐ `char "c";`
    - ☐ `char c;`
    - ☐ `int char;`

15. 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

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

17. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses blocs
    - ☐ ses champs
    - ☐ ses cases
    - ☐ ses chants

18. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

    - ☐ il ne compile pas
    - ☐ il risque d'afficher bonjour à la place de coucou
    - ☐ il comporte une boucle infinie
    - ☐ il n'affiche rien

19. 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)

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 `5`
    - ☐ le terminal affiche `x = 5`
    - ☐ le terminal affiche "Faux"
    - ☐ le terminal affiche `x = 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. 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 variable non déclarée
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ un caractère interdit en C
2. L'écriture 111 en binaire correspond au nombre naturel :
   - ☐ 8
   - ☐ 7
   - ☐ 111
   - ☐ 3
3. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ dans lequel ces fonctions sont appelées dans le main
   - ☐ un ordre quelconque
   - ☐ alphabétique
   - ☐ dans lequel vous avez déclaré ces fonction
4. Vous utilisez une boucle `while` quand :
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous n'avez pas déclaré de fonction
   - ☐ vous avez déjà fait un `for` dans le même programme principal
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
5. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
   - ☐ `char 'c';`
   - ☐ `int char;`
   - ☐ `char c;`
   - ☐ `char "c";`
6. 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
7. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :
   - ☐ 3
   - ☐ 4
   - ☐ 0
8. Sous unix (ou linux), la commande `cd` permet de :
   - ☐ récupérer un programme arrêté avec la commande `ab`
   - ☐ détruire un fichier
   - ☐ ouvir un bureau partagé (common desktop)
   - ☐ jouer de la musique
   - ☐ changer de répertoire courant
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 1 2 3 4 5 6
   - ☐ 0 2 4 6 8
   - ☐ 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 `5`
    - ☐ le terminal affiche `x = 2`
    - ☐ le terminal affiche "Faux"
    - ☐ le terminal affiche `x = 5`
11. Si cet avertissement apparaît à la compilation :
    `warning: implicit declaration of function ‘max’`
    , que doit-on chercher dans le programme ?
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction déclarée mais non définie
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction appelée avant sa déclaration
12. Un bit est :
    - ☐ un chiffre binaire (0 ou 1)
    - ☐ l'instruction qui met fin à un programme
    - ☐ un battement d'horloge processeur
    - ☐ la longueur d'un mot mémoire
13. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
    - ☐ `int factorielle(double n);`
    - ☐ `int factorielle(int x);`
    - ☐ `int factorielle();`
    - ☐ `struct int factorielle(int n);`

14. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 7
- ☐ 5
- ☐ 0

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
- ☐ 6
- ☐ 0

16. 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];`

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 :

- ☐ 8
- ☐ 16
- ☐ 4
- ☐ 0

18. Le type des réels en C est :

- ☐ `double`
- ☐ `real`
- ☐ `int`
- ☐ `char`

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

- ☐ $\neq$
- ☐ `!=`
- ☐ `!`
- ☐ `<>`

20. Un enregistrement permet de grouper plusieurs valeurs dans :

- ☐ ses champs
- ☐ ses blocs
- ☐ ses chants
- ☐ ses cases

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 une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
   - ☐ `void afficher_date(struct date_s d);`
   - ☐ `int afficher_date(date_s d);`
   - ☐ `void afficher_date(date_s d);`
   - ☐ `struct date_s afficher_date(struct date_s d);`
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 :
   - ☐ 8
   - ☐ 0
   - ☐ 16
   - ☐ 4
3. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
   - ☐ 0.5
   - ☐ 0
   - ☐ 1.5
   - ☐ 1
4. L'écriture 111 en binaire correspond au nombre naturel :
   - ☐ 111
   - ☐ 7
   - ☐ 8
   - ☐ 3
5. 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 unité de calcul spécialisée de l'ordinateur
   - ☐ une gamme de fréquence de fonctionnement du processeur
6. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
   - ☐ `char c;`
   - ☐ `int char;`
   - ☐ `char 'c';`
   - ☐ `char "c";`
7. Le code suivant :

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

   affichera :
   - ☐ `Majeur`
   - ☐ `Mineur`
   - ☐ `Mineur`
`Majeur`
   - ☐ rien
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=%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");`
9. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
   - ☐ `struct int factorielle(int n);`
   - ☐ `int factorielle();`
   - ☐ `int factorielle(int x);`
   - ☐ `int factorielle(double n);`
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 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`
12. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
    - ☐ `void saisie_utilisateur(char c);`
    - ☐ `saisie_utilisateur(scanf(%d));`
    - ☐ `int saisie_utilisateur();`
    - ☐ `void saisie_utilisateur(int n);`
13. L'écriture 101 en binaire correspond au nombre naturel :
    - ☐ 5
    - ☐ 101
    - ☐ 3
    - ☐ 4
14. 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 qui doit être protégé
    - ☐ un document de référence du système

15. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :

    ☐ `x = racine(x * x) - racine(x);`

    ☐ `x - 1 = racine(x);`

    ☐ `x = racine(racine(x)*racine(x));`

    ☐ `x = racine(2/3);`

16. Vous utilisez une boucle `while` quand :

    ☐ vous avez déjà fait un `for` dans le même programme principal

    ☐ vous n'avez pas déclaré de fonction

    ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance

    ☐ l'incrément de la variable de boucle n'est pas 1

17. Une variable booléenne est un variable :

    ☐ réelle positive

    ☐ qui est vraie ou fausse

    ☐ NaN (not a number, qui n'est pas un nombre)

    ☐ à laquelle une valeur vient d'être affectée

    ☐ jamais nulle

18. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :

    ☐ `n = carre(n);`

    ☐ `int n = carre();`

    ☐ `int carre(2);`

    ☐ `n = carre(int n);`

19. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?

    ☐ `(!A || B)`

    ☐ `(A == TRUE) && (B == TRUE)`

    ☐ `A && B`

    ☐ `!(!A || B) == (A && !B)`

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'édition de liens

    ☐ 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. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

☐ 4

☐ 3

☐ 0

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 "Faux"

☐ le terminal affiche `x = 5`

☐ le terminal affiche `5`

☐ le terminal affiche `x = 2`

3. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

☐ `f(3,5)=8, a=3, b=5`

☐ `f(a,b)=8, a=8, b=5`

☐ `f(a,b)=13, a=8, b=5`

☐ `f(a,b)=8, a=3, b=5`

4. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

☐ `printf("%d", factorielle(n));`

☐ `n = factorielle();`

☐ `int factorielle(int 2);`

☐ `n = factorielle(p, q);`

5. 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

6. Le code suivant :

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

affichera :

☐ rien

☐ `Majeur`

☐ `Mineur`

☐ `Mineur`
`Majeur`

7. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

☐ `cccccc`

☐ `A`

☐ `ABCDEF`

☐ `i`

8. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

☐ dans lequel vous avez déclaré ces fonction

☐ alphabétique

☐ un ordre quelconque

☐ dans lequel ces fonctions sont appelées dans le main

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

☐ créer un répertoire

☐ créer un fichier texte

☐ changer de répertoire courant

☐ ouvrir un fichier texte

10. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

☐ `char c;`

☐ `char "c";`

☐ `char 'c';`

☐ `int char;`

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 0
- ☐ 4 3 2 1
- ☐ 0 1 2 3

12. Si cette erreur apparaît à la compilation : `erreur: conflicting types for ‘max’` , que doit-on chercher dans le programme ?
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une fonction déclarée mais non définie
    - ☐ une directive préprocesseur `#include` manquante
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. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
    - ☐ `x = racine(x * x) - racine(x);`
    - ☐ `x = racine(racine(x)*racine(x));`
    - ☐ `x = racine(2/3);`
    - ☐ `x - 1 = racine(x);`
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'édition de liens
    - ☐ l'analyse des entrées clavier
    - ☐ l'analyse sémantique
    - ☐ l'analyse harmonique
16. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
    - ☐ une fonction appelée avant sa déclaration
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction déclarée mais non définie
17. Le type des réels en C est :
    - ☐ `char`
    - ☐ `double`
    - ☐ `int`
    - ☐ `real`
18. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
    - ☐ `int[] new tableau(5);`
    - ☐ `int toto[taille=5];`
    - ☐ `int toto[5];`
    - ☐ `char tableau[5];`
    - ☐ `int tab[] = 5;`
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
- ☐ 4
- ☐ 8

20. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il n'affiche rien
- ☐ il comporte une boucle infinie
- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il ne compile pas

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 une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
   - ☐ `struct int factorielle(int n);`
   - ☐ `int factorielle(int x);`
   - ☐ `int factorielle(double n);`
   - ☐ `int factorielle();`
2. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
   - ☐ `int char;`
   - ☐ `char "c";`
   - ☐ `char c;`
   - ☐ `char 'c';`
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 illisible pour les humains
   - ☐ un document qui doit être protégé
   - ☐ un document de référence du système
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. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses champs
   - ☐ ses blocs
   - ☐ ses chants
   - ☐ ses cases
6. Une variable booléenne est un variable :
   - ☐ NaN (not a number, qui n'est pas un nombre)
   - ☐ qui est vraie ou fausse
   - ☐ jamais nulle
   - ☐ à laquelle une valeur vient d'être affectée
   - ☐ réelle positive
7. Si cette erreur apparaît à la compilation : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
   - ☐ une fonction déclarée mais non définie
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une fonction appelée avant sa déclaration
8. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

   Alors l'affichage sera :
   - ☐ `C`
   - ☐ `b`
   - ☐ `A`
   - ☐ `B`
9. 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
10. Le code suivant :

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

    affichera :
    - ☐ `Majeur`
    - ☐ `Mineur`
    - ☐ rien
    - ☐ `Mineur`
      `Majeur`
11. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
    - ☐ `void saisie_utilisateur(char c);`
    - ☐ `int saisie_utilisateur();`
    - ☐ `saisie_utilisateur(scanf(%d));`
    - ☐ `void saisie_utilisateur(int n);`
12. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

    Quelle est la condition `cond` :
    - ☐ `(a<n) || (n>b)`
    - ☐ `a<=n<=b`
    - ☐ `(a<=n) && (n<=b)`
    - ☐ `(n<=a) && (n<=b)`

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

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 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"

16. Vous utilisez une boucle `while` quand :
    - ☐ l'incrément de la variable de boucle n'est pas 1
    - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
    - ☐ vous n'avez pas déclaré de fonction
    - ☐ vous avez déjà fait un `for` dans le même programme principal

17. 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

18. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

    Alors pour tester l'égalité de a et de b on utilise la condition :
    - ☐ `a{n, x} == b{n, x}`
    - ☐ `a = b`
    - ☐ `(a.n == b.n) && (a.x == b.x)`
    - ☐ `a == b`

19. 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`

20. 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

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

2. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

   Alors l'affichage sera :
   - ☐ `A`
   - ☐ `cccccc`
   - ☐ `i`
   - ☐ `ABCDEF`

3. Le code suivant :

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

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

4. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ dans lequel ces fonctions sont appelées dans le main
   - ☐ dans lequel vous avez déclaré ces fonction
   - ☐ un ordre quelconque
   - ☐ alphabétique

5. 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`

6. Si cette erreur apparaît à la compilation :
   `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
   - ☐ une fonction déclarée mais non définie
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une directive préprocesseur `#include` manquante

7. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il ne compile pas
   - ☐ il n'affiche rien
   - ☐ il comporte une boucle infinie
   - ☐ il risque d'afficher bonjour à la place de coucou

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

9. Le code suivant :

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

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

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 ne pas perdre de temps avec la commutation de contexte
    - ☐ de doubler la mémoire disponible

11. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que `x` est une variable réelle définie et initialisée, il est incorrect d'écrire :
    - ☐ `x = racine(x * x) - racine(x);`
    - ☐ `x - 1 = racine(x);`
    - ☐ `x = racine(racine(x)*racine(x));`
    - ☐ `x = racine(2/3);`

12. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
```

```
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `(a<n) || (n>b)`
- ☐ `(a<=n) && (n<=b)`
- ☐ `(n<=a) && (n<=b)`
- ☐ `a<=n<=b`

13. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(3,5)=8, a=3, b=5`

14. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une fonction déclarée mais non définie
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une directive préprocesseur `#include` manquante

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. 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. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
    - ☐ 111
    - ☐ 8
    - ☐ 3
    - ☐ 7

18. 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

19. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 0
- ☐ 4
- ☐ 5
- ☐ 1

20. Vous utilisez une boucle `while` quand :
    - ☐ l'incrément de la variable de boucle n'est pas 1
    - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
    - ☐ vous avez déjà fait un `for` dans le même programme principal
    - ☐ vous n'avez pas déclaré de fonction

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 la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 4
- ☐ 0
- ☐ 1
- ☐ 5

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");
}
else
{
    printf("Majeur\n");
}
```

affichera :

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

4. Vous utilisez une boucle `while` quand :

- ☐ l'incrément de la variable de boucle n'est pas 1
- ☐ vous avez déjà fait un `for` dans le même programme principal
- ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
- ☐ vous n'avez pas déclaré de fonction

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

6. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :

- ☐ `int afficher_menu(int char);`
- ☐ `void afficher_menu();`
- ☐ `double afficher_menu();`
- ☐ `int afficher_menu();`
- ☐ `char afficher_menu(printf("menu"));`

7. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :

- ☐ `struct toto_s toto;`
- ☐ `toto_s n, x;`
- ☐ `toto_s struct z = {3, 0.5};`
- ☐ `int toto.n = 3;`
- ☐ `int struct toto_s = {3, -1e10};`

8. 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

9. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 5
- ☐ 7
- ☐ 0

10. 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

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

12. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction déclarée mais non définie
    - ☐ une fonction appelée avant sa déclaration
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
13. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
    - ☐ `int afficher_date(date_s d);`
    - ☐ `void afficher_date(struct date_s d);`
    - ☐ `void afficher_date(date_s d);`
    - ☐ `struct date_s afficher_date(struct date_s d);`
14. 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
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=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)`
16. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(3,5)=8, a=3, b=5`

17. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ créer un fichier texte
    - ☐ créer un répertoire
    - ☐ changer de répertoire courant
    - ☐ ouvrir un fichier texte
18. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
    - ☐ `int saisie_utilisateur();`
    - ☐ `void saisie_utilisateur(char c);`
    - ☐ `saisie_utilisateur(scanf(%d));`
    - ☐ `void saisie_utilisateur(int n);`
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 0 1 0 1 0 1`
- ☐ `0 1 2 0 1 2`
- ☐ `1 2 1 2 3`
- ☐ `0 0 0 1 1 1`

20. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
    - ☐ 0.5
    - ☐ 1.5
    - ☐ 0
    - ☐ 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. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une fonction déclarée mais non définie
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une fonction appelée avant sa déclaration
2. Si cette erreur apparaît à la compilation : `erreur: conflicting types for ‘max’` , que doit-on chercher dans le programme ?
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une fonction déclarée mais non définie
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
3. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il risque d'afficher bonjour à la place de coucou
   - ☐ il comporte une boucle infinie
   - ☐ il n'affiche rien
   - ☐ il ne compile pas
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. Une variable booléenne est un variable :
   - ☐ qui est vraie ou fausse
   - ☐ à laquelle une valeur vient d'être affectée
   - ☐ réelle positive
   - ☐ NaN (not a number, qui n'est pas un nombre)
   - ☐ jamais nulle
6. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
   - ☐ `printf("%d", factorielle(n));`
   - ☐ `n = factorielle();`
   - ☐ `int factorielle(int 2);`
   - ☐ `n = factorielle(p, q);`
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
   - ☐ 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
8. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
   - ☐ `int pgcd(int x, y);`
   - ☐ `int pgcd(int x, int x);`
   - ☐ `void pgcd(int x, int y);`
   - ☐ `int pgcd(int y, int x);`
9. Si `x` est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
   - ☐ 1
   - ☐ 0
   - ☐ 1.5
   - ☐ 0.5
10. 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
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  ...
11  int main() {
12      int x = 5;
13
14      x = 3 * x + 1;
15
16      ...
17  }
```

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

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. L'écriture 101 en binaire correspond au nombre naturel :
    - ☐ 5
    - ☐ 4
    - ☐ 3
    - ☐ 101
15. Avant de faire appel à une fonction il est nécessaire de :
    - ☐ avoir défini une constante symbolique de la taille de cette fonction
    - ☐ l'avoir déclarée et définie
    - ☐ l'avoir définie
    - ☐ l'avoir déclarée
16. Le code suivant :

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

    affichera :
    - ☐ `Mineur`
`Majeur`
    - ☐ `Mineur`
    - ☐ rien
    - ☐ `Majeur`
17. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

    Alors l'expression `f(0)` prendra la valeur :
    - ☐ 3
    - ☐ 0
    - ☐ 4
18. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses chants
    - ☐ ses cases
    - ☐ ses champs
    - ☐ ses blocs
19. 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>`
20. 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)

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 < 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`

2. 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

3. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
   - ☐ 111
   - ☐ 8
   - ☐ 3
   - ☐ 7

4. Le code suivant :

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

affichera :

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

5. Si cette erreur apparaît à la compilation :
   `erreur: conflicting types for ‘max’` , que doit-on chercher dans le programme ?
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une fonction appelée avant sa déclaration
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une fonction déclarée mais non définie

6. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
   - ☐ `void exposant(double x^n);`
   - ☐ `int exposant(double n, int x);`
   - ☐ `double exposant(double x, int n);`
   - ☐ `exposant(double x, int n, int r);`

7. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

Alors l'affichage sera :

- ☐ `A`
- ☐ `C`
- ☐ `b`
- ☐ `B`

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=%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);`

9. Si cet avertissement apparaît à la compilation :
   `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une fonction déclarée mais non définie
   - ☐ un désaccord entre la déclaration et la définition d'une fonction

10. Un fichier source est :
    - ☐ un document illisible pour les humains
    - ☐ un fichier texte qui sera traduit en instructions processeur
    - ☐ 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

11. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :
    - ☐ `int pgcd(2);`
    - ☐ `n = pgcd(n, 3);`
    - ☐ `int n = pgcd();`
    - ☐ `n = pgcd(int p, int q);`

12. 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

13. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=3, b=5`

14. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
    - ☐ `int carre(2);`
    - ☐ `int n = carre();`
    - ☐ `n = carre(int n);`
    - ☐ `n = carre(n);`

15. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ un ordre quelconque
    - ☐ alphabétique
    - ☐ dans lequel ces fonctions sont appelées dans le main
    - ☐ dans lequel vous avez déclaré ces fonction

16. Avant de faire appel à une fonction il est nécessaire de :
    - ☐ l'avoir déclarée
    - ☐ l'avoir déclarée et définie
    - ☐ avoir défini une constante symbolique de la taille de cette fonction
    - ☐ l'avoir définie

17. 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>`

18. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
    - ☐ `void saisie_utilisateur(int n);`
    - ☐ `int saisie_utilisateur();`
    - ☐ `void saisie_utilisateur(char c);`
    - ☐ `saisie_utilisateur(scanf(%d));`

19. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
    - ☐ `char "c";`
    - ☐ `int char;`
    - ☐ `char c;`
    - ☐ `char 'c';`

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 3 2 1
- ☐ 0 1 2 3 4
- ☐ 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 déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
   - ☐ `char tableau[5];`
   - ☐ `int[] new tableau(5);`
   - ☐ `int tab[] = 5;`
   - ☐ `int toto[taille=5];`
   - ☐ `int toto[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 x
   - ☐ la variable x vaut $-\frac{1}{2}$
   - ☐ le programme affiche `****`

3. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une fonction déclarée mais non définie

4. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
   - ☐ `printf("Valeur de n ? %d\n", n);`
   - ☐ `scanf("%d", &n);`
   - ☐ un débogueur
   - ☐ `printf("Valeur de n ? %g\n", n);`

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

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
   - ☐ 10
   - ☐ 0
   - ☐ 6

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

8. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   - ☐ `f(a,b)=8, a=3, b=5`
   - ☐ `f(a,b)=8, a=8, b=5`
   - ☐ `f(a,b)=13, a=8, b=5`
   - ☐ `f(3,5)=8, a=3, b=5`

9. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
   - ☐ `int exposant(double n, int x);`
   - ☐ `exposant(double x, int n, int r);`
   - ☐ `double exposant(double x, int n);`
   - ☐ `void exposant(double x^n);`

10. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
    - ☐ `int factorielle(double n);`
    - ☐ `struct int factorielle(int n);`
    - ☐ `int factorielle();`
    - ☐ `int factorielle(int x);`

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

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 :

- ☐ 3
- ☐ 20
- ☐ 16
- ☐ 6

13. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

- ☐ `n = factorielle();`
- ☐ `int factorielle(int 2);`
- ☐ `n = factorielle(p, q);`
- ☐ `printf("%d", factorielle(n));`

14. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 0
- ☐ 7
- ☐ 5

15. 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

16. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

- ☐ dans lequel vous avez déclaré ces fonction
- ☐ un ordre quelconque
- ☐ dans lequel ces fonctions sont appelées dans le main
- ☐ alphabétique

17. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 3
- ☐ 4
- ☐ 0

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

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

20. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :

- ☐ 0
- ☐ 0.5
- ☐ 1
- ☐ 1.5

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. Sous unix (ou linux), la commande `cd` permet de :
   - ☐ changer de répertoire courant
   - ☐ détruire un fichier
   - ☐ récupérer un programme arrêté avec la commande `ab`
   - ☐ jouer de la musique
   - ☐ ouvir un bureau partagé (common desktop)
2. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
   - ☐ `char 'c';`
   - ☐ `char "c";`
   - ☐ `int char;`
   - ☐ `char c;`
3. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function 'max'` , que doit-on chercher dans le programme ?
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une fonction déclarée mais non définie
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une directive préprocesseur `#include` manquante
4. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

   Alors l'affichage sera :
   - ☐ `i`
   - ☐ `cccccc`
   - ☐ `A`
   - ☐ `ABCDEF`
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 :
   - ☐ 6
   - ☐ 15
   - ☐ 10
   - ☐ 0
6. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
   - ☐ `int factorielle();`
   - ☐ `int factorielle(int x);`
   - ☐ `int factorielle(double n);`
   - ☐ `struct int factorielle(int n);`
7. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

   Alors pour tester l'égalité de a et de b on utilise la condition :
   - ☐ `(a.n == b.n) && (a.x == b.x)`
   - ☐ `a == b`
   - ☐ `a{n, x} == b{n, x}`
   - ☐ `a = b`
8. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   - ☐ 5
   - ☐ 0
   - ☐ 7
9. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
   - ☐ `n = factorielle(p, q);`
   - ☐ `int factorielle(int 2);`
   - ☐ `printf("%d", factorielle(n));`
   - ☐ `n = factorielle();`
10. 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
11. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

    Alors l'expression `f(1)` prendra la valeur :
    - ☐ 5
    - ☐ 0
    - ☐ 1
    - ☐ 4

12. 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

13. 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

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

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

16. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :

- ☐ `int saisie_utilisateur();`
- ☐ `void saisie_utilisateur(char c);`
- ☐ `saisie_utilisateur(scanf(%d));`
- ☐ `void saisie_utilisateur(int n);`

17. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :

- ☐ `printf("Valeur de n ? %d\n", n);`
- ☐ `printf("Valeur de n ? %g\n", n);`
- ☐ un débogueur
- ☐ `scanf("%d", &n);`

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

19. 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

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. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=13, a=8, b=5`

2. 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

3. 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

4. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
   - ☐ 3
   - ☐ 7
   - ☐ 8
   - ☐ 111

5. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il ne compile pas
   - ☐ il risque d'afficher bonjour à la place de coucou
   - ☐ il comporte une boucle infinie
   - ☐ il n'affiche rien

6. 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

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
   - ☐ 1 2 3 4
   - ☐ 4 3 2 1 0

8. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
   - ☐ `exposant(double x, int n, int r);`
   - ☐ `int exposant(double n, int x);`
   - ☐ `double exposant(double x, int n);`
   - ☐ `void exposant(double x^n);`

9. Le code suivant :

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

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

10. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
    - ☐ `int char;`
    - ☐ `char "c";`
    - ☐ `char c;`
    - ☐ `char 'c';`

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

12. Si cette erreur apparaît à la compilation : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction déclarée mais non définie
    - ☐ une fonction appelée avant sa déclaration

13. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

    Alors l'expression `g(0)` prendra la valeur :
    - ☐ 5
    - ☐ 7
    - ☐ 0

14. 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

15. Le code suivant :

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

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

16. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
    - ☐ `x = racine(2/3);`
    - ☐ `x = racine(x * x) - racine(x);`
    - ☐ `x - 1 = racine(x);`
    - ☐ `x = racine(racine(x)*racine(x));`

17. 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

18. Une variable booléenne est un variable :
    - ☐ réelle positive
    - ☐ qui est vraie ou fausse
    - ☐ NaN (not a number, qui n'est pas un nombre)
    - ☐ à laquelle une valeur vient d'être affectée
    - ☐ jamais nulle

19. 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`

20. Avant de faire appel à une fonction il est nécessaire de :
    - ☐ avoir défini une constante symbolique de la taille de cette fonction
    - ☐ l'avoir définie
    - ☐ l'avoir déclarée et définie
    - ☐ l'avoir déclarée

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 de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
   - ☐ `char c;`
   - ☐ `int char;`
   - ☐ `char "c";`
   - ☐ `char 'c';`
2. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
   - ☐ `x - 1 = racine(x);`
   - ☐ `x = racine(2/3);`
   - ☐ `x = racine(x * x) - racine(x);`
   - ☐ `x = racine(racine(x)*racine(x));`
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
   - ☐ 1 2 3 4
   - ☐ 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 y vaut 5
   - ☐ la variable x vaut 0
   - ☐ le programme affiche "Faux"
   - ☐ la variable x vaut 5 et la variable y vaut 0
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 3 0 1 2`
   - ☐ `0 0 1 1 2 2 3`
   - ☐ `0 1 2 0 1 2`
   - ☐ `0 1 2 0 1 2 3`
6. 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
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 bus explose
   - ☐ le terminal affiche 8
   - ☐ la case mémoire 8 contiendra 16
   - ☐ la case mémoire 8 contiendra 0
8. Vous utilisez une boucle `while` quand :
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ vous n'avez pas déclaré de fonction
   - ☐ vous avez déjà fait un `for` dans le même programme principal
9. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse syntaxique
   - ☐ analyse lexicale
   - ☐ analyse harmonique
   - ☐ analyse sémantique
10. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
    - ☐ `n = mccarthy();`
    - ☐ `int mccarthy(int 2);`
    - ☐ `x = mccarthy(n);`
    - ☐ `n = mccarthy(p, q);`
11. Quel est l'opérateur de différence en C :
    - ☐ `!=`
    - ☐ `!`
    - ☐ ≠
    - ☐ `<>`
12. Avant de faire appel à une fonction il est nécessaire de :
    - ☐ l'avoir déclarée et définie
    - ☐ l'avoir définie
    - ☐ avoir défini une constante symbolique de la taille de cette fonction
    - ☐ l'avoir déclarée

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  }
```

- ☐ 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

14. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 0
- ☐ 4
- ☐ 3

15. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il comporte une boucle infinie
- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il n'affiche rien
- ☐ il ne compile pas

16. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :

- ☐ `n = pgcd(n, 3);`
- ☐ `int pgcd(2);`
- ☐ `n = pgcd(int p, int q);`
- ☐ `int n = pgcd();`

17. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?

- ☐ `(A == TRUE) && (B == TRUE)`
- ☐ `(!A || B)`
- ☐ `!(!A || B) == (A && !B)`
- ☐ `A && B`

18. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :

- ☐ `scanf("%d", &n);`
- ☐ un débogueur
- ☐ `printf("Valeur de n ? %g\n", n);`
- ☐ `printf("Valeur de n ? %d\n", n);`

19. 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)

20. 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
- ☐ une faute de frappe dans un appel de fonction
- ☐ un caractère interdit en 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. Le code suivant :

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

affichera :

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

2. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
   - ☐ une fonction déclarée mais non définie
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une directive préprocesseur `#include` manquante

3. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

Alors l'affichage sera :

- ☐ `C`
- ☐ `B`
- ☐ `b`
- ☐ `A`

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

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

6. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
   - ☐ `int pgcd(int x, int x);`
   - ☐ `int pgcd(int x, y);`
   - ☐ `void pgcd(int x, int y);`
   - ☐ `int pgcd(int y, int x);`

7. 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`

8. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

- ☐ `cccccc`
- ☐ `A`
- ☐ `i`
- ☐ `ABCDEF`

9. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
   - ☐ `void exposant(double x^n);`
   - ☐ `int exposant(double n, int x);`
   - ☐ `exposant(double x, int n, int r);`
   - ☐ `double exposant(double x, int n);`

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

11. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il comporte une boucle infinie
- ☐ il ne compile pas
- ☐ il n'affiche rien

12. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
    - ☐ `x - 1 = racine(x);`
    - ☐ `x = racine(x * x) - racine(x);`
    - ☐ `x = racine(2/3);`
    - ☐ `x = racine(racine(x)*racine(x));`
13. 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
14. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
    - ☐ `int afficher_menu(int char);`
    - ☐ `int afficher_menu();`
    - ☐ `void afficher_menu();`
    - ☐ `double afficher_menu();`
    - ☐ `char afficher_menu(printf("menu"));`
15. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses blocs
    - ☐ ses chants
    - ☐ ses cases
    - ☐ ses champs
16. 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
17. Le langage C est un langage
    - ☐ composé
    - ☐ interprété
    - ☐ compilé
    - ☐ lu, écrit, parlé
18. 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>`
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
    - ☐ certaines données de la mémoire de travail
    - ☐ des processus
    - ☐ les fichiers du disque
20. Si cette erreur apparaît à la compilation :
    `erreur: conflicting types for ‘max’` , que doit-on chercher dans le programme ?
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une fonction déclarée mais non définie

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

affichera :

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

2. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `a<=n<=b`
- ☐ `(n<=a) && (n<=b)`
- ☐ `(a<n) || (n>b)`
- ☐ `(a<=n) && (n<=b)`

3. 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

4. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :

- ☐ `x - 1 = racine(x);`
- ☐ `x = racine(x * x) - racine(x);`
- ☐ `x = racine(2/3);`
- ☐ `x = racine(racine(x)*racine(x));`

5. Le langage C est un langage

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

6. Si cet avertissement apparaît à la compilation :
`warning: implicit declaration of function ‘max’`
, que doit-on chercher dans le programme ?

- ☐ une directive préprocesseur `#include` manquante
- ☐ une fonction déclarée mais non définie
- ☐ un désaccord entre la déclaration et la définition d'une fonction
- ☐ une fonction appelée avant sa déclaration

7. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 7
- ☐ 0
- ☐ 5

8. Vous utilisez une boucle `while` quand :

- ☐ vous n'avez pas déclaré de fonction
- ☐ vous avez déjà fait un `for` dans le même programme principal
- ☐ l'incrément de la variable de boucle n'est pas 1
- ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance

9. Un fichier source est :

- ☐ un fichier texte qui sera traduit en instructions processeur
- ☐ un document de référence du système
- ☐ 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

10. 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

11. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 4
- ☐ 3
- ☐ 0

12. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

- ☐ `n = factorielle(p, q);`
- ☐ `n = factorielle();`
- ☐ `int factorielle(int 2);`
- ☐ `printf("%d", factorielle(n));`

13. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

- ☐ `char c;`
- ☐ `char "c";`
- ☐ `char 'c';`
- ☐ `int char;`

14. 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

15. 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>`

16. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :

- ☐ `n = mccarthy(p, q);`
- ☐ `x = mccarthy(n);`
- ☐ `int mccarthy(int 2);`
- ☐ `n = mccarthy();`

17. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 4
- ☐ 1
- ☐ 0
- ☐ 5

18. 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 toto[5];`
- ☐ `int[] new tableau(5);`

19. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :

- ☐ `int saisie_utilisateur();`
- ☐ `void saisie_utilisateur(char c);`
- ☐ `saisie_utilisateur(scanf(%d));`
- ☐ `void saisie_utilisateur(int n);`

20. L'écriture $\underline{101}$ en binaire correspond au nombre naturel :

- ☐ 3
- ☐ 5
- ☐ 4
- ☐ 101

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 la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 3
- ☐ 4
- ☐ 0

2. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(a,b)=8, a=3, b=5`

3. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :

- ☐ `scanf("%d", &n);`
- ☐ `printf("Valeur de n ? %d\n", n);`
- ☐ un débogueur
- ☐ `printf("Valeur de n ? %g\n", n);`

4. 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

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. Vous utilisez une boucle `while` quand :

- ☐ vous avez déjà fait un `for` dans le même programme principal
- ☐ vous n'avez pas déclaré de fonction
- ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
- ☐ l'incrément de la variable de boucle n'est pas 1

7. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :

- ☐ `n = carre(n);`
- ☐ `int carre(2);`
- ☐ `int n = carre();`
- ☐ `n = carre(int n);`

8. 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`

9. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :

- ☐ `void afficher_date(struct date_s d);`
- ☐ `int afficher_date(date_s d);`
- ☐ `void afficher_date(date_s d);`
- ☐ `struct date_s afficher_date(struct date_s d);`

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. 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`

12. Le code suivant :

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

affichera :

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

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
    - ☐ mettre les blocs en séquence les uns à la suite des autres
    - ☐ retourner un bloc
14. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses champs
    - ☐ ses cases
    - ☐ ses blocs
    - ☐ ses chants
15. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :
    - ☐ `n = pgcd(n, 3);`
    - ☐ `n = pgcd(int p, int q);`
    - ☐ `int pgcd(2);`
    - ☐ `int n = pgcd();`
16. Un fichier source est :
    - ☐ 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
    - ☐ un fichier texte qui sera traduit en instructions processeur
17. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction appelée avant sa déclaration
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction déclarée mais non définie
18. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
    - ☐ `exposant(double x, int n, int r);`
    - ☐ `void exposant(double x^n);`
    - ☐ `double exposant(double x, int n);`
    - ☐ `int exposant(double n, int x);`
19. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
    - ☐ `int char;`
    - ☐ `char 'c';`
    - ☐ `char c;`
    - ☐ `char "c";`
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. Vous utilisez une boucle `while` quand :
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ vous avez déjà fait un `for` dans le même programme principal
   - ☐ vous n'avez pas déclaré de fonction
2. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
   - ☐ `int factorielle();`
   - ☐ `struct int factorielle(int n);`
   - ☐ `int factorielle(double n);`
   - ☐ `int factorielle(int x);`
3. 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`
4. Sur unix (ou linux), la commande `mkdir` permet de :
   - ☐ créer un fichier texte
   - ☐ créer un répertoire
   - ☐ ouvrir un fichier texte
   - ☐ changer de répertoire courant
5. 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`
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 :
   - ☐ 4
   - ☐ 8
   - ☐ 0
   - ☐ 16
7. 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"
8. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
   - ☐ une fonction déclarée mais non définie
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
9. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses champs
   - ☐ ses cases
   - ☐ ses chants
   - ☐ ses blocs
10. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
    - ☐ 8
    - ☐ 111
    - ☐ 3
    - ☐ 7
11. Avant de faire appel à une fonction il est nécessaire de :
    - ☐ avoir défini une constante symbolique de la taille de cette fonction
    - ☐ l'avoir déclarée
    - ☐ l'avoir déclarée et définie
    - ☐ l'avoir définie
12. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :
    - ☐ `int pgcd(2);`
    - ☐ `n = pgcd(n, 3);`
    - ☐ `int n = pgcd();`
    - ☐ `n = pgcd(int p, int q);`
13. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
    - ☐ `int char;`
    - ☐ `char "c";`
    - ☐ `char 'c';`
    - ☐ `char c;`
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=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)`

15. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :

- ☐ 1.5
- ☐ 0.5
- ☐ 0
- ☐ 1

16. Le code suivant :

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

affichera :

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

17. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 1
- ☐ 4
- ☐ 0
- ☐ 5

18. L'écriture 101 en binaire correspond au nombre naturel :

- ☐ 5
- ☐ 101
- ☐ 3
- ☐ 4

19. Le code suivant :

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

affichera :

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

20. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il n'affiche rien
- ☐ il comporte une boucle infinie
- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il ne compile pas

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 :
   - ☐ 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
2. Avant de faire appel à une fonction il est nécessaire de :
   - ☐ l'avoir définie
   - ☐ l'avoir déclarée
   - ☐ avoir défini une constante symbolique de la taille de cette fonction
   - ☐ l'avoir déclarée et définie
3. 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>`
4. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
   - ☐ `n = mccarthy(p, q);`
   - ☐ `int mccarthy(int 2);`
   - ☐ `x = mccarthy(n);`
   - ☐ `n = mccarthy();`
5. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
   - ☐ 1.5
   - ☐ 0
   - ☐ 1
   - ☐ 0.5
6. 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 document de référence du système
   - ☐ un document qui doit être protégé
   - ☐ un fichier texte qui sera traduit en instructions processeur
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
   - ☐ 4 3 2 1 0
   - ☐ 4 3 2 1
   - ☐ 1 2 3 4
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. Sur unix (ou linux), la commande `mkdir` permet de :
   - ☐ ouvrir un fichier texte
   - ☐ changer de répertoire courant
   - ☐ créer un répertoire
   - ☐ créer un fichier texte
10. 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`
    - ☐ `1 2 3 1 2`
    - ☐ `0 1 0 1 0 1`
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. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

    - ☐ il ne compile pas
    - ☐ il comporte une boucle infinie
    - ☐ il risque d'afficher bonjour à la place de coucou
    - ☐ il n'affiche rien
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
- ☐ 4 3 2 1 0
- ☐ 4 3 2 1
- ☐ 1 2 3 4

14. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
    - ☐ `int n = carre();`
    - ☐ `n = carre(int n);`
    - ☐ `n = carre(n);`
    - ☐ `int carre(2);`
15. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses cases
    - ☐ ses blocs
    - ☐ ses chants
    - ☐ ses champs
16. Le code suivant :

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

affichera :

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

17. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `a == b`
- ☐ `a{n, x} == b{n, x}`
- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a = b`

18. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ un ordre quelconque
    - ☐ dans lequel vous avez déclaré ces fonction
    - ☐ alphabétique
    - ☐ dans lequel ces fonctions sont appelées dans le main
19. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 4
- ☐ 1
- ☐ 0
- ☐ 5

20. Si cette erreur apparaît à la compilation :
`erreur: conflicting types for ‘max’` , que doit-on chercher dans le programme ?
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction déclarée mais non définie
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction appelée avant sa déclaration

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 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
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 :
   - ☐ 6
   - ☐ 0
   - ☐ 10
   - ☐ 15
3. Le langage C est un langage
   - ☐ interprété
   - ☐ composé
   - ☐ lu, écrit, parlé
   - ☐ compilé
4. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
   - ☐ `x = racine(x * x) - racine(x);`
   - ☐ `x - 1 = racine(x);`
   - ☐ `x = racine(racine(x)*racine(x));`
   - ☐ `x = racine(2/3);`
5. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
   - ☐ `!(!A || B) == (A && !B)`
   - ☐ `(A == TRUE) && (B == TRUE)`
   - ☐ `A && B`
   - ☐ `(!A || B)`
6. Une variable booléenne est un variable :
   - ☐ NaN (not a number, qui n'est pas un nombre)
   - ☐ jamais nulle
   - ☐ qui est vraie ou fausse
   - ☐ réelle positive
   - ☐ à laquelle une valeur vient d'être affectée
7. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
   - ☐ `int afficher_date(date_s d);`
   - ☐ `void afficher_date(struct date_s d);`
   - ☐ `struct date_s afficher_date(struct date_s d);`
   - ☐ `void afficher_date(date_s d);`
8. Le type des réels en C est :
   - ☐ `real`
   - ☐ `char`
   - ☐ `int`
   - ☐ `double`
9. 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
10. Un programme en langage C doit comporter une et une seule définition de la fonction :
    - ☐ include
    - ☐ init
    - ☐ main
    - ☐ begin
11. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ ouvrir un fichier texte
    - ☐ créer un répertoire
    - ☐ créer un fichier texte
    - ☐ changer de répertoire courant
12. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

    Alors pour tester l'égalité de a et de b on utilise la condition :
    - ☐ `a = b`
    - ☐ `a{n, x} == b{n, x}`
    - ☐ `a == b`
    - ☐ `(a.n == b.n) && (a.x == b.x)`
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  }
```

    - ☐ 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

14. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

Alors l'affichage sera :

- ☐ `A`
- ☐ `C`
- ☐ `b`
- ☐ `B`

15. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :

- ☐ `printf("Valeur de n ? %d\n", n);`
- ☐ `printf("Valeur de n ? %g\n", n);`
- ☐ `scanf("%d", &n);`
- ☐ un débogueur

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

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

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

19. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 7
- ☐ 5
- ☐ 0

20. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

- ☐ `int factorielle(int 2);`
- ☐ `n = factorielle(p, q);`
- ☐ `printf("%d", factorielle(n));`
- ☐ `n = factorielle();`

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 : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
   - ☐ une fonction déclarée mais non définie
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ une fonction appelée avant sa déclaration
   - ☐ un désaccord entre la déclaration et la définition d'une fonction

2. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   - ☐ 7
   - ☐ 0
   - ☐ 5

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
   - ☐ 16
   - ☐ 4
   - ☐ 0

4. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
   - ☐ `(!A || B)`
   - ☐ `A && B`
   - ☐ `(A == TRUE) && (B == TRUE)`
   - ☐ `!(!A || B) == (A && !B)`

5. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
   - ☐ 1
   - ☐ 0.5
   - ☐ 1.5
   - ☐ 0

6. 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

7. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses cases
   - ☐ ses blocs
   - ☐ ses chants
   - ☐ ses champs

8. 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 = 5`
   - ☐ `j = 0`

9. 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`

10. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
    - ☐ `void afficher_menu();`
    - ☐ `int afficher_menu();`
    - ☐ `int afficher_menu(int char);`
    - ☐ `char afficher_menu(printf("menu"));`
    - ☐ `double afficher_menu();`

11. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
    - ☐ `printf("%d", factorielle(n));`
    - ☐ `n = factorielle();`
    - ☐ `n = factorielle(p, q);`
    - ☐ `int factorielle(int 2);`

12. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

    Alors l'expression `f(0)` prendra la valeur :
    - ☐ 4
    - ☐ 3
    - ☐ 0

13. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction déclarée mais non définie
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction appelée avant sa déclaration
14. 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;`
15. Vous utilisez une boucle `while` quand :
    - ☐ vous avez déjà fait un `for` dans le même programme principal
    - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
    - ☐ l'incrément de la variable de boucle n'est pas 1
    - ☐ vous n'avez pas déclaré de fonction
16. Sur unix (ou linux), la commande `mkdir` permet de :
    - ☐ créer un répertoire
    - ☐ changer de répertoire courant
    - ☐ créer un fichier texte
    - ☐ ouvrir un fichier texte
17. 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
18. Le langage C est un langage
    - ☐ interprété
    - ☐ lu, écrit, parlé
    - ☐ composé
    - ☐ compilé
19. Le code suivant :

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

    affichera :
    - ☐ `Mineur`
`Majeur`
    - ☐ `Majeur`
    - ☐ `Mineur`
    - ☐ rien
20. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
    - ☐ `n = mccarthy();`
    - ☐ `x = mccarthy(n);`
    - ☐ `int mccarthy(int 2);`
    - ☐ `n = mccarthy(p, q);`

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 une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
   - ☐ `int afficher_menu();`
   - ☐ `int afficher_menu(int char);`
   - ☐ `void afficher_menu();`
   - ☐ `double afficher_menu();`
   - ☐ `char afficher_menu(printf("menu"));`
2. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

   Alors l'expression `f(1)` prendra la valeur :
   - ☐ 5
   - ☐ 0
   - ☐ 4
   - ☐ 1
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 fichier texte qui sera traduit en instructions processeur
   - ☐ un document qui doit être protégé
   - ☐ un document illisible pour les humains
4. 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 = 5`
   - ☐ `j = 4`
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 2
   - ☐ 8 6 4 2
   - ☐ 8 6 4 2 0
6. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses champs
   - ☐ ses chants
   - ☐ ses cases
   - ☐ ses blocs
7. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il comporte une boucle infinie
   - ☐ il ne compile pas
   - ☐ il risque d'afficher bonjour à la place de coucou
   - ☐ il n'affiche rien
8. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
   - ☐ `int k;`
   - ☐ `loop i;`
   - ☐ `int %d;`
   - ☐ `int loop n;`
9. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

   précède la fonction `main()`, alors on peut écrire en début de `main()` :
   - ☐ `int toto.n = 3;`
   - ☐ `struct toto_s toto;`
   - ☐ `int struct toto_s = {3, -1e10};`
   - ☐ `toto_s n, x;`
   - ☐ `toto_s struct z = {3, 0.5};`
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 sémantique
    - ☐ l'analyse harmonique
    - ☐ l'analyse des entrées clavier
    - ☐ l'édition de liens
11. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
    - ☐ `int carre(2);`
    - ☐ `int n = carre();`
    - ☐ `n = carre(int n);`
    - ☐ `n = carre(n);`

12. 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`
13. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
    - ☐ `saisie_utilisateur(scanf(%d));`
    - ☐ `void saisie_utilisateur(int n);`
    - ☐ `void saisie_utilisateur(char c);`
    - ☐ `int saisie_utilisateur();`
14. 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)
15. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 3
- ☐ 4
- ☐ 0

16. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
    - ☐ `int pgcd(int y, int x);`
    - ☐ `int pgcd(int x, y);`
    - ☐ `int pgcd(int x, int x);`
    - ☐ `void pgcd(int x, int y);`
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. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction déclarée mais non définie
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction appelée avant sa déclaration
19. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(a,b)=13, a=8, b=5`

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=1;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. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

☐ 0

☐ 3

☐ 4

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 = 5`

☐ le terminal affiche `x = 2`

☐ le terminal affiche "Faux"

☐ le terminal affiche `5`

3. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :

☐ `int struct toto_s = {3, -1e10};`

☐ `struct toto_s toto;`

☐ `int toto.n = 3;`

☐ `toto_s struct z = {3, 0.5};`

☐ `toto_s n, x;`

4. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :

☐ `struct int factorielle(int n);`

☐ `int factorielle(double n);`

☐ `int factorielle(int x);`

☐ `int factorielle();`

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 :

☐ 16

☐ 3

☐ 6

☐ 20

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 1 2 0 1 2`

☐ `0 0 1 1 2 2 3`

☐ `0 1 2 0 1 2 3`

7. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :

☐ `int afficher_menu(int char);`

☐ `int afficher_menu();`

☐ `double afficher_menu();`

☐ `void afficher_menu();`

☐ `char afficher_menu(printf("menu"));`

8. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

☐ 0

☐ 5

☐ 7

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 = %d`

☐ `j = 5`

☐ `j = 0`

10. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
    - ☐ `char tableau[5];`
    - ☐ `int toto[taille=5];`
    - ☐ `int toto[5];`
    - ☐ `int[] new tableau(5);`
    - ☐ `int tab[] = 5;`
11. 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
12. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
    - ☐ `n = factorielle();`
    - ☐ `int factorielle(int 2);`
    - ☐ `printf("%d", factorielle(n));`
    - ☐ `n = factorielle(p, q);`
13. 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>`
14. Un fichier source est :
    - ☐ 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
    - ☐ un fichier texte qui sera traduit en instructions processeur
15. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ dans lequel ces fonctions sont appelées dans le main
    - ☐ dans lequel vous avez déclaré ces fonction
    - ☐ un ordre quelconque
    - ☐ alphabétique
16. 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
17. 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 variable non déclarée
    - ☐ une faute de frappe dans un appel de fonction
    - ☐ une directive préprocesseur `#include` manquante
18. Vous utilisez une boucle `while` quand :
    - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
    - ☐ vous avez déjà fait un `for` dans le même programme principal
    - ☐ vous n'avez pas déclaré de fonction
    - ☐ l'incrément de la variable de boucle n'est pas 1
19. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses champs
    - ☐ ses chants
    - ☐ ses cases
    - ☐ ses blocs
20. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

    Alors l'affichage sera :
    - ☐ `i`
    - ☐ `ccccc`
    - ☐ `A`
    - ☐ `ABCDEF`

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 a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a == b`
- ☐ `a{n, x} == b{n, x}`
- ☐ `a = b`

2. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :

- ☐ `void afficher_date(date_s d);`
- ☐ `int afficher_date(date_s d);`
- ☐ `void afficher_date(struct date_s d);`
- ☐ `struct date_s afficher_date(struct date_s d);`

3. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :

- ☐ 0.5
- ☐ 1.5
- ☐ 0
- ☐ 1

4. 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

5. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :

- ☐ `x = racine(2/3);`
- ☐ `x = racine(racine(x)*racine(x));`
- ☐ `x = racine(x * x) - racine(x);`
- ☐ `x - 1 = racine(x);`

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. 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

8. Le code suivant :

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

affichera :

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

9. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

- ☐ alphabétique
- ☐ dans lequel ces fonctions sont appelées dans le main
- ☐ dans lequel vous avez déclaré ces fonction
- ☐ un ordre quelconque

10. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il comporte une boucle infinie
- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il n'affiche rien
- ☐ il ne compile pas

11. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 3
- ☐ 0
- ☐ 4

12. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

- ☐ `printf("%d", factorielle(n));`
- ☐ `n = factorielle();`
- ☐ `n = factorielle(p, q);`
- ☐ `int factorielle(int 2);`

13. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 4
- ☐ 0
- ☐ 1
- ☐ 5

14. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :

- ☐ `double exposant(double x, int n);`
- ☐ `exposant(double x, int n, int r);`
- ☐ `int exposant(double n, int x);`
- ☐ `void exposant(double x^n);`

15. 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`

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
- ☐ 8
- ☐ 4

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

18. 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>`

19. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

Alors l'affichage sera :

- ☐ `C`
- ☐ `b`
- ☐ `A`
- ☐ `B`

20. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 0
- ☐ 5
- ☐ 7

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

2. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :
   - ☐ 4
   - ☐ 0
   - ☐ 3

3. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   - ☐ 0
   - ☐ 7
   - ☐ 5

4. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
   - ☐ `n = carre(n);`
   - ☐ `int carre(2);`
   - ☐ `int n = carre();`
   - ☐ `n = carre(int n);`

5. 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
   - ☐ ne comportent aucune erreur
   - ☐ comportent une erreur qui sera détectée au cours de l'édition de lien

6. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
   - ☐ `!(!A || B) == (A && !B)`
   - ☐ `(A == TRUE) && (B == TRUE)`
   - ☐ `(!A || B)`
   - ☐ `A && B`

7. Le type des réels en C est :
   - ☐ `int`
   - ☐ `double`
   - ☐ `char`
   - ☐ `real`

8. 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

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

10. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
    - ☐ `int pgcd(int x, int x);`
    - ☐ `int pgcd(int x, y);`
    - ☐ `void pgcd(int x, int y);`
    - ☐ `int pgcd(int y, int x);`

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
    - ☐ 10
    - ☐ 6
    - ☐ 15

12. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction déclarée mais non définie
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une directive préprocesseur `#include` manquante

13. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

- ☐ `cccccc`
- ☐ `A`
- ☐ `i`
- ☐ `ABCDEF`

14. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
    - ☐ `int char;`
    - ☐ `char 'c';`
    - ☐ `char "c";`
    - ☐ `char c;`
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
    - ☐ une variable non déclarée
    - ☐ une faute de frappe dans un appel de fonction
    - ☐ un caractère interdit en C
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. Une variable booléenne est un variable :
    - ☐ jamais nulle
    - ☐ réelle positive
    - ☐ à laquelle une valeur vient d'être affectée
    - ☐ qui est vraie ou fausse
    - ☐ NaN (not a number, qui n'est pas un nombre)
18. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
    - ☐ `int factorielle(int 2);`
    - ☐ `n = factorielle(p, q);`
    - ☐ `n = factorielle();`
    - ☐ `printf("%d", factorielle(n));`
19. 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
- ☐ 4 3 2 1 0

20. 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

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 type des réels en C est :
   - ☐ `int`
   - ☐ `real`
   - ☐ `double`
   - ☐ `char`
2. Un fichier source est :
   - ☐ 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
   - ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur
3. Vous utilisez une boucle `while` quand :
   - ☐ vous n'avez pas déclaré de fonction
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous avez déjà fait un `for` dans le même programme principal
4. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
   - ☐ `double afficher_menu();`
   - ☐ `int afficher_menu();`
   - ☐ `int afficher_menu(int char);`
   - ☐ `char afficher_menu(printf("menu"));`
   - ☐ `void afficher_menu();`
5. Le code suivant :

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

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

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

   affichera :
   - ☐ `Majeur`
   - ☐ rien
   - ☐ `Mineur` `Majeur`
   - ☐ `Mineur`
7. 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
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. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ alphabétique
   - ☐ dans lequel ces fonctions sont appelées dans le main
   - ☐ un ordre quelconque
   - ☐ dans lequel vous avez déclaré ces fonction
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 `****`
    - ☐ le programme affiche x
    - ☐ la variable x vaut $-\frac{1}{2}$
    - ☐ la variable x vaut 16
11. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

    précède la fonction `main()`, alors on peut écrire en début de `main()` :
    - ☐ `toto_s n, x;`
    - ☐ `toto_s struct z = {3, 0.5};`
    - ☐ `int struct toto_s = {3, -1e10};`
    - ☐ `struct toto_s toto;`
    - ☐ `int toto.n = 3;`

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

13. 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

14. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a{n, x} == b{n, x}`
- ☐ `a == b`
- ☐ `a = b`

15. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 0
- ☐ 4
- ☐ 5
- ☐ 1

16. Si cette erreur apparaît à la compilation :
`erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?

- ☐ une fonction appelée avant sa déclaration
- ☐ une directive préprocesseur `#include` manquante
- ☐ une fonction déclarée mais non définie
- ☐ un désaccord entre la déclaration et la définition d'une fonction

17. L'écriture 111 en binaire correspond au nombre naturel :

- ☐ 7
- ☐ 8
- ☐ 3
- ☐ 111

18. Un enregistrement permet de grouper plusieurs valeurs dans :

- ☐ ses cases
- ☐ ses chants
- ☐ ses champs
- ☐ ses blocs

19. 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

20. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

- ☐ `printf("%d", factorielle(n));`
- ☐ `n = factorielle(p, q);`
- ☐ `int factorielle(int 2);`
- ☐ `n = factorielle();`

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

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
   ☐ 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

3. 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

4. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
   ☐ `int pgcd(int x, int x);`
   ☐ `void pgcd(int x, int y);`
   ☐ `int pgcd(int y, int x);`
   ☐ `int pgcd(int x, y);`

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

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
   ☐ le terminal affiche 8
   ☐ la case mémoire 8 contiendra 16

7. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   ☐ 5
   ☐ 7
   ☐ 0

8. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   ☐ il ne compile pas
   ☐ il n'affiche rien
   ☐ il comporte une boucle infinie
   ☐ il risque d'afficher bonjour à la place de coucou

9. 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

10. Le code suivant :

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

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

11. 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

12. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

    appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

    L'affichage dans le main est le suivant :
    - ☐ `f(a,b)=8, a=8, b=5`
    - ☐ `f(3,5)=8, a=3, b=5`
    - ☐ `f(a,b)=8, a=3, b=5`
    - ☐ `f(a,b)=13, a=8, b=5`

13. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses blocs
    - ☐ ses champs
    - ☐ ses cases
    - ☐ ses chants

14. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
    - ☐ `int factorielle(int x);`
    - ☐ `int factorielle(double n);`
    - ☐ `int factorielle();`
    - ☐ `struct int factorielle(int n);`

15. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
    - ☐ `n = mccarthy();`
    - ☐ `x = mccarthy(n);`
    - ☐ `n = mccarthy(p, q);`
    - ☐ `int mccarthy(int 2);`

16. 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`

17. Si cette erreur apparaît à la compilation : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
    - ☐ une fonction déclarée mais non définie
    - ☐ une fonction appelée avant sa déclaration
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une directive préprocesseur `#include` manquante

18. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ dans lequel ces fonctions sont appelées dans le main
    - ☐ alphabétique
    - ☐ un ordre quelconque
    - ☐ dans lequel vous avez déclaré ces fonction

19. 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`

20. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
    - ☐ `int saisie_utilisateur();`
    - ☐ `void saisie_utilisateur(int n);`
    - ☐ `saisie_utilisateur(scanf(%d));`
    - ☐ `void saisie_utilisateur(char 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
   - ☐ un chiffre binaire (0 ou 1)
   - ☐ l'instruction qui met fin à un programme

2. 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

3. 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>`

4. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
   - ☐ `char "c";`
   - ☐ `char c;`
   - ☐ `int char;`
   - ☐ `char 'c';`

5. 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

6. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il comporte une boucle infinie
   - ☐ il ne compile pas
   - ☐ il n'affiche rien
   - ☐ il risque d'afficher bonjour à la place de coucou

7. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
   - ☐ 8
   - ☐ 7
   - ☐ 3
   - ☐ 111

8. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   - ☐ 7
   - ☐ 5
   - ☐ 0

9. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
   - ☐ `scanf("%d", &n);`
   - ☐ `printf("Valeur de n ? %d\n", n);`
   - ☐ `printf("Valeur de n ? %g\n", n);`
   - ☐ un débogueur

10. Avant de faire appel à une fonction il est nécessaire de :
    - ☐ l'avoir définie
    - ☐ l'avoir déclarée
    - ☐ avoir défini une constante symbolique de la taille de cette fonction
    - ☐ l'avoir déclarée et définie

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

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

13. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses cases
    - ☐ ses blocs
    - ☐ ses chants
    - ☐ ses champs

14. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
    - ☐ `int factorielle(int 2);`
    - ☐ `n = factorielle();`
    - ☐ `printf("%d", factorielle(n));`
    - ☐ `n = factorielle(p, q);`

15. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

    Alors l'expression `f(0)` prendra la valeur :
    - ☐ 0
    - ☐ 3
    - ☐ 4

16. 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

17. 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

18. 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
    - ☐ retourner un bloc
    - ☐ mettre les blocs en séquence les uns à la suite des autres

19. 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`

20. Si cette erreur apparaît à la compilation :
    `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction déclarée mais non définie
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction appelée avant sa déclaration

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 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
   - ☐ un document de référence du système

2. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   - ☐ 7
   - ☐ 5
   - ☐ 0

3. 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)

4. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
   - ☐ `n = mccarthy();`
   - ☐ `n = mccarthy(p, q);`
   - ☐ `int mccarthy(int 2);`
   - ☐ `x = mccarthy(n);`

5. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

   Alors l'affichage sera :
   - ☐ `A`
   - ☐ `cccccc`
   - ☐ `ABCDEF`
   - ☐ `i`

6. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
   - ☐ `char c;`
   - ☐ `char "c";`
   - ☐ `int char;`
   - ☐ `char 'c';`

7. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
   - ☐ `struct int factorielle(int n);`
   - ☐ `int factorielle(double n);`
   - ☐ `int factorielle(int x);`
   - ☐ `int factorielle();`

8. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
   - ☐ un débogueur
   - ☐ `scanf("%d", &n);`
   - ☐ `printf("Valeur de n ? %d\n", n);`
   - ☐ `printf("Valeur de n ? %g\n", n);`

9. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
   - ☐ `void saisie_utilisateur(int n);`
   - ☐ `int saisie_utilisateur();`
   - ☐ `void saisie_utilisateur(char c);`
   - ☐ `saisie_utilisateur(scanf(%d));`

10. Si cette erreur apparaît à la compilation : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction déclarée mais non définie
    - ☐ un désaccord entre la déclaration et la définition d'une fonction

11. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

    Alors pour tester l'égalité de a et de b on utilise la condition :
    - ☐ `a == b`
    - ☐ `a{n, x} == b{n, x}`
    - ☐ `a = b`
    - ☐ `(a.n == b.n) && (a.x == b.x)`

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=%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");`

13. 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`

14. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(3,5)=8, a=3, b=5`

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

- ☐ `<>`
- ☐ `!`
- ☐ `!=`
- ☐ $\neq$

16. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il ne compile pas
- ☐ il n'affiche rien
- ☐ il comporte une boucle infinie
- ☐ il risque d'afficher bonjour à la place de coucou

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

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

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

19. Le code suivant :

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

affichera :

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

20. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :

- ☐ `int pgcd(int x, y);`
- ☐ `void pgcd(int x, int y);`
- ☐ `int pgcd(int y, int x);`
- ☐ `int pgcd(int x, int x);`

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. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
   - ☐ `A && B`
   - ☐ `(A == TRUE) && (B == TRUE)`
   - ☐ `!(!A || B) == (A && !B)`
   - ☐ `(!A || B)`
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 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`
3. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

   Alors l'affichage sera :
   - ☐ `C`
   - ☐ `A`
   - ☐ `b`
   - ☐ `B`
4. Vous utilisez une boucle `while` quand :
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous n'avez pas déclaré de fonction
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ vous avez déjà fait un `for` dans le même programme principal
5. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

   précède la fonction `main()`, alors on peut écrire en début de `main()` :
   - ☐ `int toto.n = 3;`
   - ☐ `toto_s struct z = {3, 0.5};`
   - ☐ `int struct toto_s = {3, -1e10};`
   - ☐ `struct toto_s toto;`
   - ☐ `toto_s n, x;`
6. Le code suivant :

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

   affichera :
   - ☐ `Mineur`
   - ☐ rien
   - ☐ `Majeur`
   - ☐ `Mineur`
     `Majeur`
7. Avant de faire appel à une fonction il est nécessaire de :
   - ☐ l'avoir définie
   - ☐ avoir défini une constante symbolique de la taille de cette fonction
   - ☐ l'avoir déclarée et définie
   - ☐ l'avoir déclarée
8. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function 'max'` , que doit-on chercher dans le programme ?
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une fonction déclarée mais non définie
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une directive préprocesseur `#include` manquante
9. 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
10. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
    - ☐ `int n = carre();`
    - ☐ `n = carre(int n);`
    - ☐ `int carre(2);`
    - ☐ `n = carre(n);`
11. Le code suivant :

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

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

12. Une variable booléenne est un variable :
    - ☐ réelle positive
    - ☐ jamais nulle
    - ☐ qui est vraie ou fausse
    - ☐ NaN (not a number, qui n'est pas un nombre)
    - ☐ à laquelle une valeur vient d'être affectée

13. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

    appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

    L'affichage dans le main est le suivant :
    - ☐ `f(a,b)=8, a=8, b=5`
    - ☐ `f(a,b)=13, a=8, b=5`
    - ☐ `f(a,b)=8, a=3, b=5`
    - ☐ `f(3,5)=8, a=3, b=5`

14. 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
    - ☐ 4 3 2 1 0

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

16. 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

17. 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

18. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que `x` est une variable réelle définie et initialisée, il est incorrect d'écrire :
    - ☐ `x - 1 = racine(x);`
    - ☐ `x = racine(racine(x)*racine(x));`
    - ☐ `x = racine(2/3);`
    - ☐ `x = racine(x * x) - racine(x);`

19. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

    Alors l'expression `g(0)` prendra la valeur :
    - ☐ 0
    - ☐ 5
    - ☐ 7

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

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 `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
   - ☐ `int factorielle(int 2);`
   - ☐ `n = factorielle();`
   - ☐ `n = factorielle(p, q);`
   - ☐ `printf("%d", factorielle(n));`

2. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   - ☐ 7
   - ☐ 5
   - ☐ 0

3. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

   Alors l'expression `f(1)` prendra la valeur :
   - ☐ 0
   - ☐ 4
   - ☐ 5
   - ☐ 1

4. 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`

5. 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

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

7. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il ne compile pas
   - ☐ il risque d'afficher bonjour à la place de coucou
   - ☐ il n'affiche rien
   - ☐ il comporte une boucle infinie

8. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
   - ☐ `A && B`
   - ☐ `(!A || B)`
   - ☐ `!(!A || B) == (A && !B)`
   - ☐ `(A == TRUE) && (B == TRUE)`

9. Le code suivant :

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

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

10. 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

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

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

12. L'écriture 101 en binaire correspond au nombre naturel :

- ☐ 5
- ☐ 4
- ☐ 3
- ☐ 101

13. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

- ☐ un ordre quelconque
- ☐ dans lequel ces fonctions sont appelées dans le main
- ☐ dans lequel vous avez déclaré ces fonction
- ☐ alphabétique

14. Un enregistrement permet de grouper plusieurs valeurs dans :

- ☐ ses blocs
- ☐ ses chants
- ☐ ses champs
- ☐ ses cases

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
- ☐ 0
- ☐ 6
- ☐ 1

16. 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
- ☐ comportent une erreur qui sera détectée au cours de l'analyse syntaxique
- ☐ ne comportent aucune erreur

17. 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`

18. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :

- ☐ un débogueur
- ☐ `printf("Valeur de n ? %d\n", n);`
- ☐ `printf("Valeur de n ? %g\n", n);`
- ☐ `scanf("%d", &n);`

19. 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

20. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

- ☐ `A`
- ☐ `cccccc`
- ☐ `ABCDEF`
- ☐ `i`

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 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
2. Pour déclarer une variable qui sera utilisée comme variable de boucle on peut utiliser l'instruction
   - ☐ `int loop n;`
   - ☐ `int %d;`
   - ☐ `int k;`
   - ☐ `loop i;`
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. 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`
5. 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
6. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   - ☐ `f(a,b)=8, a=8, b=5`
   - ☐ `f(a,b)=8, a=3, b=5`
   - ☐ `f(3,5)=8, a=3, b=5`
   - ☐ `f(a,b)=13, a=8, b=5`
7. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
   - ☐ 7
   - ☐ 3
   - ☐ 8
   - ☐ 111
8. 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
9. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
   - ☐ `(A == TRUE) && (B == TRUE)`
   - ☐ `(!A || B)`
   - ☐ `A && B`
   - ☐ `!(!A || B) == (A && !B)`
10. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
    - ☐ `n = carre(int n);`
    - ☐ `int carre(2);`
    - ☐ `n = carre(n);`
    - ☐ `int n = carre();`
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
    - ☐ 8
    - ☐ 4
    - ☐ 16
12. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
    - ☐ `printf("%d", factorielle(n));`
    - ☐ `n = factorielle();`
    - ☐ `n = factorielle(p, q);`
    - ☐ `int factorielle(int 2);`

13. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 4
- ☐ 0
- ☐ 3

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. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 4
- ☐ 5
- ☐ 0
- ☐ 1

16. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :

- ☐ `void afficher_date(date_s d);`
- ☐ `struct date_s afficher_date(struct date_s d);`
- ☐ `void afficher_date(struct date_s d);`
- ☐ `int afficher_date(date_s d);`

17. 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)

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 = 5`
- ☐ `j = 4`
- ☐ `j = %d`

19. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

- ☐ `ABCDEF`
- ☐ `i`
- ☐ `cccccc`
- ☐ `A`

20. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :

- ☐ `int pgcd(int y, int x);`
- ☐ `int pgcd(int x, int x);`
- ☐ `void pgcd(int x, int y);`
- ☐ `int pgcd(int x, y);`

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 la fonction g définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

☐ 0
☐ 5
☐ 7

2. Le code suivant :

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

affichera :

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

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 = 0`
☐ `j = 4`
☐ `j = %d`

4. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?

☐ `(!A || B)`
☐ `A && B`
☐ `!(!A || B) == (A && !B)`
☐ `(A == TRUE) && (B == TRUE)`

5. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

☐ 3
☐ 4
☐ 0

6. 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`

7. 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

8. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :

☐ `void exposant(double x^n);`
☐ `double exposant(double x, int n);`
☐ `exposant(double x, int n, int r);`
☐ `int exposant(double n, int x);`

9. 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

10. Une variable booléenne est un variable :

☐ à laquelle une valeur vient d'être affectée
☐ jamais nulle
☐ réelle positive
☐ NaN (not a number, qui n'est pas un nombre)
☐ qui est vraie ou fausse

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 déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :

- ☐ `void afficher_date(struct date_s d);`
- ☐ `void afficher_date(date_s d);`
- ☐ `int afficher_date(date_s d);`
- ☐ `struct date_s afficher_date(struct date_s d);`

13. Le type des réels en C est :

- ☐ `real`
- ☐ `char`
- ☐ `double`
- ☐ `int`

14. Vous utilisez une boucle `while` quand :

- ☐ vous avez déjà fait un `for` dans le même programme principal
- ☐ l'incrément de la variable de boucle n'est pas 1
- ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
- ☐ vous n'avez pas déclaré de fonction

15. 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

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

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

17. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

- ☐ `int factorielle(int 2);`
- ☐ `n = factorielle();`
- ☐ `printf("%d", factorielle(n));`
- ☐ `n = factorielle(p, q);`

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. 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`

20. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(3,5)=8, a=3, b=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. 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

2. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :

   - ☐ `f(a,b)=13, a=8, b=5`
   - ☐ `f(a,b)=8, a=8, b=5`
   - ☐ `f(a,b)=8, a=3, b=5`
   - ☐ `f(3,5)=8, a=3, b=5`

3. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
   - ☐ `void saisie_utilisateur(int n);`
   - ☐ `saisie_utilisateur(scanf(%d));`
   - ☐ `int saisie_utilisateur();`
   - ☐ `void saisie_utilisateur(char c);`

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

5. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

   Alors pour tester l'égalité de a et de b on utilise la condition :
   - ☐ `a{n, x} == b{n, x}`
   - ☐ `(a.n == b.n) && (a.x == b.x)`
   - ☐ `a == b`
   - ☐ `a = b`

6. Le code suivant :

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

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

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. 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

9. Vous utilisez une boucle `while` quand :
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ vous avez déjà fait un `for` dans le même programme principal
   - ☐ vous n'avez pas déclaré de fonction

10. 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 0
- ☐ 4 3 2 1

11. 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
12. Avant de faire appel à une fonction il est nécessaire de :
    - ☐ l'avoir définie
    - ☐ l'avoir déclarée
    - ☐ avoir défini une constante symbolique de la taille de cette fonction
    - ☐ l'avoir déclarée et définie
13. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :

- ☐ `struct toto_s toto;`
- ☐ `toto_s n, x;`
- ☐ `int struct toto_s = {3, -1e10};`
- ☐ `toto_s struct z = {3, 0.5};`
- ☐ `int toto.n = 3;`

14. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

- ☐ `ABCDEF`
- ☐ `i`
- ☐ `cccccc`
- ☐ `A`

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 = 0`
- ☐ `j = 4`

16. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 3
- ☐ 4
- ☐ 0

17. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
    - ☐ `int factorielle(int x);`
    - ☐ `struct int factorielle(int n);`
    - ☐ `int factorielle();`
    - ☐ `int factorielle(double n);`
18. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
    - ☐ `n = mccarthy();`
    - ☐ `x = mccarthy(n);`
    - ☐ `int mccarthy(int 2);`
    - ☐ `n = mccarthy(p, q);`
19. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
    - ☐ `A && B`
    - ☐ `!(!A || B) == (A && !B)`
    - ☐ `(A == TRUE) && (B == TRUE)`
    - ☐ `(!A || B)`
20. 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

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 en trop
   - ☐ un point-virgule manquant
   - ☐ une accolade en trop
   - ☐ une accolade manquante
2. Quel est l'opérateur de différence en C :
   - ☐ `!=`
   - ☐ `<>`
   - ☐ `!`
   - ☐ $\neq$
3. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse sémantique
   - ☐ analyse harmonique
   - ☐ analyse lexicale
   - ☐ analyse syntaxique
4. 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
5. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
   - ☐ `int carre(2);`
   - ☐ `n = carre(int n);`
   - ☐ `int n = carre();`
   - ☐ `n = carre(n);`
6. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
   - ☐ `n = factorielle();`
   - ☐ `n = factorielle(p, q);`
   - ☐ `printf("%d", factorielle(n));`
   - ☐ `int factorielle(int 2);`
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 x
   - ☐ la variable x vaut 16
   - ☐ le programme affiche `****`
   - ☐ la variable x vaut $-\frac{1}{2}$
8. 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
9. Quel est le problème d'un programme comportant les lignes suivantes ?
   ```
   while (1)
   {
     printf("coucou\n");
   }
   ```
   - ☐ il comporte une boucle infinie
   - ☐ il n'affiche rien
   - ☐ il ne compile pas
   - ☐ il risque d'afficher bonjour à la place de coucou
10. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ un ordre quelconque
    - ☐ dans lequel ces fonctions sont appelées dans le main
    - ☐ alphabétique
    - ☐ dans lequel vous avez déclaré ces fonction
11. Soit la fonction `g` définie par :
    ```
    int g(int a)
    {
      printf("a = \n", %d);
      if (1 > 0)
      {
        return 5;
      }
      return 7;
    }
    ```
    Alors l'expression `g(0)` prendra la valeur :
    - ☐ 7
    - ☐ 5
    - ☐ 0
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 = 5`
- ☐ `j = 0`
- ☐ `j = %d`
- ☐ `j = 4`

13. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 0
- ☐ 3
- ☐ 4

14. Avant de faire appel à une fonction il est nécessaire de :

- ☐ l'avoir déclarée et définie
- ☐ avoir défini une constante symbolique de la taille de cette fonction
- ☐ l'avoir définie
- ☐ l'avoir déclarée

15. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?

- ☐ `(A == TRUE) && (B == TRUE)`
- ☐ `!(!A || B) == (A && !B)`
- ☐ `A && B`
- ☐ `(!A || B)`

16. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :

- ☐ `int pgcd(int x, int x);`
- ☐ `int pgcd(int x, y);`
- ☐ `int pgcd(int y, int x);`
- ☐ `void pgcd(int x, int y);`

17. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :

- ☐ `printf("Valeur de n ? %d\n", n);`
- ☐ `printf("Valeur de n ? %g\n", n);`
- ☐ `scanf("%d", &n);`
- ☐ un débogueur

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
- ☐ 4 3 2 1
- ☐ 1 2 3 4

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

20. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :

- ☐ `int factorielle();`
- ☐ `int factorielle(double n);`
- ☐ `struct int factorielle(int n);`
- ☐ `int factorielle(int x);`

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. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ dans lequel vous avez déclaré ces fonction
   - ☐ un ordre quelconque
   - ☐ alphabétique
   - ☐ dans lequel ces fonctions sont appelées dans le main

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

3. L'écriture 101 en binaire correspond au nombre naturel :
   - ☐ 3
   - ☐ 5
   - ☐ 101
   - ☐ 4

4. 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`

5. 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
   - ☐ 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

6. 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

7. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
   - ☐ `x - 1 = racine(x);`
   - ☐ `x = racine(x * x) - racine(x);`
   - ☐ `x = racine(2/3);`
   - ☐ `x = racine(racine(x)*racine(x));`

8. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses champs
   - ☐ ses chants
   - ☐ ses blocs
   - ☐ ses cases

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
   - ☐ 0 1 2 3 4

10. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

    Alors l'expression `g(0)` prendra la valeur :
    - ☐ 7
    - ☐ 0
    - ☐ 5

11. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
    - ☐ `saisie_utilisateur(scanf(%d));`
    - ☐ `void saisie_utilisateur(int n);`
    - ☐ `void saisie_utilisateur(char c);`
    - ☐ `int saisie_utilisateur();`

12. 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 `****`

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
- ☐ 6
- ☐ 0
- ☐ 15

14. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `(a<n) || (n>b)`
- ☐ `(n<=a) && (n<=b)`
- ☐ `(a<=n) && (n<=b)`
- ☐ `a<=n<=b`

15. Vous utilisez une boucle `while` quand :
    - ☐ vous avez déjà fait un `for` dans le même programme principal
    - ☐ vous n'avez pas déclaré de fonction
    - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
    - ☐ l'incrément de la variable de boucle n'est pas 1

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 0
- ☐ 8 2
- ☐ 8 6 4 2

17. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
    - ☐ `int factorielle(double n);`
    - ☐ `int factorielle();`
    - ☐ `int factorielle(int x);`
    - ☐ `struct int factorielle(int n);`

18. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
    - ☐ `x = mccarthy(n);`
    - ☐ `int mccarthy(int 2);`
    - ☐ `n = mccarthy();`
    - ☐ `n = mccarthy(p, q);`

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

20. 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

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 `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :
   - ☐ `int pgcd(2);`
   - ☐ `int n = pgcd();`
   - ☐ `n = pgcd(n, 3);`
   - ☐ `n = pgcd(int p, int q);`
2. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

   Alors l'affichage sera :
   - ☐ `ABCDEF`
   - ☐ `i`
   - ☐ `A`
   - ☐ `cccccc`
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
   - ☐ les fichiers du disque
   - ☐ certaines données de la mémoire de travail
   - ☐ des processus
4. Quel est l'opérateur de différence en C :
   - ☐ `!`
   - ☐ $\neq$
   - ☐ `!=`
   - ☐ `<>`
5. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses chants
   - ☐ ses cases
   - ☐ ses blocs
   - ☐ ses champs
6. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
   - ☐ `x = racine(racine(x)*racine(x));`
   - ☐ `x = racine(x * x) - racine(x);`
   - ☐ `x = racine(2/3);`
   - ☐ `x - 1 = racine(x);`
7. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :
   - ☐ 3
   - ☐ 0
   - ☐ 4
8. 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>`
9. 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
10. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

    Alors l'expression `f(1)` prendra la valeur :
    - ☐ 5
    - ☐ 0
    - ☐ 4
    - ☐ 1
11. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
    - ☐ 111
    - ☐ 8
    - ☐ 3
    - ☐ 7
12. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ dans lequel ces fonctions sont appelées dans le main
    - ☐ un ordre quelconque
    - ☐ dans lequel vous avez déclaré ces fonction
    - ☐ alphabétique
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=%d et y=%d\n",x y);`
    - ☐ `printf("x=%x et y=%y\n");`

14. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :

- ☐ `int struct toto_s = {3, -1e10};`
- ☐ `toto_s n, x;`
- ☐ `toto_s struct z = {3, 0.5};`
- ☐ `int toto.n = 3;`
- ☐ `struct toto_s toto;`

15. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

- ☐ `int char;`
- ☐ `char "c";`
- ☐ `char 'c';`
- ☐ `char c;`

16. Si cette erreur apparaît à la compilation :
`erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?

- ☐ une directive préprocesseur `#include` manquante
- ☐ une fonction appelée avant sa déclaration
- ☐ un désaccord entre la déclaration et la définition d'une fonction
- ☐ une fonction déclarée mais non définie

17. 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
- ☐ 6
- ☐ 42

18. Le code suivant :

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

affichera :

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

19. 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

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 en trop
- ☐ une accolade manquante

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 type des réels en C est :
   - ☐ `double`
   - ☐ `char`
   - ☐ `real`
   - ☐ `int`

2. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ dans lequel vous avez déclaré ces fonction
   - ☐ alphabétique
   - ☐ dans lequel ces fonctions sont appelées dans le main
   - ☐ un ordre quelconque

3. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses champs
   - ☐ ses blocs
   - ☐ ses cases
   - ☐ ses chants

4. L'écriture 101 en binaire correspond au nombre naturel :
   - ☐ 3
   - ☐ 101
   - ☐ 4
   - ☐ 5

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

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

7. 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

8. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
   - ☐ 0
   - ☐ 1.5
   - ☐ 1
   - ☐ 0.5

9. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse lexicale
   - ☐ analyse harmonique
   - ☐ 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. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
    - ☐ `int pgcd(int x, y);`
    - ☐ `int pgcd(int x, int x);`
    - ☐ `void pgcd(int x, int y);`
    - ☐ `int pgcd(int y, int x);`

12. 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

13. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

    Alors l'affichage sera :
    - ☐ `i`
    - ☐ `ABCDEF`
    - ☐ `A`
    - ☐ `cccccc`

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. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une fonction déclarée mais non définie
    - ☐ un désaccord entre la déclaration et la définition d'une fonction

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

17. 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)

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 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`

19. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :

- ☐ `n = mccarthy();`
- ☐ `x = mccarthy(n);`
- ☐ `int mccarthy(int 2);`
- ☐ `n = mccarthy(p, q);`

20. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :

- ☐ `int n = carre();`
- ☐ `n = carre(n);`
- ☐ `int carre(2);`
- ☐ `n = carre(int n);`

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 le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   ☐ il comporte une boucle infinie
   ☐ il n'affiche rien
   ☐ il ne compile pas
   ☐ il risque d'afficher bonjour à la place de coucou

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

3. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
   ☐ `int factorielle(int 2);`
   ☐ `printf("%d", factorielle(n));`
   ☐ `n = factorielle();`
   ☐ `n = factorielle(p, q);`

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. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
   ☐ `n = mccarthy();`
   ☐ `int mccarthy(int 2);`
   ☐ `n = mccarthy(p, q);`
   ☐ `x = mccarthy(n);`

6. Un enregistrement permet de grouper plusieurs valeurs dans :
   ☐ ses cases
   ☐ ses chants
   ☐ ses champs
   ☐ ses blocs

7. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
   ☐ `int carre(2);`
   ☐ `n = carre(int n);`
   ☐ `int n = carre();`
   ☐ `n = carre(n);`

8. 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`

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

10. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    ☐ dans lequel vous avez déclaré ces fonction
    ☐ un ordre quelconque
    ☐ alphabétique
    ☐ dans lequel ces fonctions sont appelées dans le main

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
    ☐ 8
    ☐ 0
    ☐ 16

12. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
    ☐ `x = racine(2/3);`
    ☐ `x = racine(x * x) - racine(x);`
    ☐ `x = racine(racine(x)*racine(x));`
    ☐ `x - 1 = racine(x);`

13. Soit la fonction g définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 7
- ☐ 0
- ☐ 5

14. Le code suivant :

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

affichera :

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

15. L'écriture 101 en binaire correspond au nombre naturel :

- ☐ 5
- ☐ 4
- ☐ 3
- ☐ 101

16. 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`

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

18. Si cet avertissement apparaît à la compilation :
`warning: implicit declaration of function ‘max’`
, que doit-on chercher dans le programme ?

- ☐ une directive préprocesseur `#include` manquante
- ☐ un désaccord entre la déclaration et la définition d'une fonction
- ☐ une fonction déclarée mais non définie
- ☐ une fonction appelée avant sa déclaration

19. 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

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 3 2 1 0
- ☐ 0 1 2 3 4
- ☐ 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. 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
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 = %d`
   - ☐ `j = 4`
   - ☐ `j = 0`
3. 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
   - ☐ ne comportent aucune erreur
   - ☐ comportent une erreur qui ne sera pas détectée
4. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ dans lequel vous avez déclaré ces fonction
   - ☐ alphabétique
   - ☐ un ordre quelconque
   - ☐ dans lequel ces fonctions sont appelées dans le main
5. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
   - ☐ `printf("%d", factorielle(n));`
   - ☐ `n = factorielle(p, q);`
   - ☐ `int factorielle(int 2);`
   - ☐ `n = factorielle();`
6. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   - ☐ 5
   - ☐ 0
   - ☐ 7
7. Le code suivant :

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

   affichera :
   - ☐ `Mineur`
   - ☐ rien
   - ☐ `Majeur`
   - ☐ `Mineur`
     `Majeur`
8. Un fichier source est :
   - ☐ 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
   - ☐ un fichier texte qui sera traduit en instructions processeur
9. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse harmonique
   - ☐ analyse syntaxique
   - ☐ analyse sémantique
   - ☐ analyse lexicale
10. 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
11. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `(a<n) || (n>b)`
- ☐ `a<=n<=b`
- ☐ `(n<=a) && (n<=b)`
- ☐ `(a<=n) && (n<=b)`

12. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
    - ☐ `void saisie_utilisateur(int n);`
    - ☐ `int saisie_utilisateur();`
    - ☐ `void saisie_utilisateur(char c);`
    - ☐ `saisie_utilisateur(scanf(%d));`
13. L'écriture 111 en binaire correspond au nombre naturel :
    - ☐ 7
    - ☐ 3
    - ☐ 111
    - ☐ 8
14. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

- ☐ `A`
- ☐ `ccccc`
- ☐ `ABCDEF`
- ☐ `i`

15. Un programme en langage C doit comporter une et une seule définition de la fonction :
    - ☐ main
    - ☐ init
    - ☐ include
    - ☐ begin
16. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

Alors l'affichage sera :

- ☐ `C`
- ☐ `B`
- ☐ `b`
- ☐ `A`

17. Vous utilisez une boucle `while` quand :
    - ☐ l'incrément de la variable de boucle n'est pas 1
    - ☐ vous n'avez pas déclaré de fonction
    - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
    - ☐ vous avez déjà fait un `for` dans le même programme principal
18. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
    - ☐ `scanf("%d", &n);`
    - ☐ un débogueur
    - ☐ `printf("Valeur de n ? %d\n", n);`
    - ☐ `printf("Valeur de n ? %g\n", n);`
19. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function 'max'` , que doit-on chercher dans le programme ?
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction déclarée mais non définie
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
- ☐ 4 3 2 1
- ☐ 0 1 2 3
- ☐ 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. 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
2. Vous utilisez une boucle `while` quand :
   - ☐ vous n'avez pas déclaré de fonction
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous avez déjà fait un `for` dans le même programme principal
3. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il risque d'afficher bonjour à la place de coucou
   - ☐ il ne compile pas
   - ☐ il n'affiche rien
   - ☐ il comporte une boucle infinie
4. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
   - ☐ `n = factorielle(p, q);`
   - ☐ `int factorielle(int 2);`
   - ☐ `printf("%d", factorielle(n));`
   - ☐ `n = factorielle();`
5. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
   - ☐ 1.5
   - ☐ 0.5
   - ☐ 0
   - ☐ 1
6. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
   - ☐ analyse syntaxique
   - ☐ analyse sémantique
   - ☐ analyse lexicale
   - ☐ analyse harmonique
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'édition de lien
   - ☐ comportent une erreur qui sera détectée au cours de l'analyse syntaxique
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
   - ☐ 0
   - ☐ 8
   - ☐ 4
9. Sur unix (ou linux), la commande `mkdir` permet de :
   - ☐ créer un fichier texte
   - ☐ ouvrir un fichier texte
   - ☐ créer un répertoire
   - ☐ changer de répertoire courant
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
    - ☐ 16
    - ☐ 0
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 0
    - ☐ le programme affiche "Faux"
    - ☐ la variable y vaut 5
    - ☐ la variable x vaut 5 et la variable y vaut 0
12. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

    Alors l'expression `f(0)` prendra la valeur :
    - ☐ 4
    - ☐ 3
    - ☐ 0

13. Un enregistrement permet de grouper plusieurs valeurs dans :

- ☐ ses chants
- ☐ ses cases
- ☐ ses blocs
- ☐ ses champs

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
- ☐ 4 3 2 1 0
- ☐ 0 1 2 3 4

15. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a{n, x} == b{n, x}`
- ☐ `a == b`
- ☐ `a = b`

16. Le code suivant :

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

affichera :

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

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. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :

- ☐ `int pgcd(int x, int x);`
- ☐ `void pgcd(int x, int y);`
- ☐ `int pgcd(int x, y);`
- ☐ `int pgcd(int y, int x);`

19. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :

- ☐ `n = mccarthy(p, q);`
- ☐ `int mccarthy(int 2);`
- ☐ `n = mccarthy();`
- ☐ `x = mccarthy(n);`

20. L'écriture 111 en binaire correspond au nombre naturel :

- ☐ 7
- ☐ 3
- ☐ 8
- ☐ 111

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. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

   Quelle est la condition `cond` :

   ☐ `(a<=n) && (n<=b)`

   ☐ `a<=n<=b`

   ☐ `(a<n) || (n>b)`

   ☐ `(n<=a) && (n<=b)`

2. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :

   ☐ `printf("Valeur de n ? %g\n", n);`

   ☐ un débogueur

   ☐ `scanf("%d", &n);`

   ☐ `printf("Valeur de n ? %d\n", n);`

3. Le type des réels en C est :

   ☐ `real`

   ☐ `int`

   ☐ `double`

   ☐ `char`

4. L'écriture 101 en binaire correspond au nombre naturel :

   ☐ 4

   ☐ 3

   ☐ 5

   ☐ 101

5. 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

6. L'écriture 111 en binaire correspond au nombre naturel :

   ☐ 7

   ☐ 3

   ☐ 8

   ☐ 111

7. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :

   ☐ 4

   ☐ 0

   ☐ 3

8. 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>`

9. Un enregistrement permet de grouper plusieurs valeurs dans :

   ☐ ses blocs

   ☐ ses chants

   ☐ ses champs

   ☐ ses cases

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. 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

12. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :

    ☐ `void exposant(double x^n);`

    ☐ `exposant(double x, int n, int r);`

    ☐ `int exposant(double n, int x);`

    ☐ `double exposant(double x, int n);`

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

    ☐ 20

    ☐ 16

    ☐ 6

14. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

- ☐ `A`
- ☐ `ABCDEF`
- ☐ `i`
- ☐ `cccccc`

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

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 ?

- ☐ `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`

17. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
    - ☐ `char "c";`
    - ☐ `char 'c';`
    - ☐ `char c;`
    - ☐ `int char;`

18. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ un ordre quelconque
    - ☐ dans lequel vous avez déclaré ces fonction
    - ☐ alphabétique
    - ☐ dans lequel ces fonctions sont appelées dans le main

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

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
- ☐ 8
- ☐ 0
- ☐ 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. 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 ?

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

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  }
```

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

3. Le code suivant :

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

affichera :

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

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. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :

- ☐ `x - 1 = racine(x);`
- ☐ `x = racine(racine(x)*racine(x));`
- ☐ `x = racine(x * x) - racine(x);`
- ☐ `x = racine(2/3);`

6. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :

- ☐ `int afficher_menu();`
- ☐ `char afficher_menu(printf("menu"));`
- ☐ `int afficher_menu(int char);`
- ☐ `double afficher_menu();`
- ☐ `void afficher_menu();`

7. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :

- ☐ 1.5
- ☐ 0.5
- ☐ 0
- ☐ 1

8. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=8, b=5`

9. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 5
- ☐ 7
- ☐ 0

10. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :

- ☐ `int pgcd(int x, int x);`
- ☐ `int pgcd(int y, int x);`
- ☐ `int pgcd(int x, y);`
- ☐ `void pgcd(int x, int y);`

11. 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 = 0`
- ☐ `j = 4`

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. 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

14. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses blocs
    - ☐ ses cases
    - ☐ ses champs
    - ☐ ses chants
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 :

- ☐ 0
- ☐ 16
- ☐ 8
- ☐ 4

16. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
    - ☐ `!(!A || B) == (A && !B)`
    - ☐ `(A == TRUE) && (B == TRUE)`
    - ☐ `A && B`
    - ☐ `(!A || B)`
17. 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
18. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
    - ☐ `void afficher_date(date_s d);`
    - ☐ `struct date_s afficher_date(struct date_s d);`
    - ☐ `int afficher_date(date_s d);`
    - ☐ `void afficher_date(struct date_s d);`
19. 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`
20. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
    - ☐ `void saisie_utilisateur(char c);`
    - ☐ `int saisie_utilisateur();`
    - ☐ `void saisie_utilisateur(int n);`
    - ☐ `saisie_utilisateur(scanf(%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. Avant de faire appel à une fonction il est nécessaire de :
   - ☐ l'avoir déclarée
   - ☐ avoir défini une constante symbolique de la taille de cette fonction
   - ☐ l'avoir définie
   - ☐ l'avoir déclarée et définie
2. Le langage C est un langage
   - ☐ interprété
   - ☐ compilé
   - ☐ lu, écrit, parlé
   - ☐ composé
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
   - ☐ 4 3 2 1
   - ☐ 4 3 2 1 0
   - ☐ 0 1 2 3
4. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
   - ☐ `x = racine(x * x) - racine(x);`
   - ☐ `x = racine(2/3);`
   - ☐ `x - 1 = racine(x);`
   - ☐ `x = racine(racine(x)*racine(x));`
5. Vous utilisez une boucle `while` quand :
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous avez déjà fait un `for` dans le même programme principal
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ vous n'avez pas déclaré de fonction
6. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :
   - ☐ `int n = pgcd();`
   - ☐ `int pgcd(2);`
   - ☐ `n = pgcd(int p, int q);`
   - ☐ `n = pgcd(n, 3);`
7. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
   - ☐ `void afficher_date(struct date_s d);`
   - ☐ `struct date_s afficher_date(struct date_s d);`
   - ☐ `int afficher_date(date_s d);`
   - ☐ `void afficher_date(date_s d);`
8. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

   Alors l'expression `f(1)` prendra la valeur :
   - ☐ 5
   - ☐ 0
   - ☐ 1
   - ☐ 4
9. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ main
   - ☐ include
   - ☐ init
   - ☐ begin
10. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

    Alors l'expression `g(0)` prendra la valeur :
    - ☐ 5
    - ☐ 7
    - ☐ 0
11. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
    - ☐ 0
    - ☐ 1
    - ☐ 1.5
    - ☐ 0.5
12. 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 fichier texte qui sera traduit en instructions processeur
    - ☐ un document illisible pour les humains

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

14. 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
- ☐ 16
- ☐ 4

15. 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`

16. Le code suivant :

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

affichera :

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

17. 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
- ☐ 10
- ☐ 6

18. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

Alors l'affichage sera :

- ☐ `b`
- ☐ `B`
- ☐ `C`
- ☐ `A`

19. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(a,b)=8, a=3, b=5`

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. 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

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

3. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
   - ☐ `void afficher_date(struct date_s d);`
   - ☐ `int afficher_date(date_s d);`
   - ☐ `void afficher_date(date_s d);`
   - ☐ `struct date_s afficher_date(struct date_s d);`

4. 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`

5. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses blocs
   - ☐ ses chants
   - ☐ ses champs
   - ☐ ses cases

6. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il n'affiche rien
   - ☐ il comporte une boucle infinie
   - ☐ il ne compile pas
   - ☐ il risque d'afficher bonjour à la place de coucou

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

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. Vous utilisez une boucle `while` quand :
   - ☐ l'incrément de la variable de boucle n'est pas 1
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ vous avez déjà fait un `for` dans le même programme principal
   - ☐ vous n'avez pas déclaré de fonction

10. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
    - ☐ `x = racine(2/3);`
    - ☐ `x - 1 = racine(x);`
    - ☐ `x = racine(racine(x)*racine(x));`
    - ☐ `x = racine(x * x) - racine(x);`

11. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

    Alors l'affichage sera :
    - ☐ `ABCDEF`
    - ☐ `i`
    - ☐ `cccccc`
    - ☐ `A`

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

13. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function 'max'` , que doit-on chercher dans le programme ?
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction déclarée mais non définie
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une directive préprocesseur `#include` manquante

14. L'écriture 101 en binaire correspond au nombre naturel :
    - ☐ 3
    - ☐ 5
    - ☐ 4
    - ☐ 101

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

16. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
    - ☐ `void saisie_utilisateur(char c);`
    - ☐ `void saisie_utilisateur(int n);`
    - ☐ `int saisie_utilisateur();`
    - ☐ `saisie_utilisateur(scanf(%d));`

17. Si cette erreur apparaît à la compilation :
    `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction déclarée mais non définie
    - ☐ une fonction appelée avant sa déclaration

18. Une variable booléenne est un variable :
    - ☐ qui est vraie ou fausse
    - ☐ NaN (not a number, qui n'est pas un nombre)
    - ☐ à laquelle une valeur vient d'être affectée
    - ☐ réelle positive
    - ☐ jamais nulle

19. 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);`

20. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
    - ☐ analyse sémantique
    - ☐ analyse syntaxique
    - ☐ analyse lexicale
    - ☐ 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. Si cet avertissement apparaît à la compilation :
`warning: implicit declaration of function ‘max’`
, que doit-on chercher dans le programme ?
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une fonction déclarée mais non définie
   - ☐ une fonction appelée avant sa déclaration

2. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   - ☐ `f(a,b)=8, a=3, b=5`
   - ☐ `f(a,b)=13, a=8, b=5`
   - ☐ `f(a,b)=8, a=8, b=5`
   - ☐ `f(3,5)=8, a=3, b=5`

3. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
   - ☐ `n = factorielle(p, q);`
   - ☐ `printf("%d", factorielle(n));`
   - ☐ `n = factorielle();`
   - ☐ `int factorielle(int 2);`

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 4
   - ☐ 0 1 2 3
   - ☐ 4 3 2 1

5. 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

6. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

   précède la fonction `main()`, alors on peut écrire en début de `main()` :
   - ☐ `int struct toto_s = {3, -1e10};`
   - ☐ `toto_s n, x;`
   - ☐ `int toto.n = 3;`
   - ☐ `toto_s struct z = {3, 0.5};`
   - ☐ `struct toto_s toto;`

7. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses blocs
   - ☐ ses champs
   - ☐ ses cases
   - ☐ ses chants

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. 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`

10. Si cette erreur apparaît à la compilation :
`erreur: conflicting types for ‘max’` , que doit-on chercher dans le programme ?
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction déclarée mais non définie
    - ☐ un désaccord entre la déclaration et la définition d'une fonction

11. 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 tab[] = 5;`
    - ☐ `int toto[taille=5];`

12. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :

- ☐ `x = racine(x * x) - racine(x);`
- ☐ `x - 1 = racine(x);`
- ☐ `x = racine(2/3);`
- ☐ `x = racine(racine(x)*racine(x));`

13. 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 document de référence du système
- ☐ un fichier texte qui sera traduit en instructions processeur
- ☐ un document qui doit être protégé

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

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

16. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :

- ☐ `int saisie_utilisateur();`
- ☐ `void saisie_utilisateur(int n);`
- ☐ `void saisie_utilisateur(char c);`
- ☐ `saisie_utilisateur(scanf(%d));`

17. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 5
- ☐ 0
- ☐ 7

18. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

- ☐ dans lequel ces fonctions sont appelées dans le main
- ☐ un ordre quelconque
- ☐ dans lequel vous avez déclaré ces fonction
- ☐ alphabétique

19. Le bus système sert à :

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

20. 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

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

2. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `a == b`
- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a = b`
- ☐ `a{n, x} == b{n, x}`

3. Une variable booléenne est un variable :

- ☐ jamais nulle
- ☐ réelle positive
- ☐ qui est vraie ou fausse
- ☐ NaN (not a number, qui n'est pas un nombre)
- ☐ à laquelle une valeur vient d'être affectée

4. L'écriture 101 en binaire correspond au nombre naturel :

- ☐ 3
- ☐ 5
- ☐ 4
- ☐ 101

5. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(a,b)=13, a=8, b=5`

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

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

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

8. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :

- ☐ `n = factorielle();`
- ☐ `printf("%d", factorielle(n));`
- ☐ `int factorielle(int 2);`
- ☐ `n = factorielle(p, q);`

9. 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

10. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 4
- ☐ 0
- ☐ 1
- ☐ 5

11. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 3
- ☐ 0
- ☐ 4

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

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

13. Le langage C est un langage

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

14. Le type des réels en C est :

- ☐ `real`
- ☐ `int`
- ☐ `double`
- ☐ `char`

15. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :

- ☐ `int factorielle();`
- ☐ `int factorielle(double n);`
- ☐ `struct int factorielle(int n);`
- ☐ `int factorielle(int x);`

16. 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

17. 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>`

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 = 5`
- ☐ `j = %d`
- ☐ `j = 4`

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
- ☐ 0
- ☐ 8

20. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :

- ☐ `n = carre(n);`
- ☐ `n = carre(int n);`
- ☐ `int carre(2);`
- ☐ `int n = carre();`

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

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. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

   Alors l'expression `f(1)` prendra la valeur :
   - ☐ 5
   - ☐ 4
   - ☐ 1
   - ☐ 0

4. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   - ☐ `f(a,b)=8, a=8, b=5`
   - ☐ `f(a,b)=13, a=8, b=5`
   - ☐ `f(3,5)=8, a=3, b=5`
   - ☐ `f(a,b)=8, a=3, b=5`

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. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
   - ☐ `int factorielle(double n);`
   - ☐ `struct int factorielle(int n);`
   - ☐ `int factorielle();`
   - ☐ `int factorielle(int x);`

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. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
   - ☐ `char 'c';`
   - ☐ `int char;`
   - ☐ `char "c";`
   - ☐ `char c;`

9. Le code suivant :

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

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

10. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
    - ☐ `struct date_s afficher_date(struct date_s d);`
    - ☐ `void afficher_date(date_s d);`
    - ☐ `void afficher_date(struct date_s d);`
    - ☐ `int afficher_date(date_s d);`

11. 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

12. Vous utilisez une boucle `while` quand :
    - ☐ vous avez déjà fait un `for` dans le même programme principal
    - ☐ vous n'avez pas déclaré de fonction
    - ☐ l'incrément de la variable de boucle n'est pas 1
    - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance

13. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

    - ☐ il comporte une boucle infinie
    - ☐ il n'affiche rien
    - ☐ il ne compile pas
    - ☐ il risque d'afficher bonjour à la place de coucou

14. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

    précède la fonction `main()`, alors on peut écrire en début de `main()` :
    - ☐ `int toto.n = 3;`
    - ☐ `struct toto_s toto;`
    - ☐ `toto_s struct z = {3, 0.5};`
    - ☐ `toto_s n, x;`
    - ☐ `int struct toto_s = {3, -1e10};`

15. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses chants
    - ☐ ses cases
    - ☐ ses blocs
    - ☐ ses champs

16. 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

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 ?
    - ☐ `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`

18. 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

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
    - ☐ mettre les blocs en séquence les uns à la suite des autres
    - ☐ répéter un bloc tant qu'une condition est vérifée

20. L'écriture $\underline{101}$ en binaire correspond au nombre naturel :
    - ☐ 101
    - ☐ 3
    - ☐ 4
    - ☐ 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 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 illisible pour les humains
   - ☐ un document qui doit être protégé
   - ☐ un document de référence du système
2. 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
3. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
   - ☐ `x = racine(x * x) - racine(x);`
   - ☐ `x = racine(racine(x)*racine(x));`
   - ☐ `x = racine(2/3);`
   - ☐ `x - 1 = racine(x);`
4. Un enregistrement permet de grouper plusieurs valeurs dans :
   - ☐ ses chants
   - ☐ ses cases
   - ☐ ses champs
   - ☐ ses blocs
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=%x et y=%y\n");`
   - ☐ `printf("x=%d et y=%d\n,x,y");`
   - ☐ `printf("x=%d et y=%d\n",x y);`
6. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
   - ☐ `int pgcd(int x, y);`
   - ☐ `void pgcd(int x, int y);`
   - ☐ `int pgcd(int x, int x);`
   - ☐ `int pgcd(int y, int x);`
7. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
   - ☐ `int factorielle(int 2);`
   - ☐ `printf("%d", factorielle(n));`
   - ☐ `n = factorielle();`
   - ☐ `n = factorielle(p, q);`
8. Le type des réels en C est :
   - ☐ `double`
   - ☐ `real`
   - ☐ `int`
   - ☐ `char`
9. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
   - ☐ `int saisie_utilisateur();`
   - ☐ `void saisie_utilisateur(char c);`
   - ☐ `saisie_utilisateur(scanf(%d));`
   - ☐ `void saisie_utilisateur(int n);`
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 :
    - ☐ les fichiers du disque
    - ☐ certaines données de la mémoire de travail
    - ☐ en temps d'accès
    - ☐ des processus
11. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
    - ☐ `char afficher_menu(printf("menu"));`
    - ☐ `int afficher_menu();`
    - ☐ `double afficher_menu();`
    - ☐ `void afficher_menu();`
    - ☐ `int afficher_menu(int char);`
12. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
    - ☐ `printf("Valeur de n ? %d\n", n);`
    - ☐ `scanf("%d", &n);`
    - ☐ `printf("Valeur de n ? %g\n", n);`
    - ☐ un débogueur
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
    - ☐ 0
    - ☐ 16
14. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
    - ☐ 1.5
    - ☐ 0
    - ☐ 1
    - ☐ 0.5
15. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

    Alors l'expression `f(0)` prendra la valeur :
    - ☐ 0
    - ☐ 3
    - ☐ 4

16. 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

17. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :

- ☐ `n = pgcd(int p, int q);`
- ☐ `n = pgcd(n, 3);`
- ☐ `int n = pgcd();`
- ☐ `int pgcd(2);`

18. 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

19. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?

- ☐ `(!A || B)`
- ☐ `A && B`
- ☐ `!(!A || B) == (A && !B)`
- ☐ `(A == TRUE) && (B == TRUE)`

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 harmonique
- ☐ 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. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

☐ 7

☐ 5

☐ 0

2. 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

3. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

☐ `char c;`

☐ `char 'c';`

☐ `int char;`

☐ `char "c";`

4. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :

☐ `printf("Valeur de n ? %g\n", n);`

☐ `printf("Valeur de n ? %d\n", n);`

☐ un débogueur

☐ `scanf("%d", &n);`

5. 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

6. 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

7. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

☐ 3

☐ 0

☐ 4

8. Si cette erreur apparaît à la compilation : `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?

☐ une fonction déclarée mais non définie

☐ un désaccord entre la déclaration et la définition d'une fonction

☐ une directive préprocesseur `#include` manquante

☐ une fonction appelée avant sa déclaration

9. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :

☐ `void saisie_utilisateur(int n);`

☐ `saisie_utilisateur(scanf(%d));`

☐ `void saisie_utilisateur(char c);`

☐ `int saisie_utilisateur();`

10. 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

11. 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

12. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

Alors l'affichage sera :

- ☐ `ABCDEF`
- ☐ `A`
- ☐ `i`
- ☐ `cccccc`

13. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :

- ☐ `int pgcd(int y, int x);`
- ☐ `int pgcd(int x, int x);`
- ☐ `int pgcd(int x, y);`
- ☐ `void pgcd(int x, int y);`

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

15. 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

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

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

17. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :

- ☐ `exposant(double x, int n, int r);`
- ☐ `double exposant(double x, int n);`
- ☐ `int exposant(double n, int x);`
- ☐ `void exposant(double x^n);`

18. 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

19. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il comporte une boucle infinie
- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il n'affiche rien
- ☐ il ne compile pas

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. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
   - ☐ `struct date_s afficher_date(struct date_s d);`
   - ☐ `int afficher_date(date_s d);`
   - ☐ `void afficher_date(date_s d);`
   - ☐ `void afficher_date(struct date_s d);`
2. Le code suivant :

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

   affichera :
   - ☐ `Mineur`
   - ☐ rien
   - ☐ `Majeur`
   - ☐ `Mineur`
     `Majeur`
3. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   - ☐ `f(3,5)=8, a=3, b=5`
   - ☐ `f(a,b)=8, a=3, b=5`
   - ☐ `f(a,b)=8, a=8, b=5`
   - ☐ `f(a,b)=13, a=8, b=5`
4. Si cette erreur apparaît à la compilation :
   `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une fonction déclarée mais non définie
   - ☐ une directive préprocesseur `#include` manquante
5. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
   - ☐ `(!A || B)`
   - ☐ `!(!A || B) == (A && !B)`
   - ☐ `A && B`
   - ☐ `(A == TRUE) && (B == TRUE)`
6. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

   Alors pour tester l'égalité de a et de b on utilise la condition :
   - ☐ `a = b`
   - ☐ `a == b`
   - ☐ `(a.n == b.n) && (a.x == b.x)`
   - ☐ `a{n, x} == b{n, x}`
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=%x et y=%y\n");`
   - ☐ `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 ?
   - ☐ une accolade en trop
   - ☐ une accolade manquante
   - ☐ un point-virgule en trop
   - ☐ un point-virgule manquant
9. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
   - ☐ `int exposant(double n, int x);`
   - ☐ `exposant(double x, int n, int r);`
   - ☐ `double exposant(double x, int n);`
   - ☐ `void exposant(double x^n);`
10. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
    - ☐ `n = carre(n);`
    - ☐ `int n = carre();`
    - ☐ `int carre(2);`
    - ☐ `n = carre(int n);`
11. L'écriture 101 en binaire correspond au nombre naturel :
    - ☐ 5
    - ☐ 101
    - ☐ 4
    - ☐ 3
12. Pour déclarer un tableau d'entiers de taille 5, on peut utiliser l'instruction
    - ☐ `int tab[] = 5;`
    - ☐ `int[] new tableau(5);`
    - ☐ `int toto[5];`
    - ☐ `int toto[taille=5];`
    - ☐ `char tableau[5];`

13. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function ‘max’` , que doit-on chercher dans le programme ?

- ☐ une directive préprocesseur `#include` manquante
- ☐ une fonction déclarée mais non définie
- ☐ une fonction appelée avant sa déclaration
- ☐ un désaccord entre la déclaration et la définition d'une fonction

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 8
- ☐ 0 2 4 6

15. 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

16. 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

17. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 3
- ☐ 4
- ☐ 0

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

19. 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 fichier texte qui sera traduit en instructions processeur
- ☐ un document qui doit être protégé

20. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

- ☐ alphabétique
- ☐ dans lequel ces fonctions sont appelées dans le main
- ☐ dans lequel vous avez déclaré ces fonction
- ☐ un ordre quelconque

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 le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(3,5)=8, a=3, b=5`

2. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :

- ☐ `double exposant(double x, int n);`
- ☐ `void exposant(double x^n);`
- ☐ `int exposant(double n, int x);`
- ☐ `exposant(double x, int n, int r);`

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

4. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :

- ☐ `void saisie_utilisateur(char c);`
- ☐ `int saisie_utilisateur();`
- ☐ `saisie_utilisateur(scanf(%d));`
- ☐ `void saisie_utilisateur(int n);`

5. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :

- ☐ `n = carre(n);`
- ☐ `n = carre(int n);`
- ☐ `int n = carre();`
- ☐ `int carre(2);`

6. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :

- ☐ `void afficher_menu();`
- ☐ `int afficher_menu();`
- ☐ `int afficher_menu(int char);`
- ☐ `double afficher_menu();`
- ☐ `char afficher_menu(printf("menu"));`

7. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :

- ☐ `void afficher_date(date_s d);`
- ☐ `int afficher_date(date_s d);`
- ☐ `struct date_s afficher_date(struct date_s d);`
- ☐ `void afficher_date(struct date_s d);`

8. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 5
- ☐ 1
- ☐ 0
- ☐ 4

9. Un registre du processeur est :

- ☐ 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
- ☐ un composant qui contient la liste des fichiers du système

10. 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`

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

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

12. 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)

13. 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

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

15. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 0
- ☐ 3
- ☐ 4

16. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?

- ☐ `(A == TRUE) && (B == TRUE)`
- ☐ `!(!A || B) == (A && !B)`
- ☐ `A && B`
- ☐ `(!A || B)`

17. 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

18. 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

19. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

Alors pour tester l'égalité de a et de b on utilise la condition :

- ☐ `(a.n == b.n) && (a.x == b.x)`
- ☐ `a == b`
- ☐ `a = b`
- ☐ `a{n, x} == b{n, x}`

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

- ☐ tous ensemble
- ☐ 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

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
   - ☐ 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
2. Le langage C est un langage
   - ☐ lu, écrit, parlé
   - ☐ composé
   - ☐ interprété
   - ☐ compilé
3. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
   - ☐ `int saisie_utilisateur();`
   - ☐ `void saisie_utilisateur(char c);`
   - ☐ `void saisie_utilisateur(int n);`
   - ☐ `saisie_utilisateur(scanf(%d));`
4. 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`
5. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
   - ☐ 1
   - ☐ 1.5
   - ☐ 0.5
   - ☐ 0
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 :
   - ☐ des processus
   - ☐ certaines données de la mémoire de travail
   - ☐ les fichiers du disque
   - ☐ en temps d'accès
7. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
   - ☐ `int pgcd(int x, int x);`
   - ☐ `int pgcd(int y, int x);`
   - ☐ `void pgcd(int x, int y);`
   - ☐ `int pgcd(int x, y);`
8. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :
   - ☐ 4
   - ☐ 3
   - ☐ 0
9. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
   - ☐ `char afficher_menu(printf("menu"));`
   - ☐ `double afficher_menu();`
   - ☐ `int afficher_menu(int char);`
   - ☐ `int afficher_menu();`
   - ☐ `void afficher_menu();`
10. L'écriture $\underline{101}$ en binaire correspond au nombre naturel :
    - ☐ 5
    - ☐ 3
    - ☐ 101
    - ☐ 4
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 :
    - ☐ 15
    - ☐ 6
    - ☐ 10
    - ☐ 0
12. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

    Alors l'expression `f(1)` prendra la valeur :
    - ☐ 5

☐ 1

☐ 0

☐ 4

13. 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

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 1 2 0 1 2`

☐ `0 0 0 1 1 1`

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=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)`

16. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

☐ `char 'c';`

☐ `int char;`

☐ `char "c";`

☐ `char c;`

17. 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

☐ une accolade en trop

☐ un point-virgule manquant

18. 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 variable non déclarée

☐ une faute de frappe dans un appel de fonction

☐ une directive préprocesseur `#include` manquante

19. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :

☐ dans lequel vous avez déclaré ces fonction

☐ alphabétique

☐ un ordre quelconque

☐ dans lequel ces fonctions sont appelées dans le main

20. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :

☐ `int afficher_date(date_s d);`

☐ `void afficher_date(date_s d);`

☐ `void afficher_date(struct date_s d);`

☐ `struct date_s afficher_date(struct date_s 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'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
2. Si cette erreur apparaît à la compilation : `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
   - ☐ un point-virgule manquant
   - ☐ une accolade en trop
   - ☐ une accolade manquante
   - ☐ un point-virgule en trop
3. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
   - ☐ 0
   - ☐ 0.5
   - ☐ 1.5
   - ☐ 1
4. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il comporte une boucle infinie
   - ☐ il ne compile pas
   - ☐ il risque d'afficher bonjour à la place de coucou
   - ☐ il n'affiche rien
5. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
   - ☐ `n = carre(int n);`
   - ☐ `int carre(2);`
   - ☐ `int n = carre();`
   - ☐ `n = carre(n);`
6. 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>`
7. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

   appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

   L'affichage dans le main est le suivant :
   - ☐ `f(a,b)=13, a=8, b=5`
   - ☐ `f(3,5)=8, a=3, b=5`
   - ☐ `f(a,b)=8, a=8, b=5`
   - ☐ `f(a,b)=8, a=3, b=5`
8. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
   - ☐ `printf("%d", factorielle(n));`
   - ☐ `n = factorielle();`
   - ☐ `n = factorielle(p, q);`
   - ☐ `int factorielle(int 2);`
9. 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>`
10. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
    - ☐ `int saisie_utilisateur();`
    - ☐ `saisie_utilisateur(scanf(%d));`
    - ☐ `void saisie_utilisateur(int n);`
    - ☐ `void saisie_utilisateur(char c);`
11. 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'édition de lien
    - ☐ comportent une erreur qui sera détectée au cours de l'analyse syntaxique
12. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

    Quelle est la condition `cond` :
    - ☐ `(a<n) || (n>b)`
    - ☐ `(a<=n) && (n<=b)`
    - ☐ `(n<=a) && (n<=b)`
    - ☐ `a<=n<=b`

13. 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)

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

15. 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

16. 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`

17. Le code suivant :

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

affichera :

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

18. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
    - ☐ `void afficher_menu();`
    - ☐ `int afficher_menu();`
    - ☐ `int afficher_menu(int char);`
    - ☐ `double afficher_menu();`
    - ☐ `char afficher_menu(printf("menu"));`

19. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
    - ☐ `int afficher_date(date_s d);`
    - ☐ `void afficher_date(date_s d);`
    - ☐ `void afficher_date(struct date_s d);`
    - ☐ `struct date_s afficher_date(struct date_s d);`

20. Le type des réels en C est :
    - ☐ `real`
    - ☐ `char`
    - ☐ `int`
    - ☐ `double`

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 une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
   - ☐ `int pgcd(int x, int x);`
   - ☐ `int pgcd(int x, y);`
   - ☐ `int pgcd(int y, int x);`
   - ☐ `void pgcd(int x, int y);`
2. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
   - ☐ `void saisie_utilisateur(int n);`
   - ☐ `void saisie_utilisateur(char c);`
   - ☐ `int saisie_utilisateur();`
   - ☐ `saisie_utilisateur(scanf(%d));`
3. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
   - ☐ `n = factorielle(p, q);`
   - ☐ `int factorielle(int 2);`
   - ☐ `printf("%d", factorielle(n));`
   - ☐ `n = factorielle();`
4. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

   Quelle est la condition `cond` :
   - ☐ `(a<n) || (n>b)`
   - ☐ `a<=n<=b`
   - ☐ `(n<=a) && (n<=b)`
   - ☐ `(a<=n) && (n<=b)`
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. 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
7. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
   - ☐ `(A == TRUE) && (B == TRUE)`
   - ☐ `(!A || B)`
   - ☐ `A && B`
   - ☐ `!(!A || B) == (A && !B)`
8. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   - ☐ 7
   - ☐ 5
   - ☐ 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
   - ☐ 4 3 2 1
   - ☐ 0 1 2 3
   - ☐ 4 3 2 1 0
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. 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 `****`

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. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il ne compile pas
- ☐ il n'affiche rien
- ☐ il comporte une boucle infinie

14. Un enregistrement permet de grouper plusieurs valeurs dans :

- ☐ ses champs
- ☐ ses blocs
- ☐ ses cases
- ☐ ses chants

15. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :

- ☐ `toto_s n, x;`
- ☐ `int toto.n = 3;`
- ☐ `struct toto_s toto;`
- ☐ `toto_s struct z = {3, 0.5};`
- ☐ `int struct toto_s = {3, -1e10};`

16. Avant de faire appel à une fonction il est nécessaire de :

- ☐ l'avoir déclarée
- ☐ l'avoir déclarée et définie
- ☐ l'avoir définie
- ☐ avoir défini une constante symbolique de la taille de cette fonction

17. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

- ☐ `int char;`
- ☐ `char 'c';`
- ☐ `char c;`
- ☐ `char "c";`

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
- ☐ 16
- ☐ 0

19. 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

20. Le langage C est un langage

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

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 la déclaration : `int mccarthy(int n);`, il est correct d'écrire :
   - ☐ `int mccarthy(int 2);`
   - ☐ `x = mccarthy(n);`
   - ☐ `n = mccarthy(p, q);`
   - ☐ `n = mccarthy();`
2. 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
   - ☐ une gamme de fréquence de fonctionnement du processeur
   - ☐ un composant qui contient la liste des fichiers du système
3. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

   Quelle est la condition `cond` :
   - ☐ `(a<n) || (n>b)`
   - ☐ `(n<=a) && (n<=b)`
   - ☐ `a<=n<=b`
   - ☐ `(a<=n) && (n<=b)`
4. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :
   - ☐ 4
   - ☐ 0
   - ☐ 3
5. 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;`
6. Le langage C est un langage
   - ☐ composé
   - ☐ lu, écrit, parlé
   - ☐ compilé
   - ☐ interprété
7. 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
8. L'écriture $\underline{101}$ en binaire correspond au nombre naturel :
   - ☐ 4
   - ☐ 3
   - ☐ 101
   - ☐ 5
9. Le type des réels en C est :
   - ☐ `int`
   - ☐ `real`
   - ☐ `double`
   - ☐ `char`
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
```

    - ☐ la case mémoire 8 contiendra 16
    - ☐ le bus explose
    - ☐ la case mémoire 8 contiendra 0
    - ☐ le terminal affiche 8
11. Un programme en langage C doit comporter une et une seule définition de la fonction :
    - ☐ main
    - ☐ include
    - ☐ init
    - ☐ begin
12. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

    - ☐ il ne compile pas
    - ☐ il n'affiche rien
    - ☐ il risque d'afficher bonjour à la place de coucou
    - ☐ il comporte une boucle infinie
13. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
    - ☐ `exposant(double x, int n, int r);`
    - ☐ `int exposant(double n, int x);`
    - ☐ `double exposant(double x, int n);`
    - ☐ `void exposant(double x^n);`

14. 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`
15. Laquelle des analyses suivantes ne fait pas partie des étapes de la compilation :
    - ☐ analyse sémantique
    - ☐ analyse harmonique
    - ☐ analyse syntaxique
    - ☐ analyse lexicale
16. 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
17. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

    Alors pour tester l'égalité de a et de b on utilise la condition :
    - ☐ `(a.n == b.n) && (a.x == b.x)`
    - ☐ `a = b`
    - ☐ `a == b`
    - ☐ `a{n, x} == b{n, x}`
18. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses blocs
    - ☐ ses chants
    - ☐ ses champs
    - ☐ ses cases
19. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

    Alors l'expression `f(1)` prendra la valeur :
    - ☐ 0
    - ☐ 4
    - ☐ 1
    - ☐ 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 :
    - ☐ 0
    - ☐ 42
    - ☐ 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. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   ☐ il ne compile pas
   ☐ il risque d'afficher bonjour à la place de coucou
   ☐ il n'affiche rien
   ☐ il comporte une boucle infinie

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. 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

4. 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`

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

6. Pour déclarer une fonction `exposant` qui prend en argument un réel x et un entier positif n et renvoie la valeur de x^n on écrit :
   ☐ `int exposant(double n, int x);`
   ☐ `exposant(double x, int n, int r);`
   ☐ `double exposant(double x, int n);`
   ☐ `void exposant(double x^n);`

7. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
   ☐ 8
   ☐ 111
   ☐ 7
   ☐ 3

8. Avant de faire appel à une fonction il est nécessaire de :
   ☐ l'avoir déclarée et définie
   ☐ avoir défini une constante symbolique de la taille de cette fonction
   ☐ l'avoir définie
   ☐ l'avoir déclarée

9. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :
   ☐ `int n = pgcd();`
   ☐ `n = pgcd(int p, int q);`
   ☐ `n = pgcd(n, 3);`
   ☐ `int pgcd(2);`

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. Une variable booléenne est un variable :
   ☐ jamais nulle
   ☐ NaN (not a number, qui n'est pas un nombre)
   ☐ qui est vraie ou fausse
   ☐ à laquelle une valeur vient d'être affectée
   ☐ réelle positive

12. 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;`

13. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
   ☐ `scanf("%d", &n);`
   ☐ un débogueur
   ☐ `printf("Valeur de n ? %d\n", n);`
   ☐ `printf("Valeur de n ? %g\n", n);`

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

15. 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`
16. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 0
- ☐ 4
- ☐ 1
- ☐ 5

17. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 3
- ☐ 4
- ☐ 0

18. 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
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 0 1 1 2 2`
- ☐ `0 1 2 0 1 2`
- ☐ `1 2 3 1 2`
- ☐ `0 1 0 1 0 1`

20. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=8, b=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. 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

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

3. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :
   - ☐ 4
   - ☐ 0
   - ☐ 3

4. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
   - ☐ 1.5
   - ☐ 1
   - ☐ 0.5
   - ☐ 0

5. L'écriture 111 en binaire correspond au nombre naturel :
   - ☐ 7
   - ☐ 3
   - ☐ 8
   - ☐ 111

6. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :
   - ☐ un débogueur
   - ☐ `scanf("%d", &n);`
   - ☐ `printf("Valeur de n ? %g\n", n);`
   - ☐ `printf("Valeur de n ? %d\n", n);`

7. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
   - ☐ `char "c";`
   - ☐ `int char;`
   - ☐ `char 'c';`
   - ☐ `char c;`

8. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

   Alors l'affichage sera :
   - ☐ b
   - ☐ A
   - ☐ C
   - ☐ B

9. 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

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

11. L'ordonnancement par tourniquet permet :
    - ☐ de doubler la mémoire disponible
    - ☐ 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

12. L'écriture 101 en binaire correspond au nombre naturel :
    - ☐ 4
    - ☐ 101
    - ☐ 5
    - ☐ 3

13. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

    Alors l'affichage sera :
    - ☐ A
    - ☐ ABCDEF
    - ☐ cccccc
    - ☐ i

14. 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

15. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

    Alors l'expression `g(0)` prendra la valeur :
    - ☐ 7
    - ☐ 0
    - ☐ 5

16. Si cette erreur apparaît à la compilation :
    `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction déclarée mais non définie
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une directive préprocesseur `#include` manquante

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
    - ☐ 16
    - ☐ 8
    - ☐ 4

18. 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`

19. 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);`

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

2. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
   - ☐ 0.5
   - ☐ 1.5
   - ☐ 0
   - ☐ 1

3. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

   Alors pour tester l'égalité de a et de b on utilise la condition :
   - ☐ `a{n, x} == b{n, x}`
   - ☐ `a = b`
   - ☐ `a == b`
   - ☐ `(a.n == b.n) && (a.x == b.x)`

4. 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

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

6. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :
   - ☐ 0
   - ☐ 4
   - ☐ 3

7. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
   - ☐ `char 'c';`
   - ☐ `char c;`
   - ☐ `int char;`
   - ☐ `char "c";`

8. 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

9. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il risque d'afficher bonjour à la place de coucou
   - ☐ il n'affiche rien
   - ☐ il comporte une boucle infinie
   - ☐ il ne compile pas

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
    - ☐ 4 3 2 1
    - ☐ 0 1 2 3

11. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :
    - ☐ `n = pgcd(n, 3);`
    - ☐ `int n = pgcd();`
    - ☐ `int pgcd(2);`
    - ☐ `n = pgcd(int p, int q);`

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

13. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=13, a=8, b=5`

14. 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>`

15. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
    - ☐ `printf("%d", factorielle(n));`
    - ☐ `n = factorielle(p, q);`
    - ☐ `n = factorielle();`
    - ☐ `int factorielle(int 2);`

16. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
    - ☐ `void saisie_utilisateur(char c);`
    - ☐ `int saisie_utilisateur();`
    - ☐ `void saisie_utilisateur(int n);`
    - ☐ `saisie_utilisateur(scanf(%d));`

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

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

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
    - ☐ des processus
    - ☐ en temps d'accès
    - ☐ certaines données de la mémoire de travail

20. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 5
- ☐ 7
- ☐ 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. 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 somme = 0;
int i;
for (i = 1; i < 4; i = i + 1)
{
  somme = somme + i;
}
printf("%d", somme);
```

   affichera :
   - ☐ 6
   - ☐ 1
   - ☐ 0
   - ☐ 42
3. Une variable booléenne est un variable :
   - ☐ qui est vraie ou fausse
   - ☐ à laquelle une valeur vient d'être affectée
   - ☐ réelle positive
   - ☐ NaN (not a number, qui n'est pas un nombre)
   - ☐ jamais nulle
4. Si cette erreur apparaît à la compilation : `error: expected ';' before '}' token` que doit-on chercher dans le programme ?
   - ☐ un point-virgule manquant
   - ☐ une accolade en trop
   - ☐ un point-virgule en trop
   - ☐ une accolade manquante
5. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il risque d'afficher bonjour à la place de coucou
   - ☐ il comporte une boucle infinie
   - ☐ il n'affiche rien
   - ☐ il ne compile pas
6. Si cet avertissement apparaît à la compilation : `warning: implicit declaration of function 'max'` , que doit-on chercher dans le programme ?
   - ☐ une directive préprocesseur `#include` manquante
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une fonction déclarée mais non définie
   - ☐ une fonction appelée avant sa déclaration
7. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

   Alors l'expression `f(0)` prendra la valeur :
   - ☐ 3
   - ☐ 4
   - ☐ 0
8. Au début de la fonction `main()` on place le code :

```
char i;
for (i = 'A'; i <= 'F'; i = i + 1)
{
  printf("%c", i);
}
printf("\n");
```

   Alors l'affichage sera :
   - ☐ `A`
   - ☐ `cccccc`
   - ☐ `ABCDEF`
   - ☐ `i`
9. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

   Alors l'expression `f(1)` prendra la valeur :
   - ☐ 5
   - ☐ 4
   - ☐ 1
   - ☐ 0
10. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses chants
    - ☐ ses blocs
    - ☐ ses champs
    - ☐ ses cases
11. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
    - ☐ `void afficher_date(date_s d);`
    - ☐ `void afficher_date(struct date_s d);`
    - ☐ `struct date_s afficher_date(struct date_s d);`
    - ☐ `int afficher_date(date_s d);`
12. Avant de faire appel à une fonction il est nécessaire de :
    - ☐ l'avoir déclarée et définie
    - ☐ l'avoir définie
    - ☐ avoir défini une constante symbolique de la taille de cette fonction
    - ☐ l'avoir déclarée

13. 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`
14. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?
    - ☐ `!(!A || B) == (A && !B)`
    - ☐ `A && B`
    - ☐ `(!A || B)`
    - ☐ `(A == TRUE) && (B == TRUE)`
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 `x = 5`
    - ☐ le terminal affiche `5`
    - ☐ le terminal affiche "Faux"
16. Le code suivant :

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

    affichera :
    - ☐ `Mineur`
    - ☐ rien
    - ☐ `Majeur`
    - ☐ `Mineur`
      `Majeur`
17. Dans la commande gcc, l'option `-Wall` signifie :
    - ☐ qu'il faut indenter le fichier source
    - ☐ qu'il faut lancer un déboggueur
    - ☐ que l'on veut voir tous les avertissements
    - ☐ qu'on veut changer alétoirement de fond d'écran
18. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
    - ☐ `void afficher_menu();`
    - ☐ `double afficher_menu();`
    - ☐ `int afficher_menu();`
    - ☐ `char afficher_menu(printf("menu"));`
    - ☐ `int afficher_menu(int char);`
19. 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
20. Si a et b sont deux variables de type :

```
struct toto_s
{
  int n;
  double x;
};
```

    Alors pour tester l'égalité de a et de b on utilise la condition :
    - ☐ `a{n, x} == b{n, x}`
    - ☐ `a = b`
    - ☐ `a == b`
    - ☐ `(a.n == b.n) && (a.x == b.x)`

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=1;i<5;i=i+1)`
   - ☐ `for(i=0;i<5;i=i+1)`
   - ☐ `for(i=0;i<=5;i=i+1)`
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. 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`
4. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :
   - ☐ `n = pgcd(n, 3);`
   - ☐ `n = pgcd(int p, int q);`
   - ☐ `int pgcd(2);`
   - ☐ `int n = pgcd();`
5. 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 = 0`
   - ☐ `j = 5`
6. Pour déclarer une procédure `afficher_menu` sans argument et qui ne renvoie rien on utilise :
   - ☐ `void afficher_menu();`
   - ☐ `int afficher_menu();`
   - ☐ `int afficher_menu(int char);`
   - ☐ `char afficher_menu(printf("menu"));`
   - ☐ `double afficher_menu();`
7. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
   - ☐ alphabétique
   - ☐ un ordre quelconque
   - ☐ dans lequel ces fonctions sont appelées dans le main
   - ☐ dans lequel vous avez déclaré ces fonction
8. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?
   - ☐ `char 'c';`
   - ☐ `char c;`
   - ☐ `int char;`
   - ☐ `char "c";`
9. 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
10. Si `carre` est une fonction prenant en entrée un entier et renvoyant le carré de cet entier, et que n est une variable entière définie et initialisée, il est correct d'écrire :
    - ☐ `n = carre(n);`
    - ☐ `int n = carre();`
    - ☐ `n = carre(int n);`
    - ☐ `int carre(2);`
11. 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;`
12. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

    Quelle est la condition `cond` :
    - ☐ `(a<=n) && (n<=b)`
    - ☐ `(n<=a) && (n<=b)`
    - ☐ `(a<n) || (n>b)`
    - ☐ `a<=n<=b`

13. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

- ☐ 5
- ☐ 1
- ☐ 0
- ☐ 4

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

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

15. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

Alors l'affichage sera :

- ☐ `B`
- ☐ `C`
- ☐ `A`
- ☐ `b`

16. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 5
- ☐ 7
- ☐ 0

17. Si n est une variable entière pour demander sa valeur à l'utilisateur, on utilise plutôt :

- ☐ un débogueur
- ☐ `printf("Valeur de n ? %d\n", n);`
- ☐ `printf("Valeur de n ? %g\n", n);`
- ☐ `scanf("%d", &n);`

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  }
```

- ☐ 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

19. 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

20. 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

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
☐ 0
☐ 15

2. Le langage C est un langage

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

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 2
☐ 8 6 4 2 0

4. Après la déclaration : `int mccarthy(int n);`, il est correct d'écrire :

☐ `int mccarthy(int 2);`
☐ `x = mccarthy(n);`
☐ `n = mccarthy();`
☐ `n = mccarthy(p, q);`

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 = 5`
☐ `j = 4`
☐ `j = %d`

6. Si `pgcd` est une fonction prenant en entrée deux entiers et renvoyant un entier, il est correct d'écrire :

☐ `int n = pgcd();`
☐ `n = pgcd(n, 3);`
☐ `n = pgcd(int p, int q);`
☐ `int pgcd(2);`

7. Un enregistrement permet de grouper plusieurs valeurs dans :

☐ ses blocs
☐ ses cases
☐ ses chants
☐ ses champs

8. Si `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :

☐ `x = racine(2/3);`
☐ `x = racine(racine(x)*racine(x));`
☐ `x = racine(x * x) - racine(x);`
☐ `x - 1 = racine(x);`

9. On considère deux variables booléennes `A` et `B` initialisées à `TRUE` et `FALSE` respectivement. Parmi les expressions booléennes suivantes, laquelle a pour valeur `TRUE` ?

☐ `A && B`
☐ `(A == TRUE) && (B == TRUE)`
☐ `(!A || B)`
☐ `!(!A || B) == (A && !B)`

10. Avant de faire appel à une fonction il est nécessaire de :

☐ l'avoir définie
☐ l'avoir déclarée
☐ l'avoir déclarée et définie
☐ avoir défini une constante symbolique de la taille de cette fonction

11. 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

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

13. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

Alors l'affichage sera :

- ☐ `C`
- ☐ `B`
- ☐ `b`
- ☐ `A`

14. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

Alors l'expression `g(0)` prendra la valeur :

- ☐ 5
- ☐ 7
- ☐ 0

15. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :

- ☐ `int pgcd(int x, y);`
- ☐ `void pgcd(int x, int y);`
- ☐ `int pgcd(int y, int x);`
- ☐ `int pgcd(int x, int x);`

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
- ☐ 8
- ☐ 0
- ☐ 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
- ☐ 0 1 2 3 4
- ☐ 1 2 3 4
- ☐ 4 3 2 1

18. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

- ☐ il ne compile pas
- ☐ il risque d'afficher bonjour à la place de coucou
- ☐ il n'affiche rien
- ☐ il comporte une boucle infinie

19. 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 document qui doit être protégé
- ☐ un fichier que l'ont doit citer dans les documents produits sur l'ordinateur

20. Vous utilisez une boucle `while` quand :

- ☐ vous avez déjà fait un `for` dans le même programme principal
- ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
- ☐ vous n'avez pas déclaré de fonction
- ☐ l'incrément de la variable de boucle n'est pas 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 age = 15;
if (age < 18)
{
    printf("Mineur\n");
}
else
{
    printf("Majeur\n");
}
```

affichera :

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

2. Un programme en langage C doit comporter une et une seule définition de la fonction :
   - ☐ begin
   - ☐ main
   - ☐ include
   - ☐ init
3. 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 -Wall prog.exe -o prog.c`
   - ☐ `gcc prog.c -o -Wall prog.exe`
4. Pour déclarer une fonction `pgcd` qui calcule et renvoie le plus grand diviseur commun de deux entiers positifs passés en arguments on écrit :
   - ☐ `int pgcd(int x, y);`
   - ☐ `int pgcd(int x, int x);`
   - ☐ `int pgcd(int y, int x);`
   - ☐ `void pgcd(int x, int y);`
5. L'écriture $\underline{111}$ en binaire correspond au nombre naturel :
   - ☐ 3
   - ☐ 8
   - ☐ 111
   - ☐ 7
6. Si cette erreur apparaît à la compilation :
   `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une fonction déclarée mais non définie
   - ☐ une directive préprocesseur `#include` manquante
7. Si le code :

```
struct toto_s
{
  int n;
  double x;
};
```

précède la fonction `main()`, alors on peut écrire en début de `main()` :
   - ☐ `int toto.n = 3;`
   - ☐ `int struct toto_s = {3, -1e10};`
   - ☐ `toto_s n, x;`
   - ☐ `struct toto_s toto;`
   - ☐ `toto_s struct z = {3, 0.5};`
8. Si `factorielle` est une fonction prenant en entrée un entier et renvoyant un entier, il est correct d'écrire :
   - ☐ `int factorielle(int 2);`
   - ☐ `n = factorielle();`
   - ☐ `printf("%d", factorielle(n));`
   - ☐ `n = factorielle(p, q);`
9. Pour déclarer une fonction `saisie_utilisateur` qui demande à l'utilisateur d'entrer un entier au clavier et renvoie cet entier on écrit :
   - ☐ `void saisie_utilisateur(int n);`
   - ☐ `int saisie_utilisateur();`
   - ☐ `saisie_utilisateur(scanf(%d));`
   - ☐ `void saisie_utilisateur(char c);`
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 0 1 0 1 0 1`
   - ☐ `1 2 1 2 3`
   - ☐ `0 1 2 0 1 2`
11. Si x est une variable réelle (de type double) alors `x = 3/2` lui affecte la valeur :
   - ☐ 1
   - ☐ 0
   - ☐ 0.5
   - ☐ 1.5
12. 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
13. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

Alors l'expression `f(1)` prendra la valeur :

☐ 4

☐ 0

☐ 1

☐ 5

14. Le langage C est un langage

☐ lu, écrit, parlé

☐ composé

☐ compilé

☐ interprété

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 1 2 0 1 2 3

☐ 0 0 1 1 2 2 3

☐ 0 1 2 3 0 1 2

16. 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

17. Au début de la fonction `main()` on place le code :

```
char b = 'A';
b = b + 2;
printf("%c\n", b);
```

Alors l'affichage sera :

☐ b

☐ A

☐ C

☐ B

18. 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

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

☐ `for(i=1;i<=5;i=i+1)`

☐ `for(i=1;i<5;i=i+1)`

20. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

☐ `f(a,b)=13, a=8, b=5`

☐ `f(3,5)=8, a=3, b=5`

☐ `f(a,b)=8, a=8, b=5`

☐ `f(a,b)=8, a=3, b=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. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=8, b=5`
- ☐ `f(3,5)=8, a=3, b=5`

2. Vous utilisez une boucle `while` quand :
   - ☐ vous n'avez pas déclaré de fonction
   - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
   - ☐ vous avez déjà fait un `for` dans le même programme principal
   - ☐ l'incrément de la variable de boucle n'est pas 1

3. Le code suivant :

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

affichera :

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

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

5. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `a<=n<=b`
- ☐ `(n<=a) && (n<=b)`
- ☐ `(a<=n) && (n<=b)`
- ☐ `(a<n) || (n>b)`

6. Si cette erreur apparaît à la compilation :
   `erreur: conflicting types for 'max'` , que doit-on chercher dans le programme ?
   - ☐ un désaccord entre la déclaration et la définition d'une fonction
   - ☐ une fonction déclarée mais non définie
   - ☐ une fonction appelée avant sa déclaration
   - ☐ une directive préprocesseur `#include` manquante

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

8. Pour déclarer une fonction `factorielle` qui prend en argument un entier et renvoie sa factorielle on écrit :
   - ☐ `struct int factorielle(int n);`
   - ☐ `int factorielle(int x);`
   - ☐ `int factorielle();`
   - ☐ `int factorielle(double n);`

9. Le type des réels en C est :
   - ☐ `real`
   - ☐ `double`
   - ☐ `int`
   - ☐ `char`

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. 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)`

12. L'écriture $\underline{101}$ en binaire correspond au nombre naturel :
    - ☐ 101
    - ☐ 5
    - ☐ 3
    - ☐ 4

13. Un enregistrement permet de grouper plusieurs valeurs dans :
    - ☐ ses chants
    - ☐ ses cases
    - ☐ ses champs
    - ☐ ses blocs

14. Vous avez déclaré préalablement un ensemble de fonctions utilisées par votre programme principal. L'ordre dans lequel vous devez maintenant définir ces fonctions est l'ordre :
    - ☐ alphabétique
    - ☐ un ordre quelconque
    - ☐ dans lequel ces fonctions sont appelées dans le main
    - ☐ dans lequel vous avez déclaré ces fonction

15. Un fichier source est :
    - ☐ 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 de référence du système
    - ☐ un document illisible pour les humains

16. 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>`

17. Si cet avertissement apparaît à la compilation :
    `warning: implicit declaration of function ‘max’`
    , que doit-on chercher dans le programme ?
    - ☐ un désaccord entre la déclaration et la définition d'une fonction
    - ☐ une fonction appelée avant sa déclaration
    - ☐ une directive préprocesseur `#include` manquante
    - ☐ une fonction déclarée mais non définie

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

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

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  }
```

    - ☐ 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"

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 `racine` est une fonction prenant en entrée un réel et renvoyant la racine carrée de cet réel, et que x est une variable réelle définie et initialisée, il est incorrect d'écrire :
   - ☐ `x - 1 = racine(x);`
   - ☐ `x = racine(2/3);`
   - ☐ `x = racine(x * x) - racine(x);`
   - ☐ `x = racine(racine(x)*racine(x));`
2. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return f(a - 1) + 1;
  }
  return 4;
}
```

   Alors l'expression `f(1)` prendra la valeur :
   - ☐ 5
   - ☐ 0
   - ☐ 1
   - ☐ 4
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. 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 illisible pour les humains
   - ☐ un document qui doit être protégé
   - ☐ un fichier texte qui sera traduit en instructions processeur
5. Pour déclarer une procédure `afficher_date` qui prend en argument un `struct date_s` et affiche le contenu du struct, on écrit :
   - ☐ `void afficher_date(struct date_s d);`
   - ☐ `struct date_s afficher_date(struct date_s d);`
   - ☐ `void afficher_date(date_s d);`
   - ☐ `int afficher_date(date_s d);`
6. Le code suivant :

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

   affichera :
   - ☐ `Mineur`
   - ☐ `Mineur`
`Majeur`
   - ☐ `Majeur`
   - ☐ rien
7. Sous unix (ou linux), la commande `cd` permet de :
   - ☐ récupérer un programme arrêté avec la commande `ab`
   - ☐ ouvir un bureau partagé (common desktop)
   - ☐ détruire un fichier
   - ☐ jouer de la musique
   - ☐ changer de répertoire courant
8. Soit la fonction `g` définie par :

```
int g(int a)
{
  printf("a = \n", %d);
  if (1 > 0)
  {
    return 5;
  }
  return 7;
}
```

   Alors l'expression `g(0)` prendra la valeur :
   - ☐ 0
   - ☐ 5
   - ☐ 7
9. Quel est le problème d'un programme comportant les lignes suivantes ?

```
while (1)
{
  printf("coucou\n");
}
```

   - ☐ il comporte une boucle infinie
   - ☐ il n'affiche rien
   - ☐ il ne compile pas
   - ☐ il risque d'afficher bonjour à la place de coucou
10. Vous utilisez une boucle `while` quand :
    - ☐ vous ne connaissez pas le nombre d'itérations de la boucle à l'avance
    - ☐ vous n'avez pas déclaré de fonction
    - ☐ l'incrément de la variable de boucle n'est pas 1
    - ☐ vous avez déjà fait un `for` dans le même programme principal

11. On souhaite faire une boucle de contrôle de saisie : tant que l'entier `n` n'appartient pas à l'intervalle $[a..b]$, on recommence la saisie de `n`. Soit le programme suivant :

```
int a = 0;
int b = 20;
int n;
scanf("%d", &n);
while(cond)
{
  scanf("%d", &n);
}
```

Quelle est la condition `cond` :

- ☐ `a<=n<=b`
- ☐ `(a<=n) && (n<=b)`
- ☐ `(a<n) || (n>b)`
- ☐ `(n<=a) && (n<=b)`

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
- ☐ 0 2 4 6 8
- ☐ 0 1 2 3 4 5 6

13. Soit la fonction `f` définie par :

```
int f(int a)
{
  printf("a = \n", %d);
  if (a > 0)
  {
    return 3;
  }
  return 4;
}
```

Alors l'expression `f(0)` prendra la valeur :

- ☐ 0
- ☐ 4
- ☐ 3

14. 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

15. 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
- ☐ ne comportent aucune erreur
- ☐ comportent une erreur qui ne sera pas détectée

16. 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`

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
- ☐ 0 2 4 6 8
- ☐ 8 6 4 2 0

18. Soit le programme principal suivant :

```
int main()
{
 int a = 3;
 int b = 5;
 printf("f(a,b)=%d, a=%d, b=%d\n",f(a,b),a,b);
 return EXIT_SUCCESS;
}
```

appelant la fonction `f` ainsi définie :

```
int f(int a, int b)
{
  a = a + b;
  return a;
}
```

L'affichage dans le main est le suivant :

- ☐ `f(a,b)=13, a=8, b=5`
- ☐ `f(a,b)=8, a=3, b=5`
- ☐ `f(3,5)=8, a=3, b=5`
- ☐ `f(a,b)=8, a=8, b=5`

19. Laquelle de ces écritures correspond à la déclaration d'une variable de type caractère en langage C ?

- ☐ `char c;`
- ☐ `char "c";`
- ☐ `int char;`
- ☐ `char 'c';`

20. 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