Nom :

Prénom :

QCM de rentrée

Répondre par vrai ou faux aux questions suivantes. Barème : 0,5 pour une réponse correcte, -0,5 pour une réponse incorrecte et 0 en cas d'absence de réponse. Dans cet exercice, il n'est pas demandé de justification. Ici, **Vrai** signifie « toujours vrai » et **Faux** signifie « pas toujours vrai ».

Vrai Faux

Compilation

1. Les définitions des fonctions de bibliothèque se trouvent dans des fichiers d'en-tête (extension `.h`).....
2. Pour créer un fichier objet (`.o`) à partir d'un source C (`.c`), il n'est pas nécessaire d'avoir défini toutes les fonctions avant leur utilisation dans le source.
3. Dans un programme, une variable ne peut être définie qu'une seule fois, mais peut être déclarée plusieurs fois, dans des fichiers différents.
4. Dans un fichier C, on peut utiliser des variables de type `struct ma_structure` seulement si on a défini la structure `ma_structure` dans ce fichier.
5. On peut arrêter la compilation après la phase de préprocessing, après celle de compilation ou celle d'assemblage.
6. L'édition de liens est la dernière phase de compilation, et a pour but de s'assurer que tout symbole a une unique définition et de créer l'exécutable.
7. Le préprocesseur C ne fait que des remplacements de texte, sans vérifier si le résultat est syntaxiquement correct.....
8. On peut écrire des macros prenant un nombre variable d'arguments.
9. C'est grâce à l'instruction `typedef` que l'on peut construire de nouveaux types. ...

Variables et expressions

10. En l'absence d'initialisation explicite, une variable de type `int` est initialisée à 0.
11. Les deux séquences suivantes

Séquence 1 : <code>static int x;</code> <code> x = 1;</code>	Séquence 2 : <code>static int x = 1;</code>
--	---

sont équivalentes si elles sont placées au début d'une définition de fonction.....
12. Une variable (ou fonction) `static` est privée au fichier dans lequel est déclarée...
13. Une variable `static` est allouée pendant toute la durée du programme, même si elle est locale à une fonction.....
14. La séquence suivante : `for (unsigned int x = 10; x >= 0; x--)`
`printf("%d\n", x);`
affiche exactement 11 lignes.
15. Le Booléen (`010 == 8`) est vrai.....
16. Le Booléen (`0 == '\0'`) est vrai.....

Exercice 1

17. La séquence suivante : `unsigned char c = 1089;`
`printf("%d %c\n", c, c);`
affiche 65 A.
18. L'expression `f(i++,i++)` s'évalue comme `f(1,2)` lorsque `i` vaut initialement 0 ...
19. Les constructions `x++` et `++x` sont équivalentes.
20. L'expression `(f() && g())` est équivalente à l'expression `(g() && f())`
21. Si `x` est une variable de type `int`, `(x = x ? 0 : 1)` équivaut à `(x = !x)`.....

Tableaux et pointeurs

22. Si un tableau `tab` a 3 cases, ces cases sont `tab[0]`, `tab[1]` et `tab[2]`.....
23. On peut calculer la taille d'un tableau à partir du nom de ce tableau.
24. Le nom d'un tableau a pour valeur l'adresse de sa première case.
25. Il est interdit d'affecter le nom d'un tableau.
26. On peut affecter un `void *` à un `int *`, sans avertissement du compilateur.
27. Si `p` est un pointeur valant `0x16d02a410`, `(p + 5)` vaut `0x16d02a415`.
28. Si `p` désigne un pointeur, les expressions `&p[1]` et `(p + 1)` sont équivalentes.
29. Si `p` désigne un pointeur, les expressions `p[1]`, `*(p+1)` et `1[p]` sont équivalentes. .
30. Après la séquence : `char c[5];`
`c[3] = '\0';`
il est interdit d'écrire dans `c[4]`.....
31. La ligne `int *p, *q, *r;` définit des variables `*p`, `*q` et `*r` de type `int`.
32. La séquence suivante est incorrecte en C : `char c;`
`char p[10], *q;`
`p = q = &c;` ;

33. On a écrit la fonction `swap` suivante :
- ```
void swap(int *x, int *y){
 int *tmp = x;
 x = y;
 y = tmp;
}
```
- Si `a` et `b` sont deux variables de type `int`, l'instruction `swap(&a, &b);` échange les valeurs de `a` et `b`. ....

### Divers

34. On peut écrire des fonctions prenant un nombre variable de paramètres. ....
35. L'instruction `break` n'est permise que dans les boucles (`while`, `for`, `do...while`).
36. L'instruction `continue` n'est permise que dans les boucles. ....
37. On peut afficher le nom, la fonction courante et la ligne courante d'un source C...
38. Avant `free(p);`, on doit s'assurer que `p` n'est pas `NULL`. ....
39. Avant `free(p);`, on doit s'assurer que `p` a été alloué par `re/c/malloc()`. ....
40. Les fichiers manipulés par les fonctions de la bibliothèque standard sont terminés par un caractère spécial `EOF`. ....