Identification /5

Prénom: .....

Nom: .....

Code permanent: .....

Veuillez choisir **une seule réponse** par question, avec une croix (×) ou un crochet (✓).

Considérez le programme `prog.c` suivant:

```
#include <stdio.h>

void f(int a, int* b) {
  ++a;
  *b += a;
}

int main(void) {
  int a = 1, b = 2;
  f(a, &b);
  printf("%d\n%d\n", a, b);
  return 0;
}
```

Q1. Qu'est-ce qui sera affiché sur la première ligne de la sortie standard lorsqu'on exécute le programme `prog.c`?

☐ 0 ■ 1 ☐ 2 ☐ 4

Q2. Qu'est-ce qui sera affiché sur la deuxième ligne de la sortie standard lorsqu'on exécute le programme `prog.c`?

☐ 0 ☐ 1 ☐ 2 ■ 4

Considérez le programme `str.c` suivant:

```
#include <stdio.h>
#include <string.h>

int main(void) {
  char s[] = "quiz", t[10];
  strcpy(t, s); strcat(t, s);
  printf("%lu\n%lu\n", sizeof(t),
                       strlen(t));
  return 0;
}
```

Q3. Qu'est-ce qui sera affiché sur la première ligne de la sortie standard lorsqu'on exécute le programme `str.c`?

☐ 1 ☐ 8 ☐ 9 ■ 10

Q4. Qu'est-ce qui sera affiché sur la deuxième ligne de la sortie standard lorsqu'on exécute le programme `str.c`?

☐ 1 ■ 8 ☐ 9 ☐ 10

Q5. Qu'est-ce qui sera affiché sur la sortie standard si on exécute le programme suivant?

```
#include <stdio.h>

int main(void) {
  int c = 0;
  for (int i = 0; i < 5; ++i) {
    for (int j = i; j < 5; ++j) {
      if ((j - i) % 2 == 0)
        continue;
      ++c;
    }
  }
  printf("%d\n", c);
  return 0;
}
```

☐ 0 ■ 6 ☐ 9 ☐ 15

Voir explications page suivante

Q1. Le premier argument de la fonction `f` est passé par valeur. Cela signifie donc que le paramètre `int a` est modifié localement lorsqu'on exécute l'instruction `++a`. La valeur `int a` de la fonction `main` reste donc inchangée par l'appel.
Q2. Le deuxième argument de la fonction `f` est une adresse, ce qui correspond à un pointeur lorsqu'on utilise le paramètre `int* b`. L'instruction `++a` incrémente la valeur (locale) de `a` à `2`, de sorte que l'affectation `*b += a` ajoute `2` à la valeur pointée par `b`, qui est déjà à `2` et donc qui passe à `4`. Comme on a cette fois-ci passé une adresse, la modification persiste, c'est-à-dire que la variable `b` de la fonction `main` est bien égale à `4` après avoir appelé `f`.
Q3. L'opérateur `sizeof` appliqué à un tableau retourne le nombre d'octets occupé par ce tableau. Comme `t` est un tableau de 10 caractères, sa taille est égale à `10 * sizeof(char)`, qui vaut bien 10, puisque `sizeof(char)` vaut 1.
Q4. L'appel de la fonction `strcpy` copie la chaîne `"quiz"` à l'intérieur du tableau `t`, en insérant le caractère nul à la fin de la chaîne en 5e position. Ensuite, l'appel de la fonction `strcat` concatène la chaîne `"quiz"` à la suite, ce qui fait que `t` contient la chaîne `"quizquiz"`. La fonction `strlen` retourne la longueur de cette chaîne de caractère, qui est bien formée, car `t` a suffisamment d'espace. Ne pas oublier que le caractère nul se trouvant à la fin de la chaîne n'est pas comptabilisé.
Q5. On remarque que le programme énumère les paires (j, i) telles que $0 \leq i \leq j \leq 4$ et incrémente la variable `c`, sauf lorsque `j - i` est pair, où l'incrémentation est ignorée grâche au mot `continue` qui passe au prochain tour de boucle directement. Autrement dit, le programme compte le nombre de paires (j, i) telles que $j - i$ est impair. Les paires qui satisfont ce critère sont $(1, 0)$, $(3, 0)$, $(2, 1)$, $(4, 1)$, $(3, 2)$ et $(4, 3)$. Il y en a bien 6. Si on a de la difficulté à comprendre ce que fait la fonction, une alternative consiste à faire une trace détaillée en la présentant sous forme de tableau:

`c`	0		1		2			3		4		5			6
`i`	0					1				2			3		4
`j`	0	1	2	3	4	1	2	3	4	2	3	4	3	4	4
`j - i`	0	1	2	3	4	0	1	2	3	0	1	2	0	1	0
`(j - i) % 2`	0	1	0	1	0	0	1	0	1	0	1	0	0	1	0

Plus spécifiquement, chaque ligne correspond à une expression apparaissant dans le programme et à l'évolution des valeurs de ces expressions au fur et à mesure. On laisse l'entrée du tableau vide si la valeur est restée inchangée.