Identification /5

Prénom: .....

Nom: .....

Code permanent: .....

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

- `sizeof(char) = 1`,
- `sizeof(int) = 4`,
- `sizeof(double) = 8` et
- `sizeof(T*) = 8` peu importe le type `T`.

Vous pouvez aussi supposer qu'aucun alignement n'est effectué par le compilateur.

Considérez le programme suivant:

```
#include <stdio.h>

struct S {
  double x;
  int y[4];
};
union U {
  struct S* s;
  int t;
  char* c[2];
};

int main(void) {
  printf("%ld\n%ld\n", sizeof(struct S),
                       sizeof(union U));
  return 0;
}
```

Q1. Qu'est-ce qui sera affiché sur la 1re ligne de la sortie standard lors de son exécution?

☐ 8 ☐ 16 ■ 24 ☐ 32

Q2. Qu'est-ce qui sera affiché sur la 2e ligne de la sortie standard lors de son exécution?

☐ 8 ■ 16 ☐ 24 ☐ 32

Q3. Considérez le programme suivant:

```
#include <stdio.h>
#define i 1

int main(void) {
  int i = 0;
  printf("%d\n", 2 * i);
  return 0;
}
```

Qu'est-ce qui se passera si on compile et on exécute le programme?

☐ La valeur `0` sera affichée
☐ La valeur `2` sera affichée
■ Le programme ne compilera pas
☐ Le programme plantera à l'exécution

Q4. Considérez le test Bats suivant:

```
@test "Un test Bats" {
  run echo 'abracadabra'
  # TODO
}
```

Par quoi doit-on remplacer `# TODO` pour que le test réussisse?

☐ `assert_failure`
☐ `assert_line --index 0 'abra'`
■ `assert_output --partial 'brac'`
☐ `assert_output --regexp 'd.*d'`

Q5. Considérez la fonction `p` suivante:

```
bool p(const char* s) {
  for (int i = 0; s[i] != '\0'; ++i)
    if (!isdigit(s[i]))
      return false;
  return true;
}
```

Parmi les identifiants suivants, lequel est le plus approprié pour remplacer `p`?

☐ `is_digit`
☐ `str_has_digit`
■ `str_has_only_digits`
☐ `str_has_no_digit`

Voir explications page suivante

Q1. Comme on suppose qu'il n'y a pas d'alignement, l'espace mémoire occupé par une structure peut être obtenu en calculant la somme des espaces mémoire occupés par chacun de ses champs. On obtient donc

$$\texttt{sizeof(struct S)} = \texttt{sizeof(double)} + 4 \cdot \texttt{sizeof(int)} = 8 + 4 \cdot 4 = 8 + 16 = 24.$$

Q2. L'espace mémoire occupé par une union peut être obtenu en calculant le maximum des espaces mémoire occupés par chacun de ses champs. On obtient donc

$$\texttt{sizeof(union U)} = \max(\texttt{sizeof(struct S*)}, \texttt{sizeof(int)}, 2 \cdot \texttt{sizeof(char*)}) = \max(8, 4, 16) = 16.$$

Q3. La directive `#define i 1` remplace toutes les occurrences de `i` (en tant que mot) par `1`. Cela revient donc à tenter de compiler le programme suivant:

```
#include <stdio.h>

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

Ce programme ne compile pas, car l'instruction `int 1 = 0` est invalide.

Q4. Le test `assert_failure` échoue toujours. Le test `assert_line --index 0 'abra'` échoue: il ne réussit que si la première ligne (les indices commencent à 0) est exactement la chaine `abracadabra`, ce qui n'est pas le cas. Le test `assert_output --regexp 'd.*d'` échoue: il ne réussit que si l'expression régulière `d.*d` (le caractère `d`, suivi de 0 répétition ou plus d'un caractère quelconque, suivi du caractère `d`) apparaît sur la sortie standard, ce qui n'est pas le cas ici puisque le caractère `d` n'apparaît qu'une seule fois. Le test `assert_output --partial 'brac'` réussit si `brac` apparaît comme sous-chaîne (caractères consécutifs) sur la sortie standard, ce qui est bien notre cas ici.

Q5. La fonction `p` parcourt la chaîne de caractères `s` jusqu'à ce qu'elle trouve un caractère qui n'est pas un chiffre et elle retourne alors `false`. Sinon, elle retourne `true`. Autrement dit, `p` retourne `true` si et seulement si la chaîne `s` ne contient que des chiffres. Le nom `is_digit` n'est pas très précis, notamment parce qu'il ne prend pas en compte qu'il faut que *tous* les caractères soient des chiffres (il est plus adapté pour un seul caractère que pour une chaîne de caractères). Les noms `str_has_digit` (est-ce que la chaîne contient un chiffre?) et `str_has_no_digit` (est-ce que la chaîne ne contient aucun chiffre?) ne sont pas appropriés, car ils ne reflètent pas le comportement de la fonction. C'est donc `str_has_only_digits` (est-ce que la chaîne a seulement des chiffres?) qui est le plus adapté.