Identification

Prénom: .....

Nom: .....

Code permanent: .....

Veuillez choisir **une seule réponse**, avec une croix (×) ou un crochet (✓). Pour chacune des questions, vous pouvez supposer que:

- `sizeof(int)` retourne 4;
- `sizeof(char)` retourne 1;
- `sizeof(bool)` retourne 1 et
- `sizeof(T*)` retourne 8, peu importe le type `T`.

Q1. Qu'est-ce qui sera affiché sur la sortie standard par le programme suivant?

```
#include <stdio.h>
#include <string.h>
int main(void) {
  char s[5] = "abc";
  s[2] = '\0';
  strncat(s, "defg", 2);
  printf("%s\n", s);
  return 0;
}
```

- ■ `abde`
- ☐ `abcde`
- ☐ `abdef`
- ☐ Le comportement est indéterminé

Q2. Parmi les signatures de fonction suivante, laquelle est valide pour la fonction `main`?

- ☐ `int main(int argc, char* argv)`
- ☐ `int main(int* argc, char* argv[])`
- ■ `int main(int argc, char* argv[])`
- ☐ `int main(int* argc, char argv[][])`

Q3. Soit la déclaration suivante:

```
struct S {
  bool b[8];
  int* i;
  const char* s[4];
};
```

Quel sera le résultat de `sizeof(struct S)`

☐ 4 ☐ 16 ☐ 20 ■ 48

Q4. Soit la déclaration suivante:

```
union U {
  bool b[8];
  char c[8];
  int i[2];
  int* j;
};
```

Quel sera le résultat de `sizeof(union U)`

☐ 4 ■ 8 ☐ 16 ☐ 32

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

```
#include <stdio.h>
void f(int* i) {
  *i *= (*i) * (*i);
}
int main(void) {
  int i = 2; f(&i); printf("%d\n", i);
  return 0;
}
```

- ☐ 2
- ☐ 4
- ■ 8
- ☐ Le comportement est indéterminé

Q6. Toutes les recommandation suivantes font partie des 7 recommandations de Chris Beam, sauf une. Laquelle?

- ☐ Commencer le sujet par une lettre majuscule
- ■ Commencer le sujet par un nom
- ☐ Ne pas terminer le sujet par un point
- ☐ Limiter le sujet à 50 caractères maximum

Q7. Laquelle des sous-commandes Git suivantes n'entraîne jamais de conflit?

- ☐ `merge`
- ■ `push`
- ☐ `pull`
- ☐ `rebase`

Q8. Quelle est la première étape de la méthodologie *test-driven development* (TDD)?

- ☐ Écrire un test qui réussit
- ■ Écrire un test qui échoue
- ☐ Réusiner le code
- ☐ Identifier un bogue

Q1. À l'initialisation, la variable `s` contient les caractères `a b c \0 ?? ??`, qui constituent une chaîne de caractères bien formée (`??` désigne un caractère non connu quelconque). À la ligne suivante, l'affectation `s[2] = '\0'` transforme `s` en `a b \0 \0 ??`, qui est toujours bien formée. Après l'appel de `strncat`, au plus 2 caractères de la chaîne `"defg"` sont concaténés, ce qui fait que `s` devient `a b d e \0`, qui est toujours bien formée. Dans tous les cas, aucune modification n'a entraîné d'écriture à l'extérieur des bornes de la chaîne de caractères, de sorte que le programme n'a pas un comportement indéterminé.

Q2. Le premier argument de la fonction `main` (`argc`) décrit le nombre d'arguments lors de l'invocation du programme (incluant la commande invoquée). Il s'agit donc d'une valeur de type `int` et non `int*`: il n'y a pas de raison que ce soit un pointeur, puisqu'on ne s'intéresse qu'à la valeur. Le deuxième argument est un tableau de chaînes de caractères. On peut donc utiliser le type `char**` ou `char*[]`, qui sont équivalents. Noter qu'en C, `argv[][]` n'est pas autorisé.

Q3. Il suffit d'additionner les tailles de chacun des champs de la structure. Le premier champ a une taille de $8 \times 1\text{o} = 8\text{o}$, car `sizeof(bool)` retourne 1o et le tableau est de taille 8. Le deuxième champ a une taille de 8o, car `sizeof(int*)` retourne 8o. Le troisième champ a une taile de $4 \times 8\text{o} = 32\text{o}$, car `sizeof(char*)` retourne 8o et le tableau est de taille 4. Au total, on a donc $8\text{o} + 8\text{o} + 32\text{o} = 48\text{o}$.

Q4. Il suffit de prendre la taille maximale de chacun des champs de l'union. Le premier champ a une taille de $8 \times 1\text{o} = 8\text{o}$, car `sizeof(bool)` retourne 1o et le tableau est de taille 8. Le deuxième champ a une taille de $8 \times 1\text{o} = 8\text{o}$, car `sizeof(char)` retourne 1o et le tableau est de taille 8. Le troisième champ a une taille de $4 \times 2\text{o} = 8\text{o}$, car `sizeof(int)` retourne 4o et le tableau est de taille 2. Le quatrième champ a une taille de 8o, car `sizeof(int*)` retourne 8o. Ainsi, `sizeof(union U)` retourne $\max(8\text{o}, 8\text{o}, 8\text{o}, 8\text{o}) = 8\text{o}$.

Q5. Lors de l'appel à la fonction `f`, l'adresse de la variable `i` est fournie, de sorte que la fonction `f` peut modifier le contenu pointé par `i`. Ce contenu peut être lu et modifié (car il n'y a pas le qualificatif `const`) à l'aide de l'opérateur de déréférencement `*`. Comme `*i` vaut 2 au début de la fonction, l'expression `*i *= (*i) * (*i)` est équivalente à `*i *= 2 * 2` qui est équivalente à `*i *= 4`, qui est équivalente à `*i = (*i) * 4`, qui est équivalente à `*i = 2 * 4`, qui est équivalente à `*i = 8`. Ainsi, c'est la valeur `8` qui sera affichée.

Q6. En français, la règle *Commencer le sujet par un nom* devrait être *Commencer le sujet par un verbe à l'indicatif.*

Q7. Les conflits surviennent lorsque Git tente de fusionner deux versions d'un même fichier. Ainsi, la sous-commande `merge` est la commande de base qui peut entraîner un conflit. La sous-commande `pull` est une combinaison de `fetch` et `merge`, elle peut donc entraîner un conflit aussi. Lors d'un rebasement, il est possible qu'on ait besoin de résoudre des conflits également, car on doit s'assurer que la branche qu'on rebase est compatible avec l'historique plus récent. Autrement dit, un rebasement est une suite de *commits* que Git tente de réappliquer un par un, en fusionnant à chaque fois avec la branche en construction. Finalement, on note que la sous-commande `push` n'entraîne jamais de conflit. Elle peut cependant échouer: si on tente de pousser des modifications et que les branches locales et distantes ne sont pas à jour, Git écrira un message d'erreur nous suggérant de faire une mise à jour, par exemple avec `pull`.

Q8. Les trois étapes de base de la méthodologie TDD sont, dans l'ordre, (1) écrire un test qui échoue, (2) modifier minimalement la base de code pour que le test réussisse et (3) réusiner la base de code sans changer le comportement du programme. Bien que l'identification d'un bogue ne soit pas une étape de base, la résolution de bogue s'inscrit naturellement dans la méthodologie TDD: si on identifie un bogue, une bonne façon de se convaincre qu'on l'a résolu est (1) d'écrire un test qui met en évidence le bogue en échouant alors qu'il devrait réussir, (2) corriger la base de code afin que le bogue ne survienne plus et (3) réusiner la base de code.