1. Simulez l'exécution du code suivant. Que fait-il ?

```c
int main(void)
{
    int a = 4, b = 2, c;
    c = a;
    a = b;
    b = c;
    return 0;
}
```

2. Écrivez un programme qui demande à l'utilisateur un nombre de **miles**, les convertit en **kilomètres** puis affiche le résultat.
   Guide : 1 *mile* $\approx$ 1.609 *km*.
3. Écrivez un programme qui permette d'échanger le contenu de trois variables entières **a**, **b** et **c**.

   Par exemple :

	a	b	c
État initial	1	2	4
État final	2	4	1

4. Dessinez l'arbre d'évaluation des expressions suivantes (un arbre par ligne, ignorez la première ligne). Combien valent **a** et **b** à la fin du code ?

```c
    int a = 0, b = 6;
    a = b * 7 + 2 * a;
    b = (a == 2*3*7) * (b % 7);
    b = !(b);
```

5. Écrivez des **expressions booléennes** représentant les énoncés suivants. On considère que toutes les variables nécessaires sont définies et initialisées.
   1. L'entier x est impair
   2. L'entier x est strictement plus petit que l'entier y
   3. Les entiers x, y et z sont dans l'ordre décroissant (non strict)
   4. Le caractère c est un "b majuscule"
   5. L'entier y est soit multiple de 5, soit multiple de 7, mais pas les deux.
   6. Le caractère b est une lettre minuscule de l'alphabet
   7. Le caractère b est une lettre de l'alphabet (minuscule ou majuscule)
   8. Le caractère b n'est **pas** une lettre de l'alphabet (ni majuscule, ni minuscule)
   9. Faites la table de vérité de l'expression booléenne suivante :
      (!a || !b) && (a || b)
   10. Que représente l'expression précédente ?
   11. Développez la négation de l'expression booléenne suivante :
       (i >= 0) && (i < 10) && ((i % 2) || (i % 3))