1. Écrire une fonction de prototype `int indice(int t[], n)` qui renvoie un indice i tel que `t[i] = n` si un tel entier existe. Sinon, la fonction renvoie -1.

2. On suppose maintenant que le tableau t est trié par ordre croissant. Il va alors de soi que la recherche d'un indice tel que `t[i] = n` doit pouvoir être accélérée. Implémenter l'algorithme par recherche dichotomique : poser inf=0 et sup=MAX où MAX est le plus grand indice d'un élément du tableau. Tant que $t[\text{sup}] > t[\text{inf}]$, calculer $m = (\text{sup} + \text{inf})/2$, comparer $t[m]$ à n et remettre à jour inf et sup en fonction du résultat. À tout moment, on doit avoir $t[\text{inf}] \leq n \leq t[\text{sup}]$.

Exercice 1

QCM

Attention : Pour chaque question, une et une seule réponse est correcte. Une bonne réponse rapporte des points, une mauvaise réponse coûte des points, l'absence de réponse est neutre.

1. Dans la ligne de programme `printf("Le nombre vaut %i.", a) ;`
   a. `%` est un opérateur permettant de calculer un reste
   b. On utilise `%i` car a est de type `int`
   c. Il y a une faute de syntaxe
   d. Il y a une faute d'orthographe

   2
2. L'expression `a<=1`
   a. Réalise une affectation
   b. Diminue de 1 la valeur de a
   c. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
   d. Utilise les opérateurs `<` et `=`

3. `if (a<5) printf("Bonjour") ;`
   `a=a+1 ;`
   a. Affiche bonjour et augmente la valeur de a quelque soit a
   b. Augmente la valeur de a quelque soit a
   c. Affiche bonjour quelque soit a
   d. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a

4. Après `char c ; c='a' ; c=c+1 ;`
   a. c vaut `'A'`
   b. c vaut `'a'`
   c. c vaut `'b'`
   d. Un message d'erreur s'affiche

5. `printf("%c", 'A')`
   a. Affiche le caractère `'A'`
   b. Affiche le code ASCII du caractère `'A'`
   c. Affiche le code ASCII du caractère stocké dans la variable A
   d. Affiche le caractère dont le code ASCII est stocké dans la variable A

6. `printf("%i", 'A')`
   a. Affiche le caractère `'A'`

b. Affiche le code ASCII du caractère `'A'`
c. Affiche le code ASCII du caractère stocké dans la variable A
d. Affiche le caractère dont le code ASCII est stocké dans la variable A

7. `printf("%c", A)`
   a. Affiche le caractère `'A'`
   b. Affiche le code ASCII du caractère `'A'`
   c. Affiche le code ASCII du caractère stocké dans la variable A
   d. Affiche le caractère dont le code ASCII est stocké dans la variable A

8. `printf("%i", A)`
   a. Affiche le caractère `'A'`
   b. Affiche le code ASCII du caractère `'A'`
   c. Affiche le code ASCII du caractère stocké dans la variable A
   d. Affiche le caractère dont le code ASCII est stocké dans la variable A

9. `if (a%2) printf("bonjour");`
   a. Affiche bonjour quand a est un entier pair
   b. Affiche bonjour quand a est un entier impair
   c. Déclenche le message d'erreur `invalid lvalue in assignment`
   d. N'affiche rien (quelque soit le type et la valeur de a)

10. `if (a%2 == 0) printf("bonjour");`
    a. Affiche bonjour quand a est un entier pair
    b. Affiche bonjour quand a est un entier impair
    c. Déclenche le message d'erreur `invalid lvalue in assignment`
    d. N'affiche rien (quelque soit la valeur de a)

11. `if (a%2 = 0) printf("bonjour");`
    a. Affiche bonjour quand a est un entier pair
    b. Affiche bonjour quand a est un entier impair
    c. Déclenche le message d'erreur `invalid lvalue in assignment`
    d. N'affiche rien (quelque soit la valeur de a)

12. En langage C, `++` est
    a. Un opérateur de décrémentation
    b. Un opérateur d'accrétion
    c. Un opérateur d'incrémentation
    d. Un opérateur de décrétinisation