Université de Nice-Sophia Antipolis
Polytech'Sophia

Université de Xidian
2026–2027

Examen de Langage C

Durée : 1h30
Aucun document autorisé – Mobiles interdits

Notez que les affirmations (antécédents, conséquents, rôles et invariants) dans vos codes C entreront pour partie dans la note finale.

▶ 1. Combien de mode(s) transmission des paramètres existe(nt) en C ? *Cochez une seule réponse* !

- ☐ un seul, la transmission par référence.
- ☐ un seul, la transmission par valeur.
- ☐ deux, la transmission par valeur et référence.
- ☐ zéro.

Question sur 1 pt Réponse : **un seul** mode de transmission **par valeur**.

.....

▶ 2. Dans l'appel de la fonction `printf("%d\n", x)`, il y a :

- ☐ un paramètre *donnée* et un paramètre *résultat* ;
- ☐ deux paramètres *données* ;
- ☐ trois paramètres *données* ;
- ☐ deux paramètres *résultats*.

Question sur 1 pt Réponse : deux paramètres *donnés*.

.....

▶ 3. Écrivez en C la procédure `init` qui prend en paramètre un tableau `tab` d'entiers (**int**) et son nombre d'éléments `n` et qui initialise le tableau de façon aléatoire avec la fonction `rand`.

Question sur 2 pts

```c
/*
 * antécédent : n>0 et tab tableau de n entiers
 * conséquent : tab est initialisé de façon aléatoire
 */
void init(const int n, int tab[]) {
  assert(n>0);
  for (int i=0; i<n; i++)
    tab[i]=rand();
  //
```



```c
}
```

.....

▶ 4. Écrivez en C la fonction `nbImpairs` qui prend en paramètre un tableau `tab` d'entiers (**int**) et son nombre d'éléments `n` et qui renvoie le nombre d'entiers impairs contenus dans le tableau.

Question sur 3 pts

```c
/*
 * antécédent : n>0 et t tableau de n entiers
 * rôle : renvoie le nb d'entiers pairs contenus dans t
 */
int nbImpairs(const int n, const int tab[]) {
  assert(n>0);
  int nbP=0;
  for (int i=0; i<n; i++)
     if ((tab[i]&1)==1)
        // t[i] est un entier impair
        nbP++;
  // nbP est le nombre d'entiers impairs contenus dans tab
  return nbP;
}
```

.....

▶ 5. En utilisant les fonctions `init` et `nbImairs` précédentes, écrivez en C la fonction `main` qui :
- lit sur l'entrée standard un entier `n` (> 0, à vérifier) ;
- déclare et initialise un tableau d'entiers `tab` de `n` éléments ;
- écrit sur la sortie standard le nombre d'entiers impairs contenus dans `tab`.

Question sur 3 pts

```c
int main(void) {
  int n;
  scanf("%d", &n);
  assert(n>0);
  // n>0
  int tab[n];
  // initialiser aléatoirement le tableau tab
  init(n, tab);
  // écrire sur la S/S le nb d'entiers pairs contenus dans tab
  printf("%d\n", nbPairs(n, tab));

  return EXIT_SUCCESS;
}
```

.....

▶ 6. Écrivez la fonction booléenne `estPremier` qui teste si son paramètre `n` (un **int** $\geqslant 2$) est premier ou non. *Rappel* : un nombre premier admet uniquement deux diviseurs *distincts* 1 et lui-même. Pensez à minimiser le nombre d'itérations !

Question sur 5 pts

```
/*
 * Antécédent : n⩾2
 * Rôle : renvoie true si n est premier et false sinon
 */
bool estPremier(const int n) {
  assert(n>1);
  const int rac2N = sqrt(n);
  for (int i=2 ; i<=rac2N ; i++) {
    if ((n%i) == 0)
      // i est un diviseur ⇒ n non premier
      return false;
    // Invariant : ∀k ∈ [2;i], n mod k ≠ 0
  }
  // Invariant : ∀k ∈ [2;⌊√n⌋], n mod k ≠ 0 ⇒ n est premier
  return true;
}
```

.....

▶ 7. Écrivez en C la procédure `transposer` qui transpose une matrice carrée `mat` d'entiers `n`×`n` par rapport à sa diagonale principale.

Par exemple, la transposée de la matrice 4×4 $\begin{vmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 0 & 1 & 4 & 5 \\ 8 & 0 & 1 & 3 \end{vmatrix}$ est égale à $\begin{vmatrix} 1 & 5 & 0 & 8 \\ 2 & 6 & 1 & 0 \\ 3 & 7 & 4 & 1 \\ 4 & 8 & 5 & 3 \end{vmatrix}$.

La matrice `mat` sera représentée par un tableau à 2 dimensions d'entiers (**int**).

Question sur 5 pts

```
/*
 * Antécédedent : n>0 et 0<=i<n et 0<=j<n
 * Rôle : échanger mat[i][j] et mat[j][i]
 */
void échanger(const int i, const int j, const int n, int mat[][n]) {
  int aux = mat[i][j];
  mat[i][j] = mat[j][i];
  mat[j][i] = aux;
}

/*
 *  Antécédent : n>0 et mat matrice carrée n × n à transposer
 *  Conséquent : mat est la matrice transposée
 */
void transposer(const int n, int mat[][n]) {
  assert(n>0);
  // on parcourt la demi-matrice inférieure de mat
  for (int i=1; i<n; i++) {
    // ∀k ∈ [0;i−1], mat[0..k][0..k] est transposée
    for (int j=0; j<i; j++)
      échanger(i, j, n, mat);
  }
}
```


.....