Université de Nice-Sophia Antipolis — POLYTECH
ELEC3 — 2022–2023

Examen de Info C

Durée : 1h
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. Expliquez de façon claire et synthétique la différence entre paramètre *formel* et *effectif*.

---

voir cours

.....

► 2. Expliquez de façon claire et synthétique la différence entre les énoncés *tantque*, *répéter* et *pourtout*.

---

voir cours

.....

► 3. Expliquez de façon claire et synthétique la notion d'*invariant de boucle*.

---

voir cours

.....

► 4. Écrivez la fonction booléenne `bissextile` qui teste si une année `a` (un **int**) passée en paramètre est une année bissextile ou non.

```
/* Antécédent: a année valide
 * Conséquent: bissextile = vrai si l'année a est bissextile, et
 *             faux sinon
 */
int bissextile(const int a) {
  return (a%4==0 && a%100!=0) || a%400==0;
}
```

.....

► 5. À partir de l'antécédent suivant :

```
// jour, mois et annee : 3 variables de type int qui représentent une date valide
```


écrivez en C le fragment de code qui calcule la date de la veille.

```
// jour, mois et annee : 3 entiers qui représentent une date valide
if (jour>1)
  jour--;
else
  // 1er jour du mois
  if (mois>1)
    // la veille est le dernier jour du mois précédent
    jour = joursDansMois(--mois, annee);
  else {
    // 1er jour de l'année ⇒ la veille est le 31/12 de l'année précédente
    jour = 31;
    mois = 12;
    annee--;
  }
// jour, mois et annee est la date de la veille
```

.....

► 6. Écrivez la fonction `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 !

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

.....

► 7. Écrivez la fonction `sommePremiers` qui lit sur l'entrée standard une suite de nombres entiers (des **int**$\geqslant 2$) et qui renvoie la somme de ses nombres premiers, à l'exclusion de tous les autres. Si le nombre lu n'est pas $\geqslant 2$, un message d'erreur sera écrit sur la sortie d'erreur standard. La lecture des entiers sur l'entrée standard s'achèvera lorsque l'entier -1 sera lu.

```
/*
 *  Rôle : écrit un message d'erreur sur la S/ES
 */
void erreur(const int n) {
  fprintf(stderr, "erreur : %d<2\n", n);
}
```

```c
/*
 * Rôle : lit une suite d'entiers sur l'E/S et renvoie la somme de ses
 *        nombres premiers. La lecture s'arrête lorsqu'on lit -1
 */
int sommePremiers(void) {
  int n, som=0;
  do {
    // lire le prochain entier sur l'E/S
    scanf("%d", &n);
    if (n!=-1)
      // additionner n s'il est premier
      if (n<2)
        // nombre invalide
        erreur(n);
      else
        if (estPremier(n))
          som+=n;
      // Invariant : som = somme des nombres premiers lus sur l'E/S
  } while (n!=-1);
  return som;
}
```

.....