Université de Nice-Sophia Antipolis
Polytech'Sophia

Université de Xidian
2024–2025

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.

On représente les coordonnées d'un point p du plan cartésien par un tableau de deux réels (**double**). Le premier élément du tableau est l'abscisse du point et le deuxième élément son ordonnée.

► 1. À l'aide de **typedef**, déclarez le type `point` pour représenter un tableau à deux éléments de type **double**.

```
typedef double point[2];
```

.....

► 2. Écrivez la procédure `initPoints` qui initialise de façon aléatoire un tableau de n points. Cette procédure a l'en-tête suivant :

```
void initPoints(const int n, point tp[]) {
```

```
/*
 * antécédent : n>=1,
 * rôle :  initialise le tableau de points tp de façon aléatoire
 */
void initPoints(const int n, point tp[]) {
  for (int i=0; i<n; i++) {
    tp[i][0] = drand48();
    tp[i][1] = drand48();
  }
}
```

.....

► 3. Écrivez en C la fonction `distance` qui renvoie la distance entre deux points. Les deux points, de type `point`, sont passés en paramètre.

```
/*
 * Rôle : renvoie la distance entre 2 de points du plan
 *        de coordonnées (x1,y1) (x2,y2)
 */
double distance(const point p1, const point p2) {
  return sqrt((p2[0]-p1[0])*(p2[0]-p1[0])+(p2[1]-p1[1])*(p2[1]-p1[1]));
}
```


.....

► 4. Écrivez la fonction `distMin` qui prend comme paramètre un point `p` et un tableau `tp` de `n` points et qui renvoie la distance minimale entre le point `p` et les points du tableau `tp`. Cette fonction a l'en-tête suivant :

```
double distMin(const point p, const int n, const point tp[])
```

```
/*
 * antécédent : n>=1, p un point et tp un tableau de n points
 * conséquent : renvoie la distance minimum entre p et les points de tp
 */
double DistMin(const point p, const int n, const point tp[])
{
  double dMin = distance(p, tp[0]);
  //
  for (int i=1; i<n-1; i++) {
    double d = distance(p, tp[i]);
    if (d<dMin) dMin<d;
    // ∀k ∈ [0, i], dMin<=tp[k]
  }
  // ∀k ∈ [0, n − 1], dMin<=x_k et i=n-1
  return dMin;
}
```

.....