Appello dell'8 febbraio 2006 ~ Esercizio 2 (punti 9)

Si consideri la classe ***InsPunti***, classe immutabile che rappresenta un insieme di punti (senza duplicazioni) in un piano. Ogni punto dell'insieme è univocamente individuato da un indice intero, compreso fra 0 e la cardinalità dell'insieme (meno uno). La classe fornisce, oltre a costruttori e produttori che non sono qui riportati, le primitive seguenti:

```
class InsPunti  {
        ......
        //restituisce la cardinalità dell'insieme
        int numPunti() {}

        //restituisce la distanza tra due punti dell'insieme, individuati dai loro indici i e j,
        //con 0 ≤ i,j < numPunti()
        float distanza(int i, int j){}

        //restituisce il diametro dell'insieme, cioè la distanza massima tra coppie di punti dell'insieme
        float diametro(){}
        ......
}
```

a – Si fornisca la specifica JML (pre- e post-condizione) del metodo diametro()

Soluzione

```
//@ requires this.numPunti() >= 2;
//@ ensures (\exists int i; 0<=i<this.numPunti();
                (\exists int j; 0<j<this.numPunti();
                \result == this.distanza(i,j) &&
                (\forall int x; 0<=x<this.numPunti();
                (\forall int y; 0<y<this.numPunti(); \result >= this.distanza(x, y) )) ))
float diametro(){}
```

b –Il REP per la classe viene scelto come segue:

Un array di float `coord`, con 2·n componenti (l'insieme contiene n punti), in cui viene messa la rappresentazione polare di ogni punto (modulo, angolo). I primi due elementi di `coord` memorizzano rispettivamente modulo e angolo del primo punto, il terzo e il quarto modulo e angolo del secondo punto, etc.

Scrivere in JML l'invariante di rappresentazione, considerando che l'insieme può essere vuoto ma non può contenere punti duplicati.

Soluzione

```
//@ private invariant coord != null &&
//@ (\exists int n ; n>=0 ; 2*n == coord.length &&    coord ha un numero pari di elementi
//@     (\forall int i; 0<=i<n; coord[2*i]>=0) &&      condizione sul modulo di tutti i punti
//@     (\forall int i; 0<=i<n; 0<coord[2*i+1]<=6.28 )&&      condizione sull'angolo di tutti i
punti
//@     (\forall int x; 0<=x<n;                                tutti i punti diversi tra di loro
//@             (\forall int y; 0<=y<n;
//@                     x !=y ==> coord[2*x] != coord[2*y] || coord[2*x+1] != coord[2*y+1] ) ));
```

c – Si fornisca la specifica JML del metodo boolean triangolo(int i, int j, int k){} per sapere se i tre punti, identificati dai loro indici e ipotizzati essere fra loro distinti, possono essere i vertici di un triangolo non degenere. A tal fine si ipotizzi che un triangolo sia non degenere se e solo se la lunghezza di ognuno dei suoi lati è minore della somma della lunghezza degli altri due lati.

Soluzione

Per verificare che i tre punti distinti non siano allineati, basta imporre le tre disuguaglianze triangolari.

```
//@ requires this.numPunti() >= 3 && 0<=i,j,k <numPunti() && i != j && j != k && k != i;
//@ ensures \result == (distanza(i,j) < distanza(i, k) + distanza(k, j) &&
                        distanza(j,k) < distanza(j, i) + distanza(i, k) &&
                        distanza(k,i) < distanza(k, j) + distanza(j, i) );
boolean triangolo(int i, int j, int k){}
```