```
Precondition: a>0, b>0, c>0 and a +b +c =180
Postcondition:
right, if a  =90 or b = 90 or c= 90;
 obtuse  if a > 90 or >=90 or c > 90
else acute
public enum TriangleType {RIGHT, ACUTE, OBTUSE, OBTUSE};

public TriangleType reportTriangle(double a, double b, double c){
        if (a==90||b==90||c==90){
            return TriangleType.RIGHT;
        } else
        if (a>90||b>90||c>90)
            return TriangleType.OBTUSE;
        else
            return TriangleType.ACUTE;
}
```

Problema:

Considera el siguiente código de cliente, que primero obtiene tres ángulos y llama a reportTriangle(a,b,c):

```
double a = ...;
double b = ...;
double c = ...;
TriangleType result = reportTriangle(a,b,c);
```

¿cuáles son los resultados para (90, 45, 45), (120, 40, 20) y (50, 60, 70)= , ¿qué sucede con (90, −45, 135) ?.
¿Quién es el responsable de este fallo, el proveedor o el cliente?.
Corrige este error.

Respuesta:

```
(90, 45, 45) -> TriangleType.RIGHT
(120, 40, 20) -> TriangleType.OBTUSE
(50, 60, 70) -> TriangleType.ACUTE
(90, −45, 135) -> TriangleType.RIGHT
```

El responsable del fallo es el cliente porque no cumplió con la precondición de que los ángulos deben ser positivos.

Pregunta: En el ejemplo anterior indica una violación de precondición.

Respuesta: Se viola la precondición de que b > 0 porque b = −45

Pregunta:

El siguiente método isVowel verifica si una letra determinada es una vocal. "Y" a veces se considera una vocal cuando aparece en palabras como cry, fly y sky.

```
Precondition: letter ∈ {'a'−'z', 'A'−'Z'}
Postcond: true if letter ∈ {'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'};
otherwise, false
boolean isVowel(char letter) {
      String vowels = "aeiouy&@";
      char ch = Character.toLowerCase(letter);
returns vowels.indexOf(ch) >= 0;
}
```

¿Qué sucede cuando letter = '@' ?
Se viola la precondición de que letter debe ser una letra.

Pregunta:

Considera int [ ] genRandomIntegers(int count) que devuelve una lista de enteros aleatorios.
Su precondición y postcondición son count >0 y list.length = count, respectivamente (list denota
el valor devuelto). El código del cliente es el siguiente:

¿Qué sucede si modificamos genRandomIntegers, mantenemos la precondición pero cambiamos la
postcondición a list.length=count-1?.

En ese caso se produciría un error cuando se intente accesar al último elemento de la lista
devuelta.

Pregunta:

Considera el siguiente método getCell() en el programa TicTacToe.

```
public Cell getCell(int row, int column){
    return grid [row][column];
}
```

El código se crea para pasar la prueba del primer criterio de aceptación (tablero vacío AC 1.1).

La precondición y postcondición se dan a continuación:

Precondición: 0 <= row < 3 and 0 <= column < 3
Postcondición: return 0
Después de pasar las pruebas para AC 1.2 y AC 1.3, el código se cambia a lo siguiente:

```
public int getCell(int row, int column){
    if(row >= 0 && row < 3 && column >=0 && column < 3){
         return grid[row][column];
      } else{
                  return -1;
     }
}
```

¿Cómo cambia la precondición y la postcondición en el ejemplo anterior?.

Precondición: ninguna
Postcondición: retorna grid[row][column] si row >= 0 && row < 3 && column >=0 && column < 3
             retorna -1 en otro caso