Rebelde

Rebelde es un problema que evalúa si el alumno es capaz hacer usos complejos de la recursividad en Karel demandándole que implemente la búsqueda de un camino. Este es un problema que difícilmente se puede resolver si no se tiene algo de experiencia en Karel. Además el problema requiere de tener cuidado en la implementación para guardar el camino que llevas recorrido y no volver a pasar por él.

Este es el problema con implementación más larga del examen y en el que se tiene que tener más cuidado. Tradicionalmente el lenguaje Pascal de Karel no soporta la doble recursión, esto es, no soporta que dos funciones se llamen recursivamente una a la otra. Sin embargo el lenguaje Java de Karel si lo soporta. Aunque la doble recursión en sí misma no representa una ventaja computacional, si representa una ventaja en codificación. La siguiente versión de Karel, la cual estará disponible al público para finales del año 2010 soporta doble recursión en Pascal mediante la palabra clave *define-prototipo-instruccion*. En las soluciones oficiales hay dos archivos de solución para este problema, una utilizando la gramática actual de Karel y la otra utilizando la nueva funcionalidad. El ahorro en código gracias a la doble recursión es de un 31%.

La solución oficial hace lo siguiente:

- Recoge los zumbadores en la posición inicial y memoriza el número.
- Avanza hacia la primera casilla del mapa y comienza a buscar la salida del mapa pasando siempre a la función *buscaSalida* el número de cambios que todavía se tienen que hacer.
- En la función *buscaSalida*, si se llega a un salida y ya no es necesario hacer cambios, el programa termina.
- Si no es salida, para cada posición
    - La marca con 5 zumbadores para que si vuelve a llegar a ella no la tome en cuenta para el camino.
    - Ejecuta la función que se indica en la instrucción llamando recursivamente a *buscaSalida*
    - Posterior a ejecutar la instrucción de la posición, si aún hay cambios restantes, aplica los tres cambios posibles, dependiendo de la instrucción y llama recursivamente a *buscaSalida*

      indicándole que el número de cambios restantes ha disminuido en uno.
    - Al finalizar la recursión, si aún no ha encontrado la salida, regresa el número original a la posición.