1.9

```
//*****************
// IMPORTANTE
//
// Ana Álava Papí
// E01
// E01
//
//**************************************************

#include <iostream>
#include <fstream>
#include <vector>
#include <assert.h>
#include <stdio.h>

//*******************************
// Aqui especificacion del algoritmo (precondición y postcondición)
//requires 0 <= n <= 1000
//ensures b == forall v: 0 <= v[n] && v[n]%2==0
// Explicación del diseño que incluya invariante del bucle principal de la función
// Inicializo dos variables (max y cont) a 0 siendo estas las que lleven la cuenta de los pares consecutivos acumulados y del máximo tramo de pares consecutivos hasta el momento
// Si el número (v[n]) es par se incrementará el contador en uno, si el número es impar se comprobará si mi contador actual supera al máximo tramo de pares consecutivos hasta el momento
//Acabado el bucle se devolverá el tramo más largo de pares
//Invariant 0 <= i < n
//Invariant 0 <= cont && 0 <= max

// Coste del algoritmo con justificación
//Coste O(n) siendo n todos los elementos del vector y siendo cada vuelta al vector constante
//
//*******************************


// Función en la que se resuelve el problema planteado
int resolver(std::vector<int> const& v) {
        int cont = 0;
        int max = 0;

        for (int i = 0; i < v.size(); ++i) {
                if (v[i] % 2 == 0) cont++;
                else {
                        if (cont > max)max = cont;
                        cont = 0;
                }
        }
        if (cont > max)max = cont;
        return max;
}


// Para lectura de datos y mostrar los resultados
```

según esto b es de tipo bool pero la función devuelve un entero

Este predicado es cierto cuando el vector completo es de números pares. No especifica un problema de segmento máximo

- 0.9

→ Este invariante no permite probar la corrección del bucle.

??

El bucle da n vueltas y en cada vuelta el coste es constante.

- 0.2

1.5

```
bool resuelveCaso() {
	// Lectura de datos
	int n; std::cin >> n;
	if (n == -1) return false;

	// leer el resto de los datos
	std::vector<int> v(n);
	for (int& m : v) std::cin >> m;

	// LLamar a la funcion que resuelve el problema
	std::cout << resolver(v) << "\n";

	// escribir el resultado



	return true;
}



int main() {
	// Para redireccionar la entrada de datos a un fichero
#ifndef DOMJUDGE
	std::ifstream in("E1.txt");
	auto cinbuf = std::cin.rdbuf(in.rdbuf());
#endif

	while (resuelveCaso()) {} //Resolvemos todos los casos


#ifndef DOMJUDGE // para dejar todo como estaba al principio
	std::cin.rdbuf(cinbuf);
	system("PAUSE");
#endif

	return 0;
}
```