Contents

1 Basic Test Results

```
Running...
Opening tar file
ex1.c
OK
Tar extracted O.K.
Checking files...
OK
Making sure files are not empty...
OK
Importing files
OK
Compilation check...
Compiling...
gcc ./ex1.c -c -o ./ex1.o -Wall -lm -Wvla
gcc ./ex1.o -o ./ex1 -Wall -lm -Wvla
OK
Running test...
OK
Compilation seems OK! Check if you got warnings!

========================
= Checking coding style =
========================
 ** Total Violated Rules      : 0
 ** Total Errors Occurs       : 0
 ** Total Violated Files Count: 0
```

2 ex1.c

```c
//---------------------------------------------------------------------------
//                                Ex1
// General    :    find if there is an integer number -x- at a given range of
//                 which F(x)=G(x)
// Input      :    2 monotonous functions, Range and Epsilon. may or may not given
//                 in "Definitions"
// Process    :    The search is like a binary-search using the fact F and G are
//                 monotonous.
//                  The complexity of binary-search is log(n) when n=the RANGE
// Output     :    The number x and the value F(x) iff exists.
//
//---------------------------------------------------------------------------

#include <stdio.h>
#include "Definitions.h"

// Macros and const definitions if not include in "Definitions.h"
#ifndef F
    #define F(x) (x + 0.5)
#endif

#ifndef G
    #define G(x) (10.5 - x)
#endif

#ifndef EPSILON
    #define EPSILON 0.0001
#endif

#ifndef RANGE
    #define RANGE 1000
#endif

#define HALF 2
/**
 *---------------------------------------------------------------------------
 *
 * General        :    find if there is an integer number -x- at a given range of
 *                     which F(x)=G(x)
 * Paraneters     :
 * Return value   :    the number x
 *
 *---------------------------------------------------------------------------
 */
int findEq()
{
    int max = RANGE;
    int min = 0;
    int cur;
    int j = RANGE;

    for( ; j > 0 ; j = j / 2)
    {
        cur = (max + min) / HALF;


        if ((G(cur) - F(cur)) > EPSILON)
        {
            min = cur;
```

```
60          }
61          else if ((G(cur) - F(cur)) < -EPSILON)
62          {
63              max = cur;
64          }
65          else
66          {
67              printf("%d\n%.3f\n", cur, F(cur));
68              return cur;
69          }
70      }
71
72      if (-EPSILON < (G(max) - F(max)) && (G(max) - F(max)) < EPSILON)
73      {
74          printf("%d\n%.3f\n", max, F(max));
75          return cur;
76      }
77      return -1;
78  }
79  /**
80   * The main function
81   */
82  int main()
83  {
84      findEq();
85      return 0;
86  }
```

-3/-3 You are using numeric literals (magic numbers) inside your logical code. You should use constant variables or (code='using_numeric_literals')