CS 569

Assignment I

**About the Program**

My program of implementation was the “Binary Search Algorithm. For the purpose of implementation there are two files, which are ‘mybinsearch.c’ and ‘harness.c’ respectively.

**Buggy Code**

Firstly, the Binary Search Algorithm did not prove to be completely correct. The code implemented the binary search, but it was dependent upon the condition of the elements in the array to be sorted in ascending order, which the code did not contain. The ‘harness.c’ file using CBMC was able to determine the specific assertion condition as “FALSE”, and further specifying the line number in the ‘harness.c’ file calling the ‘mybinsearch()’ function.

After which I implemented the simple property of arranging the elements in ascending order, which then correctly asserted the required output. Moreover, in order to verify whether the values of the elements had been considered as according to the assumptions I further introduced “assert (0);” and “| grep LOG” to generate the output.

**Ability to find (intentionally) introduced bugs - SUCCESS**

To test the capability of the harness file, I initialized the variable ‘p’ (which tests for the condition of the element not being present in the code). I *intentionally* introduced a bug. I specified the value of ‘p’ to be less than 0, and greater than the required SIZE. The elements in the array have been constrained to have values between 0 and 20. Now making ‘p’ less than 0 and greater than SIZE, does not make sense!

But the harness was able to prove the false condition of VERIFICATION FAILED by correctly identifying the failed assertion.

**Ability to find (unintentionally) introduced bugs - FAILED**

Furthermore, upon observation, I concluded that many elements values were repeated and I wanted the array to contain unique elements. This property was further introduced within the ‘harness.c’ file, making the elements completely random within the specified range and unique.

The bug occurred when I made the second loop test the condition while the elements in the array were being initialized. This made the code buggy, and the result generated as ‘VERIFICATION SUCCESSFUL’. Actually, this bug was introduced unintentionally in the code.

The ‘harness.c’ *failed* to detect the above bug. I would not have noticed the particular situation if I hadn’t introduced an “assert(0)” in various parts of the harness. It happened that, after introducing the “assert(0)” condition in the loop, it generated a VERIFICATION SUCCESSFUL, which is completely wrong; therefore making the code buggy. As “assert(0)” condition always gives a failed verification.

**Functionality that couldn’t be specified**

Furthermore, I wanted to test a condition when the DSIZE is 20; as the values assigned in the array are between the range greater than 0 and less than 20. Under the particular circumstances, as the elements size is greater than the required field, CBMC ran the file for about 30 mins, and it did not generate the required output.

**Scenario:** The –DSIZE=20 (Size of the array) and I did the unwinding for about 40 times. Therefore, the harness did fail to produce the required output in the particular situation. I was waiting for the index-out-of-bounds exception, but it failed to produce the desired output.

**Loop Bounds and without it!**

The different bounds check conditions for the DSIZE of 5 and the unwinding limit of 11, executed the program in about 1.216 seconds. But on increasing the DSIZE (for example –DSIZE=8), took maximum 12.415 seconds.

It was regardless of the bounds, the DSIZE controlled the overall execution time of CBMC execution.

Whereas, without explicitly specifying the “Bounds Check”, it code executes for about 3.59secs with a DSIZE of 5. But on the increase of DSIZE, it further executes for about 30.315secs (DSIZE=8).