SE 636
Reflection HW2
Professor Colin Gordon
Himanshu Gupta

1) **Comparing property-based testing to blackbox testing, which seems easier to do for you at this time? (Many of you have prior experience that might influence your answer; that's okay)**

   This question depends on your knowledge of the system. This knowledge range from having a simple input and output information about the program to having info about the code and its path executions. In all of these scenarios, both the testing (property-based testing and the blackbox testing) have their own pros and cons. For this assignment, I do have the knowledge of code to its every minute detail. I have divided the pros and cons of Property-based testing over blackbox testing for this assignment, then decide which seems to be better.

   Property-based Testing over Blackbox Testing:

   1) Even though I did not have any experience with the PBT prior to this assignment, learning jqwik was easy and took only a few hours. With the power of generating random inputs for the parameters gives more confidence for the stability of the program as a developer.
   2) In the property tests for this assignment, I am generating sorted arrays of elements of random size up to 30. PBT ensured in this case that my binary search algorithm works for every size of the sorted array which blackbox testing can not achieve. This found that my algorithm is throwing index out of bound exception for null arrays. Making the fix was easy and took a few minutes.
   3) Jqwik framework generates random elements for a plethora of data types and data-structures. This lets me test my algorithm for a range of data and arrays easily.
   4) In black-box testing, I can only test my algorithm against the hard-coded arrays which do not cover my algorithm satisfyingly even though my test coverage is 100%. Think about how big of a problem this could be in a large distributed system.

   Blackbox Testing over Property-based Testing:

   1) For this assignment, black-box testing let me test the functionality of the algorithm straightway for every kind of the comparables.
   2) Through the black-box testing, we can confidently check tests against every requirement in a quick period of time.

From the pros and cons above, the Benefits of PBT overshadow the benefits of black-box tests. That's why I am convinced to say PBT seems easier to do for me this time.

2) Given a set of tests for some unfamiliar software components, which style of tests (concrete blackbox tests, or property-based tests) do you think would be most useful for you to understand the expected behavior of the software? (i.e., which would help you learn the specification from the tests, assuming you only had the tests, and most likely no comments)

Given the situation that I have no knowledge about the software component, Both the testing have their own pros and cons which are as follows.

Property-based testing over blackbox testing:

1) Although property-based tests would take a longer time to understand than the blackbox tests, PBT would be able to test the limit of our algorithm by testing it against a plethora of data types and data-structures. In my PBT tests, I am also testing for middle-indexed elements. This results in finding that my algorithm throws Index out of bounds for middle-indexed elements.
2) I do like to see how my algorithm performs in non-deterministic situations. Blackbox tests do not help you to get some sort of confidence in saying out the behavior of your algorithm in non-deterministic situations other than guessing wildly. PBT lets you test the behavior of your algorithm against those situations. You can analyze the pattern of outputs to better understand the behavior of the software.
3) Most of the time, developers believe their code is right in the first place. This perception causes developers to think from the perspective of code only while making tests. This ultimately does not cover the edge cases even though their tests might have 90% test coverage. With PBT, this problem does not exist.

Blackbox testing over Property-based testing:

1) In my cases, testers are only provided with the specifications of the program. From that point, blackbox tests can easily give you confidence about the functionalities of your program, and one can easily check if any requirement is missing. In the case of PBT, testers might find difficult to say the same for the program.

If I am using formal methods for writing formal specifications in every stage of software development, then I might question the importance of black-box tests. Overall, for this question as well, the benefits of PBT overshadow the benefits of blackbox testing.

3) If the specification for search changed in the future, which style of the test suite (blackbox or property-based) would most likely be easier/simpler to update for the new specification? This is a speculative question, and not specific to a particular change; example changes to consider include changing it to assume the array is sorted in decreasing order rather than increasing order, or to take a set of elements to search for and return true if all of them are present, though you shouldn't assume it's necessarily one of those particular changes.

For the situations where specifications changed for search in the future, I can divide the situations into two broad categories of technical and non-technical changes. To choose which one would be easier/simple to update the new specification, I am mentioning some of the situations in both the categories.

Technical changes:

1) If the change is to return the index of the element in the array, then we have to change all the tests which are index-based like first-half, last-half, and middle elements in Property-based testing. While blackbox testing does not test in this way so it would be easier to update this new specification in blackbox testing.
2) In the case where we are checking for all the elements between some range, then PBT would be easier to add tests for our new specification. This is because PBT already uses a range of elements to test our algorithm. We can easily manipulate that range. However, in the blackbox testing, we have to change all of our hard-coded input arrays to include the tests for our new range. This would be more time-consuming than that in PBT.
3) In the changes made for performance improvements, blackbox tests would be incapable of testing this new specification easily. However, we can check our change in performance using PBT for a large set of elements easily.

Non-technical changes:

1) In the changes like we are adding another new comparable type for binary search, then it would be easier to add this specification in black-box testing as it does not test the implementation of code, instead of test the new functionality. In PBT, we have add all the properties for our new specification which would be more time-consuming and harder than that in blackbox testing.
2) In the change where if an element does not exist in the array, add that element in the array. For this change, adding blackbox tests would be easier as we can just check the size of the new array if the element not present in the array. However, in the PBT, we have to change all of our existing properties to accommodate this specification.