Practical 03
Search

1. Khan Academy - Algorithms (Binary Search)

   ***Background:** Khan Academy offers practice exercises, instructional videos, and a personalized learning dashboard that empower learners to study at their own pace in and outside of the classroom. We tackle math, science, computer programming, history, art history, economics, and more. Our math missions guide learners from kindergarten to calculus using state-of-the-art, adaptive technology that identifies strengths and learning gaps. We've also partnered with institutions like NASA, The Museum of Modern Art, The California Academy of Sciences, and MIT to offer specialized content.*

   Khan Academy has partnered with Dartmouth college professors Tom Cormen and Devin Balkcom to teach introductory computer science algorithms, including searching, sorting, recursion, and graph theory. Learn with a combination of articles, visualizations, quizzes, and coding challenges.

   a. Visit the following website (*https://www.khanacademy.org/computing/computer-science/algorithms#binary-search*) to access Khan Academy's online instructional content on Algorithms (Binary Search). Go through the following instructional activities from the website:

      - Binary Search
      - Implementing binary search of an array
      - Challenge: Binary search
      - Running time of binary search

   b. With reference to the Khan Academy's instructional content on Algorithms (Binary Search), answer the following questions:

      i. What is mathematical function that means the same thing as the number of times we repeatedly halve, starting at n, until we get the value 1? For example, we need to repeatedly halve 32 for 5 times until we get the value 1: 32 ➔ 16 ➔ 8 ➔ 4 ➔ 2 ➔ 1.

      ii. The ____________ function grows very slowly. ____________ are the inverse of ____________ function, which grows very rapidly.

      iii. Complete the following pseudocode for the Binary Search:

```
1. Let min = 0 and max = n-1.
2. If ________, then stop: target is not present in array. Return -1.
3. Compute guess as the ________ of max and min, rounded down (so that it is an integer).
4. If array[guess] equals target, then stop. You found it! Return guess.
5. If the guess was too low, that is, array[guess] < target, then set min = ________.
6. Otherwise, the guess was too high. Set max = ________.
7. Go back to step 2
```

2. Sequential Search

The following code is an implementation of the Sequential Search for an unsorted list of values:

```
def sequentialSearch( theValues, target ):
    n = len( theValues )

    for i in range(n):
        # If the target is in the ith element, return True
        if theValues[i] == target:
            return True

    return False    # If not found, return False
```

Write a new function named – `sortedSequentialSearch()`, that improves the above implementation (with fewer comparisons needed) for a sorted list of values.

3. Binary Search

The following code is an incomplete implementation of the Binary Search:

```
def binarySearch( theValues, target ):
    # Start with the entire sequence of elements
    low = 0
    high = ____________________

    # Repeatedly subdivide the sequence in half
    # until the target is found
    while _______________:
        # Find the midpoint of the sequence
        mid = _______________

        # Does the midpoint contain the target?
        # If yes, return midpoint (i.e. index of the list)
        if ____________________:
            return mid
        # Or is the target before the midpoint?
        elif target < theValues[mid]:
            _______________
        # Or is the target after the midpoint?
        else:
            _______________

    # If the sequence cannot be subdivided further,
    # target is not in the list of values
    return -1
```

a. Complete the implementation by providing the rest of the required codes.

b. Modify the code to return the position of the **first occurrence** of a value that can occur multiple times in the sorted list of values.

-- End of Practical --