Exercise 1 – Searching Algorithms (33 Marks)

This is a simple program that provides two functions which, when given an integer list and an integer number, return the index of that number (if it is in the list). The first function uses a linear search algorithm to find the given integer, while the second function uses a binary search algorithm.

A program designed to test these functions is shown below (and is provided electronically). Study the code and try to understand what is happening in the program, before attempting the questions that follow.

```
1  x = 4
2
3  def linearSearch(searchList, searchVal):
4      for i in searchList:
5          if i == searchVal:
6              return i
7      return Value not found
–
9  def binarySearch(searchList, searchVal):
10     start = 0
11     end = len(searchList) - 1
12     while start <= end:
13         mid = (start + end) // 2
14         if searchList [mid] == searchVal:
15             return mid
16         elif searchList [mid] < searchVal:
17             start = mid
18         elif searchList [mid] > searchVal:
19             end = mid
20     return Value not found
–
22 searchList = [1,2,3,4,5,6,7,8,9,10]
23 print(linearSearch(searchList, x))
24 print(binarySearch(searchList, x))
25 input()
```

Section A

A 1 Give a line number from the program that contains a function call. **[1]**

.....

A 2 Give a line number from the program that contains a global variable. **[1]**

.....

A 3 Explain why, given a choice of both, a binary search is often preferably to a linear search. **[1]**

.....

.....

A 4 Explain why some lists are not searchable with a binary search algorithm. **[1]**

.....

.....

A 5 The program as it stands does not run and produces a syntax error.
Explain the cause of this error. **[1]**

.....

.....

A 6 The `linearSearch` function returns the incorrect index.
Explain the cause of this error. **[1]**

.....

.....

A 7 The `binarySearch` function does not return if it tries to find the final element in a list.
Explain the cause of this error. **[2]**

.....

.....

.....

.....

A 8 Explain what is meant by the *time complexity* of an algorithm. **[2]**

.....

.....

.....

A 9 State the time complexity of the linear search and binary search algorithms using Big-O notation. **[2]**

.....

.....

.....

A 10 The binary search algorithm can be implemented using recursion.
Explain why a recursive version of the binary search algorithm may not be suitable for a large list. **[2]**

.....

.....

.....

Section A: /14

Section B

B 1 Modify the program to remove the syntax error. **[1]**

Program updated ☐

B 2 Modify the program so that the `linearSearch` function returns the correct index. **[1]**

Program updated ☐

B 3 Modify the program so that the `binarySearch` function returns even when searching for the final element in the given list. **[1]**

Program updated ☐

B 4 Modify the program to add a `recursiveBinarySearch` function that takes a list, a search value and an index for the start of the list and uses binary search to return the index of the search value (if it is in the list), or returns the string `"Value not found"` otherwise. This function should use recursion. The main program procedure should be updated to call this procedure and print the result. **[5]**

Program updated ☐

B 5 Modify the program to add a `getVal` function that asks the user for an integer and returns the given integer. This function should take no arguments and be able to handle both valid and invalid user input. The main program procedure should be updated to call this function to set the value of `x` for the search algorithms. **[4]**

Program updated ☐

B 6 Modify the program to add a `generateList` function that is given a positive integer as input and returns an ordered list of all positive integers from 1 to the given value. The main program procedure should be updated to call this procedure to create the `list` variable of a length given by the user. **[2]**

Program updated ☐

B 7 Modify the program to compare the time efficiency of the `linearSearch` and `binarySearch` functions. The `linearSearch` and `binarySearch` functions should be modified to include a `count` variable that increments by 1 every time a new element is checked, and return `count` when the search value is found, or when it has been determined that the search value is not in the list.

A `test` function should be added that takes two integer values, `n` for the length of list and `tests` for the number of tests, and returns the average result of `tests` calls of `linearSearch` and `binarySearch` on lists generated by `generateList` of length `n`.

The main program procedure should be modified to call `test` for lists of length 10, 100, 1,000, 10,000 and 100,000, performing 1,000 tests for each, and display how much longer the `linearSearch` took in comparison to `binarySearch` on average for lists of the given size. **[5]**

Program updated ☐

Section B: /19

