Zig Zag Education

Computer Science AS & A Level | AQA | 7516 & 7517

Python Exercises

for AS & A Level AQA Computer Science

Download support files from zzed.uk/productsupport

zigzageducation.co.uk **POD 9945**

Contents

Teacher's Introduction

This resource is designed to support the development of students' programming skills at KS5 (and build on the KS4 equivalent resources published by ZigZag Education). It contains 10 unique exercises, featuring a range of scenarios that develop the core programming principles.

These include programming constructs, recursion, global and local variables, modularity, debugging programs, object-oriented techniques, divide-and-conquer algorithms, data structures and standard algorithms – all skills that are found in the AQA AS & A Level Computer Science specifications.

Each exercise contains a combination of questions and tasks, and consists of two sections – Section A and Section B.

- The purpose of **Section A** is to test students' understanding of the skeleton code; both in terms of explaining features of the code but also in identifying flaws that are present in it.
- **Section B** provides students with the opportunity to debug the issues in the code (an incredibly important skill as programmers rarely tend to write whole programs by themselves) as well as develop the functionality further. Section B should take longer than Section A to complete and will help prepare students for their NEA and any other practical assessment, such as AQA Paper 1.

Along with the worksheets, there are Python v3.6 programs that should be changed as the questions have been answered. Working Python files are provided for every worksheet, along with written answers.

Note that credit should also be given for any valid responses that are not explicitly included in this resource.

IMPORTANT – BEFORE YOU START

The skeleton code for each exercise plus the modified scripts (showing all of the changes completed) are provided on the ZigZag Education Product Support system.

This can be accessed via **zzed.uk/productsupport**

AQA Specification Map

	1	2	3	4	5	6	7	8	9	10
	Searching Algorithms	Sorting Algorithms	Towers of Hanoi	Sorting Queues	Draughts	Tree Traversal	Dijkstra's SPA	Bomb Search	Dictionaries & Hash Tables	Reverse Polish
1.1.1 – Data types	✓	✓	✓	✓	✓		✓	✓	✓	✓
1.1.2 – Programming concepts	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.1.3 – Arithmetic operations	✓	✓		✓	✓	✓	✓	✓	✓	✓
1.1.4 – Relational operations	✓	✓	✓	✓	✓		✓	✓	✓	✓
1.1.5 – Boolean operations	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.1.6 – Constants and variables	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.1.7 – String handling	✓	✓	✓		✓		✓	✓	✓	✓
1.1.8 – Random numbers	✓							✓		
1.1.9 – Exception handling	✓	✓			✓			✓		✓
1.1.10 – Subroutines		✓	✓	✓	✓	✓	✓	✓	✓	✓
1.1.11 – Parameters	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.1.12 – Returning values	✓	✓	✓		✓	✓	✓	✓	✓	✓
1.1.13 – Local variables	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.1.14 – Global variables	✓								✓	✓
1.1.16 – Recursive techniques	✓	✓				✓				✓

(Continues on next page)

(Continued from previous page)

	1	2	3	4	5	6	7	8	9	10
	Searching Algorithms	Sorting Algorithms	Towers of Hanoi	Sorting Queues	Draughts	Tree Traversal	Dijkstra's SPA	Bomb Search	Dictionaries & Hash Tables	Reverse Polish
1.2.1 – Programming paradigms	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
1.2.2 – Procedural paradigms	✓	✓							✓	
1.2.3 – Object-oriented paradigms			✓	✓	✓	✓	✓	✓		✓
2.1.2 – Arrays	✓	✓	✓	✓	✓		✓	✓	✓	✓
2.1.4 – Data structures			✓	✓		✓	✓			✓
2.2.1 – Queues				✓						
2.3.1 – Stacks			✓							✓
2.4.1 – Graphs							✓			
2.5.1 – Trees						✓				✓
2.6.1 – Hash tables									✓	
2.7.1 – Dictionaries									✓	
3.1.1 – Graph-traversal						✓				
3.2.1 – Tree-traversal						✓				✓
3.3.1 – Reverse Polish Notation										✓
3.4.1 – Linear search	✓								✓	
3.4.2 – Binary search	✓									
3.5.1 – Bubble sort		✓								
3.5.2 – Merge sort		✓								
3.6.1 – Dijkstra's shortest path							✓			

EXERCISE 1 – SEARCHING ALGORITHMS

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

A program designed to test these functions is shown b[illegible]d is provided electr
to understand what is happening in the progra[illegible], before attempting the questions

```
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
8
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
21
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.

.....

A 2 Give a line number from the program that contains a global variable.

.....

A 3 Explain why, given a choice of both, a binary search is often preferable

.....

.....

A 4 Explain why some lists are not searchable with a binary search algorith

.....

.....

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

.....

.....

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

.....

.....

A 7 The `binarySearch` function does not return if it tries to find the fina
Explain the cause of this error.

.....

.....

.....

.....

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

.....

.....

.....

A 9 State the time complexity of the linear search and binary search algo

.....

.....

.....

A 10 The binary search algorithm can be implemented using recursion.
Explain why a recursive version of the binary search algorithm may no

.....

.....

.....

Section B

B 1 Modify the program to remove the syntax error.

Program updated ☐

B 2 Modify the program so that the `linearSearch` function returns the

Program updated ☐

B 3 Modify the program so that the `binarySearch` function returns ever element in the given list.

Program updated ☐

B 4 Modify the program to add a `recursiveBinarySearch` function th an index for the start of the list and uses binary search to return the in the list), or returns the string `"Value not found"` otherwise. This fu main program procedure should be updated to call this procedure an

Program updated ☐

B 5 Modify the program to add a `getVal` function that asks the user for a integer. This function should take no arguments and be able to handle input. The main program procedure should be updated to call this fun *the search algorithms.*

Program updated ☐

B 6 Modify the program to add a `generateList` function that is given a returns an ordered list of all positive integers from 1 to the given valu should be updated to call this procedure to create the `list` variable

Program updated ☐

B 7 Modify the program to compare the time efficiency of the `linearSe` functions. The `linearSearch` and `binarySearch` functions should variable that increments by 1 every time a new element is checked, an value is found, or when it has been determined that the search value is

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

The main program procedure should be modified to call `test` for lists and 100,000, performing 1,000 tests for each, and display how much l took in comparison to `binarySearch` on average for lists of the give

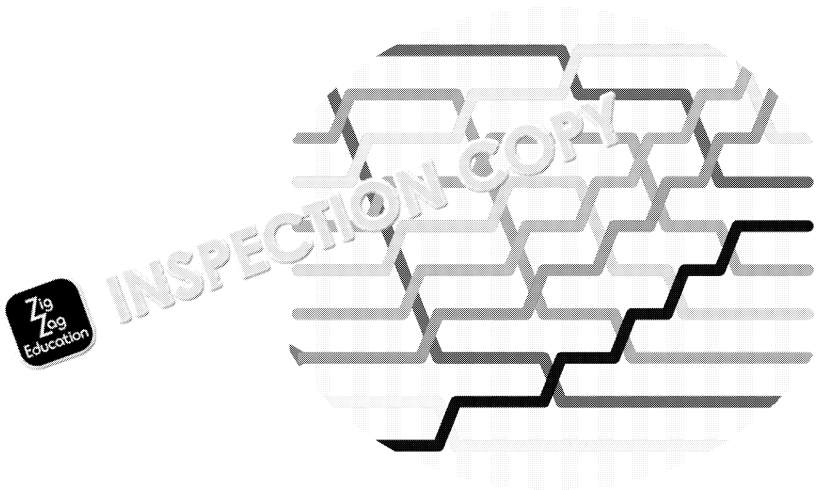
Program updated ☐

EXERCISE 2 – SORTING ALGORITHMS

This is a simple program that provides two functions that, when given an integer li
ascending order. The first function uses a bubble sort algorithm to sort the given li
uses a merge sort algorithm.

A program designed to test these functions is shown be[illegible] and is provided electr
to understand what is happening in the program, [illegible] attempting the questions

```
1   def bubbleSort(sortList):
2       sorted = False
3       length = len(sortList)
4       while not sorted:
5           for i in range(length - 2):
6               if sortList[i] > sortList[i+1]:
7                   sortList[i] = sortList[i+1]
8                   sortList[i+1] = sortList[i]
9                   sorted = False
10      return sortList
11
12  def mergeSort(sortList):
13      mid = len(sortList) // 2
14      leftHalf = sortList[:mid]
15      rightHalf = sortList[mid:]
16      if len(sortList) > 1:
18          mergeSort(leftHalf)
19          mergeSort(rightHalf)
20
21  numList = []
22  for i in range(10):
23      numList.append(int(input("Add an integer number to the list: ")))
24      numList.append(int(input()))
25  print("Bubble sort given:")
26  print(numList)
27  print("Bubble sort returns")
28  print(bubbleSort(numList))
29  input()
```



SECTION A

A 1 Give a line number from the program that contains a parameter.

.....

A 2 Give a line number from the program that contains recursion.

.....

A 3 Define 'parameter'.

.....

.....

A 4 The program does not run and produces a syntax error.
Explain the cause of this syntax error.

.....

.....

A 5 When the `bubbleSort` function is called, the program gets stuck in a
Explain the cause of this logic error.

.....

.....

A 6 Instead of swapping elements that are in the wrong order, the `bubble`
second element in the pair to the first element in the pair.
State the type of error this is, and explain the cause of the error.

.....

.....

.....

A 7 Currently, the program crashes if the user enters a non-integer value
to the list. This could be prevented by implementing exception handli
Explain what exception handling is and why it is necessary.

.....

.....

.....

.....

A 8 The merge sort algorithm is an example of a divide-and-conquer algo
Explain what a 'divide-and-conquer' algorithm is.

.....

.....

.....

.....

A 9 th[illegible] complexity of the bubble sort and merge sort algorithr

.....

.....

.....

.....

A 10 As well as taking different amounts of time, different algorithms also r
space in memory. Explain which of the two sorting algorithms has the

.....

.....

.....

.....

SECTION B

B 1 Modify the program to remove the syntax error.

Program updated ☐

B 2 Modify the program so that the `bubbleSort` function does not get s

Program updated ☐

B 3 Modify the program so that `bubbleSort` correctly swaps elements th

Program updated ☐

B 4 Modify the program so that the program does not crash if the user en prompted to add a number to the list. Your solution should display a m when they have entered a non-integer number and keep asking the us valid integer. The input should terminate once they enter a blank (just

Program updated ☐

B 5 Currently, the code that asks the user to enter the numbers in the list i program procedure, and so cannot be easily reused.

Modify the program so that this code is moved into a new `getList` f and returns the resulting list. `getList` should be called in the main pr should stop asking for input when a 'blank' number is entered (i.e. the

Program updated ☐

B 6 Modify the `getList` function so that the user can enter any number o brackets separated by commas (e.g. [7, 4, 2, 7, 14, 12]) to give their en number individually. The user should still have the option to enter num again ending with a blank.

Program updated ☐

B 7 The `bubbleSort` function checks every element of the list for each pa even the elements that it knows have been correctly sorted in the prev

Modify the `bubbleSort` function so that, after each pass, the numbe reduced so that elements that will not need to be swapped again are n

Program updated ☐

B 8 The `mergeSort` function is currently incomplete.

Complete the `mergeSort` function so that it performs a full merge so sorted list. The `mergeSort` function should be tested in the main pro `bubbleSort` function.

Program updated ☐

B 9 Modify the program to compare the time efficiency of the `bubbleSo` The `bubbleSort` and `mergeSort` functions should be modified to in counts the number of swaps that are made, and returns `swaps` once t

A `test` function should be added that takes an integer value, `n`, for th number of swaps made by `bubbleSort` in comparison to `mergeSor` randomly generated integers (between 1 and 100) of length `n`. The ma modified to call `test` three times, using 10, 100 and 1,000 as the valu

Program updated ☐

Exercise 3 – Towers of Hanoi

Towers of Hanoi is a game in which there are three towers and a number of different-sized discs. At the start of the game, all of the discs are placed in the same tower in size order, with the largest disc at the bottom and the smallest disc at the top. The aim of the game is to move all of the discs to the right-hand tower while following three rules:

1. Only one disc can be moved at a time.
2. A disc cannot be moved if there are any discs on top of it.
3. Discs can only be moved on to empty towers or on top of larger discs.

A simple program that sets up a game of *Towers of Hanoi* is shown below (and is p
the code and understand what is happening in the program, before attempti

```
1  class Tower():
2      def __init__(self, number, startingDiscs):
3          self.__towerNumber = number
4          self.__discs = []
5          for disc in startingDiscs:
6              self.__discs.append(disc)
7
8      def checkTower(self):
9          return self.__discs
10
11     def removeDisc(self):
12         return self.__discs.pop(-1)
13
14     def addDisc(self, disc):
15         self.__discs.append(disc)
16
17 class Game():
18     def __init__(self, noOfDiscs):
19         discs = [disc for disc in range(noOfDiscs,0,-1)]
20         self.towerOne = Tower(discs)
21         self.towerTwo = Tower(2)
22         self.towerThree = Tower(3)
23
24     def move(self, startTower, endTower):
25         disc = startTower.removeDisc()
26         endTowerDiscs = endTower.checkTower()
27         if not endTowerDiscs == []:
28             endTopDisc = endTowerDiscs[len(endTowerDiscs) - 1
29         if (not disc == None) and disc < endTopDisc:
30             endTower.addDisc(disc)
31             print("Disc moved!")
32             print()
33
34     def getMove(self):
35         print("Which tower would you like to remove a disc fr
36         startTower = input()
37         print()
38         print("Which tower would you like to move this disc t
39         endTower = input()
40         print()
41         if startTower == "1" or startTower.lower == "one":
42             startTower = self.towerOne
43         elif startTower == "2" or startTower.lower == "two":
44             startTower = self.towerTwo
45         elif startTower == "3" or startTower.lower == "three"
46             startTower = self.towerThree
47         if endTower == "1" or endTower.lower == "one":
48             endTower = self.towerOne
49         elif endTower == "2" or endTower.lower == "two":
50             endTower = self.towerTwo
51         elif endTower == "3" or endTower.lower == "three":
52             endTower = self.towerThree
53         self.move(startTower, endTower)
54
55 game = Game(5)
56 while True:
57     game.getMove()
```

SECTION A

A 1 Give a line number from the program that contains a constructor.

.....

A 2 Give a line number from the program that contains a local variable.

.....

A 3 A ca[illegible]ly be removed from or added to the end of a tower's list [illegible] the data structure which represents this behaviour and describe of that data structure.

.....

.....

.....

.....

.....

A 4 The program encounters an error when the Game class tries to instanti[illegible]
Explain the cause of this error.

.....

.....

A 5 [illegible]program does not accept "ONE", "TWO" or "THREE" as valid input
Explain the cause of this error.

.....

.....

A 6 The program will crash if the player tries to take and move a disc from
Explain the cause of this error.

.....

.....

A 7 The program will crash [illegible] player tries to take and move a disc to a[illegible]
Explain the [illegible] error.

.....

.....

A 8 Explain the purpose of the code `return self.__discs.pop(-1)`

.....

.....

.....

.....

A 9 The program uses multiple classes for encapsulation.
Explain the meaning of encapsulation.

.....

.....

.....

A 10 Explain why encapsulation is useful.

.....

.....

SECTION B

B 1 Modify the program so that it does not encounter an error when the G
`towerTwo` and `towerThree`.

Program updated ☐

B 2 Modify the program so that it accepts "O" "TWO" and "THREE" as
tower.

Program updated ☐

B 3 Modify the program so that it does not crash if the player tries to move

Program updated ☐

B 4 Modify the program so that it does not crash if the player tries to mo

Program updated ☐

B 5 Modify the program to add an `__str__` method in the `Tower` class
states the number of the tower and the discs that the tower contains.
in the `Game` class that prints out each tower in the game. The main pr
that it calls the `display` procedure before getting each move from t

Program updated ☐

B 6 Modify the `move` procedure so that when the player tries to make an i
to the tower from which it was taken (if a disc is taken) and a messa
that they have entered an invalid mo

Program updated ☐

B 7 Modify the `validMove` procedure so that the `move` procedure is only ca
to the numbers, or otherwise displays a message to say that the

Program updated ☐

B 8 Modify the program to add a `checkWon` function in the `Game` class th
successfully completed the game, or otherwise returns *False*. The mai
so that it uses this function to end the game once it has been won, ar
the game and prints a message to tell the player that they have com
program should also have the `while True:` loop changed to use a
based on the return value as `while True` loops should be avoided w

Program updated ☐

B 9 The minimum number of moves needed to complete the game is $2^n - 1$
So a game with three discs can be completed in seven moves, a game wit
moves, etc.

Modify the program to add an `autoSolve` procedure that automatic
complete the game in the minimum number of moves. The main proc
`autoSolve` instead of `getMove`, and, once the game is completed, t
should be displayed, along with a message that states whether or not
age.

Program updated ☐

Exercise 4 – Sorting Queues

This is a simple program that contains subroutines to create an implementation of a queue – a data structure where the first element to be stored is the first element to be accessed.

The queue has been implemented as a linked list, i.e. a list data is made up of individual elements connected by pointers. The program can add values to a queue, and display the elements in the queue in the order in which they were added to it.

Study the code (shown below and is provided electronically) and try to understand the program before attempting the questions that follow.

```
1  class Node():
2      def __init__(self, value, index):
3          self.value = value
4          self.index = index
5          self.nextNode = None
6
7  class Queue():
8      def __init__(self):
9          self.__startNode = None
10
11     def addValue(self, value):
12         if self.startNode == None:
13             newNode = Node(value, 0)
14         else:
15             currentNode = self.__startNode
16             while currentNode.nextNode == None:
17                 currentNode = currentNode.nextNode
18             newNode = Node(value, currentNode.index + 1)
19             currentNode.nextNode = newNode
20
21     def displayQueue(self):
22         currentNode = self.__startNode
23         while currentNode != None:
24             print("Node " + str(currentNode.index + 1) +
25             print("Value: " + str(currentNode.value))
26             print("Index: " + str(currentNode.index))
27             print()
28             currentNode = currentNode.nextNode
29
30 queue = Queue()
31 queue.addValue(1)
32 queue.addValue(4)
33 queue.addValue(2)
34 queue.addVal
35 que
36 que
37 input()
```

SECTION A

A 1 Give a line number from the program that contains a class declaration

.....

A 2 Give a line number from the program that contains string concatenation

.....

A 3 A queue is one type of data structure; a stack is another type.

Explain the difference between a queue and a stack.

.....

.....

.....

.....

A 4 The program encounters an error when the `addValue` procedure tries to access the `startNode` attribute.

Explain the cause of this error.

.....

.....

A 5 The `display` procedure fails to run when it is called.

Explain the cause of this error.

.....

.....

A 6 The `addValue` procedure fails to add new values to the queue.

Explain the cause of this error.

.....

.....

A 7 The queue is implemented as a linked list

Explain the advantage of using a linked list instead of a fixed-length array.

.....

.....

A 8 Describe how to remove an element from a linked list.

.....

.....

A 9 Queues can be implemented in different ways: for example, as a circul
Explain what a circular queue is.

.....

.....

.....

.....

.....

.....

A 10 Explain what type of queue is used in the program, and how you can c

.....

.....

.....

.....

Section B

B 1 Modify the program so that it does not encounter an error when the a
check the queue's `startNode` attribute.

Program updated ☐

B 2 Modify the program so that the `displayQueue` procedure runs corre

Program updated ☐

B 3 Modify the program so that the `addValue` procedure adds new value

Program updated ☐

B 4 Modify the `Node` class to add a public `previousNode` attribute that
before that node object in the queue. The `Node` constructor should re
set the value of `previousNode`. The `addValue` procedure should be
value of the `previousNode` attribute for each new node that is adde

Program updated ☐

B 5 Modify the `addValue` function so that when a node is added to the q
numerical order in the queue (so if the queue is currently [1, 4] and th
become [1, 2, 4]). The `nextNode` and `previousNode` attributes of ea
updated appropriately. No modifications should be made to any othe

Program updated ☐

B 6 Modify the program to add a `removeValue` function that is given an
removes the first node in the list that has the same value as the given
list, a message should be displayed to say that no nodes have been re
previous attributes of each node in the queue should be upda
program. procedure should be modified to remove the value '2' from t

Program updated ☐

Exercise 5 – Draughts

Draughts is a two-player game in which each player has a set of either black tokens or red tokens placed on an 8 x 8 grid. Players take it in turns to move one of their pieces diagonally forwards.

If there is an opposing token in a grid square diagonally forwards from the player's piece, they can jump over that token if the position behind that token is empty – and remove that token from the game. The aim of the game is to remove all of the opposing player's tokens.

A simple program that starts a game of draughts is shown below (and is provided and try to understand what is happening in the program, before attempting the qu

```
1  class Board():
2      def __init__(self):
3          self.__board = [[None]*8 for i in range(8)]
4          self.setUp()
5
6      def display(self):
7          firstLine = "   "
8
9          for c in range(8):
10             firstLine += ("| " + c + "  ")
11
12         firstLine += "|"
13         print(firstLine)
14         print("-"*((5*8)+4))
15         for r in range(8):
16             print(" " + str(r) + " ", end='')
17             for x in self.__board[r]:
18                 if x == None:
19                     y = "  "
20                 else:
21                     if x.king:
22                         y = x.getColour() + "(K)"
23                     else:
24                         y = " " + x.getColour() + "  "
25                 print("|" + y, end="")
26             print("|")
27             print("-"*((5*8)+4))
28         print()
29
30     def setUp(self):
31         for c in range(8):
32             for r in range(8):
33                 colour = ""
34                 if r == 0 or r == 1 or r == 2:
35                     colour = "R"
36                 elif r == 5 or r == 6 or r == 7:
37                     colour = "B"
38                 if r % 2 == 0 and c % 2 == 0 or r % 2 ==
39                     if not colour == "":
40                         self.__board[r][c] = Piece(colour)
41
42 class Piece():
43     def __init__(self, colour):
44         self.__colour = colour
45         self.king = False
46
47     def getColour(self):
48         return self.__colour
49
50 board = Board()
51 board.display()
```

SECTION A

A 1 Give a line number from the program that contains a private attribute

.....

A 2 Give a line number from the program that contains a public attribute.

.....

A 3 Explain why an attribute may be made public instead of private.

.....

.....

A 4 The keyword `self` is used throughout the program.
State what a class is referring to when using this keyword.

.....

.....

A 5 The program does not run and produces a syntax error.
Explain the cause of this error.

.....

.....

A 6 Explain why are placed on the board when the `setUp` procedure is run.
Explain the cause of this error.

.....

.....

A 7 The value *8* is hard-coded into the `Board` class to represent the size o
Explain why this is considered bad practice and what should be used i

.....

.....

.....

.....

A 8 A new `King` class could be created that inherits the `Piece` class.
Explain what inheritance is and why it is useful.

A 9 Explain the difference between functions, procedures and methods.

A 10 A `Board` object is created on line 50.
Explain the difference between an object and a class.

Section B

B 1 Modify the program to remove the syntax error.

Program updated ☐

B 2 Modify the program so that tokens are placed [illegible] the board when the

Program updated ☐

B 3 Modify the prog[illegible] to add a `size` attribute to the `Board` class. The [illegible] set to 8 [illegible] constructor, and the `size` attribute should be used in p[illegible] gh[illegible]t the `Board` class.

[illegible]ram updated ☐

B 4 Modify the program to add a `pieceAt` function in the `Board` class th[illegible] (a row and a column from the board) as input and returns the piece at

Program updated ☐

B 5 Modify the program to add a `validMove` function in the `Board` class[illegible] integers (a start position and an end position on the board) and a play[illegible] *True* if the player of the given colour can move a piece from the given position. The rules of movement are as follows: a player can only mov[illegible] can move in a straight diagonal line either one space (if that space is e[illegible] one space diagonally on from the start position contains an opposing[illegible] empty); non-king red pieces can only move do[illegible] and non-king black pieces can move either up or down; to[illegible] s[illegible] not move to a position

Program updated ☐

B 6 Modify t[illegible]ram to add a `getMove` function that asks the user for [illegible]or[illegible]nd checks whether a valid move has been given. If the mov[illegible]d be displayed to say that the move was successful, and two inte[illegible] positions) should be returned. Otherwise, a message should be displa[illegible] valid, and the user should be asked for new input. This function shoul[illegible] and invalid user input.

Program updated ☐

B 7 Modify the program to add a `movePiece` function that takes two list[illegible] and an end position on the board) and moves the piece at the start po[illegible] function should remove from the board any pieces that are taken, and reaches the opposing end of the board. A message should be displaye[illegible] upgraded. The function should return *True* if a piece is taken, or *False*

Program updated ☐

B 8 Modify the program to ad[illegible]kWon function that returns the pla[illegible] (if the game has b[illegible] wo[illegible]), or returns an empty string if neither play[illegible] procedure [illegible]d be modified to run in a loop that takes turns gettin[illegible] f t[illegible]layers has won, displaying the start of the board before ea[illegible] they should be given another turn. At the end of the game, a m[illegible] say which player has won.

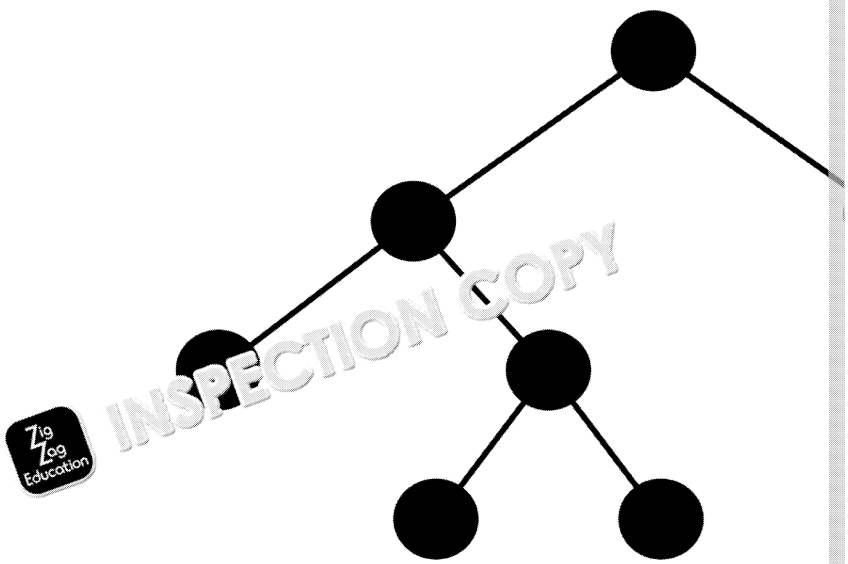
Program updated ☐

Exercise 6 – Tree Traversal

This is a simple program that creates a binary tree with set values.

Study the code (shown provided below and is provided electronically) and try to u
the program, before attempting the questions that follow.

```
1   class Node():
2       def __init__(self, value, left = None):
3           self.value = value
4           self.left = left
5
6   class Tree():
7       def __init__(self):
8           self.__rootNode = self.createBalancedTree()
9
10      def createBalancedTree(self):
11          node1 = Node(1)
12          node2 = Node(2, node1)
13          node4 = Node(4)
14          node3 = Node(3, node2, node4)
15          node6 = Node(6)
16          node8 = Node(8)
17          node7 = Node(7, node6, node8)
18          node5 = Node(5, 3, node7)
19          return node5
20
21  tree = Tree()
22  print(tree.rootNode.value)
23  print(tree.rootNode.right.value)
24  print(tree.rootNode.left.value)
25  input()
```



Section A

A 1 Give a line number from the program that contains a procedure.

.....

A 2 Give a line number from the program that contains instantiation.

.....

A 3 Draw the tree that is created by the program.

A 4 The tree created by the program is a binary tree.
Explain the difference between a binary tree and a multi-branch tree.

.....

.....

.....

A 5 The program encounters an error when trying to display the value of t
Explain the cause of this error.

.....

.....

A 6 The program encounters an error when trying to display the value of t
Explain the cause of this error.

.....

.....

A 7 The program encounters an error when trying to display the value of t
Explain the cause of this error.

.....

.....

A 8 Explain the purpose of the code `left = None` on line 2 of the progr

.....

.....

.....

A 9 Write the tree values as they woud be returned in a depth-first (post-o

.....

.....

A 10 Write the tree values as they would be returned in a breadth-first tree

.....

.....

Section B

B 1 Modify the program so that it does not encounter an error when it tri
root node.

Program updated ☐

B 2 Modify the program so that it does not encount[illegible] an error when it tri
node's right child node.

Program updated ☐

B 3 Modify the pro[illegible] o th[illegible] it does not encounter an error when it tri
node's l[illegible] node.

[illegible]m updated ☐

B 4 Modify the program to add a `depthFirstSearch` function that tak
performs a depth-first (postorder) tree traversal from that root node.
modified to call the `depthFirstSearch` function using the `rootNo`
object. Each value should be displayed in the order in which it is chec

Program updated ☐

B 5 Modify the program to add a `breadthFirstSearch` function that t
performs a breadth-first tree traversal from that root node. The main
to call the `breadthFirstSearch` function using the `rootNode` val
object. Each value should be displayed in the order in which it is chec

Program updated ☐

B 6 Modify the `Tree` constructor and `createBalan[illegible]edTree` function so
takes a sorted list of numbers as input an[illegible]te a balanced binary s
`createBalancedTree` returnin[illegible] its ro[illegible]ode. The main procedure
the list `[1,2,3,4,5,6,[illegible]8]` [illegible]onstruct a `Tree` object with this list,
and breadth-first[illegible] tr[illegible]ersals on this tree.

Pr[illegible]am [illegible]

B 7 [illegible]y the program to add a `binarySearch` function in the `Tree` cl
input, and performs a binary tree search to find the given value in a ba
message to state whether or not the given value is in the tree being se
elements that were checked, and returns the node with the given valu
(or *None* if the value is not found). The main procedure should be mo
function on the tree created from the list `[1,2,3,5,6,7,8]` to search

Program updated ☐

B 8 Modify the program to add a `removeNode` procedure in the `Tree` cla
input, searches for this value using the `binarySearch` function, and
parent of the removed node should be made to point to one of the r
had any (if there are two, it should point to the left child). If the remo
the left child node should point to the right [illegible]ild node. The main pro
remove the value *2* from the tree, an[illegible] th[illegible]n perform a binary search f

Program updated ☐

B 9 Modify the [illegible]a[illegible] to add an `addNode` procedure in the `Tree` class
[illegible] a[illegible] t[illegible]en adds this to the tree in the correct place. Note that you
`alue` method in the `Node` class to traverse the tree to the corre
`Node`. The main program should be modified to add the values *9*, *10* a
perform a binary search for the values *9–15*.

Program updated ☐

Exercise 7 – Dijkstra's Shortest Path

This is a simple program that contains subroutines to create a graph from a given list of nodes and edges, and to take any given node in the graph and find the closest connected node.

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   class Graph():
2       def __init__(nodes, edges):
3           self.nodes = nodes
4           self.edges = edges
5
6       def closestNode(self, currentNode, paths):
7           startRow = ord(currentNode) - 65
8
9           for edge in self.edges:
10              if edge[0] == currentNode:
11                  endRow = ord(edge[1]) - 65
12                  if paths[endRow][1] == None or paths[end
                    paths[startRow][1]:
13                      paths[endRow][1] = edge[2] + paths[st
14                      paths[endRow][2] = currentNode
15
16          nextNode = ''
17          shortestDistance = 0
18          for edge in edges:
19              shortestDistance = shortestDistance + edge[2
20          for node in paths:
21              if node[1] != None and node[1] < shortestDis
22                  shortestDistance = node[1]
23                  nextNode = node[0]
24
25          return nextNode
26
27  nodes = ['A', 'B', 'C', 'D', 'E', 'F']
28  edges = ['A', 'B', 12], ['A', 'C', 6], ['A', 'E', 13],
            ['B', 'F', 1], ['C', 'D', 3], ['D', 'E', 2], ['
29  graph = Graph(nodes, edges)
30  paths = []
31  for node in nodes:
32      paths.append([node, None, None])
33  startNode = 'A'
34  paths[ord(startNode) - 65][1] = 0
35  print(graph.closestNode(startNode, paths))
36  input()
```

Section A

A 1 Give a line number from the program that contains iteration.

.....

A 2 Give a line number from the program th... ta... s an attribute.

.....

A 3 Th... att... ...es ... the `Graph` class are public.

... what a public attribute is, and explain why an attribute may be ...

.....

.....

.....

.....

A 4 The code contains no comments, and the purpose of some of the un... immediately clear to anyone who sees it.

Write suitable comments to describe what is happening on the follow...

Line 7: .....

.....

Lin... 9- 1... .....

.....

Lines 20–23: .....

.....

A 5 When the program tries to run, there are two syntax errors that it enc...

State the cause of both of these syntax errors.

1. .....

.....

2. .....

.....

.....

A 6 The `closestNode` function returns the start node that it is given instea
Explain the cause of this error.

.....

.....

.....

A 7 The `closestNode` function doesn't return a node when given the inp
Explain the cause of this error.

.....

.....

.....

A 8 The program defines a graph data structure. A tree is a specific type o
Explain what a graph data structure is and what the features of a tree

.....

.....

.....

.....

A 9 The `closestNode` function can be used as part of an implementation of
Explain the purpose of Dijkstra's shortest path algorithm.

.....

.....

A 10 Describe how Dijkstra's shortest path algorithm works.

.....

.....

.....

.....

.....

SECTION B

B 1

Modify the program to remove the syntax errors.

Program updated ☐

B 2

Modify the program so that the `closestNode` function returns the clos
start node that it is given.

Program updated ☐

B 3

Modify t
gi
ram updated ☐

B 4

Modify the `closestNode` function so that before the closest node is
two-dimensional list is printed on a separate line. The value of `startNo`
be set to 'A'.

Program updated ☐

B 5

Modify the `Graph` function so that the `nodes` and `edges` attributes a
`getNodes` function and a `getEdges` function that return the `nodes`
respectively. Explain the benefit of this modification.

Program updated ☐

B 6

Modify the `Graph` class to add a `pathEx`
fu
ction that takes tw
returns *True* when the given node
b
th
ist in the graph, or returns
that lists any nodes give
t
ot exist in the graph.

Modify the m
ure to call and print the result of the `pathExi`
`st`
A' and an `endNode` of 'F'.

n updated ☐

B 7

Modify the program to add a `shortestPath` function that takes a g
Dijkstra's shortest path algorithm to find and return the series of nod
between the two given nodes (if the given nodes exist in the graph).

The `closestNode` function should be modified to take a list `visite`
visited nodes, which should be used to make sure that only the short
checked. The `closestNode` function should also return the values o
`visited`.

The main procedure should be modified to call the `shortestPath` f
the start and end nodes respectively, and print the result of the `shor`

Program updated ☐

EXERCISE 8 – BOMB SEARCH

Bomb Search is a single-player game in which a number of 'bombs' are placed in random locations on a grid. The player must turn over tiles on the grid until either they turn over a tile containing a bomb (losing the game), or they turn over all tiles except those that contain bombs (winning the game)

A simple program that sets up a game of *Bomb Search* is shown below (and is provided electronically). Study the code carefully to understand what is happening in the program, before attempting the questions that follow.

```
1  class Board():
2      def __init__(self, size, bombs):
3          self.__size = size
4          self.__bombs = bombs
5          self.__board = [[None]*self.__size for i in range
6          self.setUp()
7
8      def setUp(self):
9          for r in self.__size:
10             for c in self.__size:
11                 self.__board[r][c] = Tile()
12
13     def display(self):
14         firstLine = "   "
15         for c in range(self.__size):
16             firstLine += ("| " + str(c) + " ")
17         firstLine += "|"
18         print(firstLine)
19         print("-"*((4*self.__size)+4))
20             for r in range(self.__size
21                 print(" " + str(r) + "", end='')
22                 for x in self.__board[r]:
23                     if x.isBomb:
24                         y = " B "
25                     else:
26                         y = " " + str(x.adjBombs) + " "
27                     print("|" + y, end="")
28             print("|")
29             print("-"*((4*self.__size)+4))
30         print()
31
32     def getMove(self):
33         valid = False
34         while not valid:
35             print("Which tile would you like to reveal? (
36             locStr = input()
37             print()
38             try:
39                 loc = []
40                 loc.append(int(locStr[0]))
41                 loc.append(int(locStr[2]))
42                 valid = True
43             except:
44                 print("That is not a valid move.")
45         return loc
46
47 class Tile():
48     def __init__(self, bomb):
49         self.isBomb = False
50         self.adjBombs = 0
51         self.revealed = False
52
53 board = Board(5, 5)
54 board.display()
55 input()
```

SECTION A

A 1 Give a line number from the program that contains exception handlin

.....

A 2 Give a line number from the program tha[illegible]s a method.

.....

A 3 Explain the purpose of the code `str(x.adjBombs)` on line 26 of the

.....

.....

.....

A 4 Explain the purpose of the following line of code (line 11):

```
self.__board[r][c] = Tile()
```

.....

.....

.....

A 5 There is redundant code in the program.

Explain which code in the `Tile` constructor is unnecessary.

.....

.....

A 6 The program crashes when the `setUp` procedure tries to loop through

Explain the cause of this error.

.....

.....

A 7 The `getMove` function uses a try–except statement.

Explain why and how try–except statements are used.

.....

.....

.....

.....

A 8 The `loc` variable declared on line 39 is a list.

Explain the difference between a list and an array.

.....

.....

.....

.....

A 9 Explain whether or not it would be suitable to use an array instead of a

.....

.....

.....

.....

A 10 Line 45 returns the `loc` list which contains two integer values.

Write the code that would return the two separate integers instead of

.....

.....

Section B

B 1 Modify the program to remove the redundant code on line 48.

Program updated ☐

B 2 Modify the program so that it does not crash d[illegible]ng the `setUp` proce

Program updated ☐

B 3 Modify the `set[illegible]` procedure to add a number of bombs to the board board's [illegible]ttribute. The bombs should be placed in random posi

[illegible]m updated ☐

B 4 Modify the program to add a `checkForBombs` procedure that takes integer, and increases the value of the `adjBombs` attribute of the tile one for each bomb adjacent to (in the eight squares surrounding it) t should be run in the `setUp` procedure for each tile on the board after

Program updated ☐

B 5 Modify the `display` procedure so that tiles that have not been revea board is displayed.

Program updated ☐

B 6 Modify the program to add a `reveal` fu[illegible]t gets a move from function), reveals the given tile, if it has not yet been revealed, and ret revealed, or *False* otherwise. If the player tries to reveal a tile that has message should be displayed to tell the player that they have already program procedure should be modified to continually display the stat for the player to reveal a tile.

[illegible]gram updated ☐

B 7 Modify the program to add a `gameWon` function that returns *True* if t except those containing bombs, and returns *False* otherwise. The mai modified so that the game ends when the player reveals all of the non the player reveals a bomb (and loses). An appropriate message shoul whether the player has won or lost.

Program updated ☐

B 8 Modify the `reveal` function so that all of the tiles around it are revea of 0. The modification should involve the creation of a `revealAll` me column of the tile around which all tiles are to be revealed. This shoul any tile revealed by part of this process that also has an `adjBombs` va

Program updated ☐

Exercise 9 – Dictionaries and Hash Tables

An organisation is currently storing a list of its members' information. The organisa
whose data they need to store, and so have decided to change the list into a data s
looking up data.

A simple program that stores a list of the organisation's m ers' information is sh
electronically). Study the code and try to understand what is happening in the pro
questions that follow.

```
1  size = 19
2
3  tab [] 
4  for i in range(size):
5      table.append([])
6
7  def addMember(number, name, postcode, memberList):
8      memberList.append([number,name,postcode])
9
10 members = [ [123,"Robin","AB4"],
11             [124,"Nguyen","HD12"],
12             [125,"Jev","L18"],
13             [126,"Will","OX5"],
14             [127,"Lily","CH3"],
15             [128,"Jonny","YO12"],
16             [129,"Clara","BS1"],
17             [130,"Callum""BA1"]
18           ]
19
20 addMember( ,"irsten","SE2",members)
21 for er in members:
22     (member[0],member[1],member[2])
23 input()
```

SECTION A

A 1 Give a line number from the program that contains a global variable.

.....

A 2 Give a line number from the program that contains a list declaration.

.....

A 3 Explain why it is good practice to create and use the `addMember` procedure for appending the new data to the `members` list on line 20.

.....

.....

.....

A 4 The `members` list could instead be implemented as a dictionary.

Describe what a dictionary is.

.....

.....

.....

A 5 Explain why you might choose to use a dictionary to store data.

.....

.....

A 6 The program does not run and produces a runtime error.

Explain the cause of this error.

.....

.....

A 7 The `table` variable has been set up so that it can be used as a hash table for storing member information.

Explain how data is stored in a hash table.

.....

.....

A 8 The unique number given to each member can be used as the input fo

a) Explain what a hash function is.

.....

.....

.....

b) How would the hash function work if the member IDs contained le

.....

.....

.....

c) Although primary keys for data are unique, it is often the case tha generate collisions. Explain why this is the case and why it is neces

.....

.....

.....

.....

A 9 Compare and contrast the use of a dictionary and a hash table for stori

.....

.....

.....

.....

.....

.....

A 10

The names and postcodes of all members will be stored in the hash ta
function: `(memberNumber * memberNumber) % size`

Draw the contents of this hash table once the data from the `members`
the new entry added on line 20). One entry has been given.

Hash Table Location	First Entry	Se
table[0]		
table[1]		
table[2]		
table[3]		
table[4]		
table[5]	[123,"Robin","AB4"]	
table[6]		
table[7]		
table[8]		
table[9]		
table[10]		
table[11]		
table[12]		
table[13]		
table[14]		
table[15]		
table[16]		
table[17]		
table[18]		

SECTION B

B 1 Modify the program to remove the runtime error.

Program updated ☐

B 2 Modify the program to add a `createDictionary` function that take
argument and returns a dictionary to [illegible] as `membersDictiona`
dictionary should have the m[illegible]ship number as the key for the dic
another dictionary wi[illegible] name" and "postcode" for the remai
program to [illegible]ction and store the result in the variable `membe`
[illegible]ary to check that it is correct.

[illegible] updated ☐

B 3 Create a `displayDictionary` procedure that takes two parameters,
`membersDictionary` and the second being the `sortField`. Displa
format ordered in ascending order by the `sortField` which could ta
or "postcode". Modify the main program so that, in addition to printin
procedure three times, once with each possible value of `sortField`.

Program updated ☐

B 4 Modify the program to add a `getHash` function that takes a `memberN`
a hash value calculated by the formula: `memberNumber * memberNu`
return value of `getHash` for all of the membership numbers.

Program updated ☐

B 5 Change the data type of the [illegible]ership number to string by adding
name to the start of [illegible] membership number. Update the `members`
to reflect this. Update the `getHash` function to a new algorithm that i
`[illegible]mber * asciiValueOfMemberLetter) % size`. Prin
`[illegible]sh` for all of the membership numbers.

Program updated ☐

B 6 Modify the program to add a `createHash` function that has `member`
parameters and returns a hash table containing the members, using t
position in the list and storing each member inside a dictionary withi
have multiple entries if multiple members are hashed to the same loc

Program updated ☐

B 7 Modify the program to add an `addMemberHash` procedure that will ta
number, name and postcode and add a member to the hash table. Add
to the hash table using your new procedure, a[illegible]en print out the has

Program updated ☐

B 8 Modify the p[illegible] add a `removeMemberHash` procedure that will
[illegible]be[illegible]mber, and delete the entry from the hash table for that
[illegible]" from the hash table using your new procedure, and then print ou

Program updated ☐

Exercise 10 – Reverse Polish

Mathematical expressions are usually written in 'infix' notation, meaning that oper
use the values on either side of the operation. For example, the expression '2 + 2' i
evaluated by adding the value on the left of the '+' operator to the value on its righ

An alternative way of writing mathematical expressions is using 'postfix' notation, also
commonly known as Reverse Polish Notation (RPN). *In case you were wondering, 'Poli
reference to the nationality of Jan Łukasiewicz (pictured), who invented the notation in*

In RPN, the operator comes after the values it operates on, so the infix expression '
would instead be written '2 2 +'. RPN expressions are evaluated from left to righ
the expression '3 4 2 - *' would be evaluated as follows:

3 4 2 - * *The (-) operator is reached and operates on the two operands that come*
3 2 * 4 2 - *is simplified to its result, 2. The next operator (*) operates on the t*
6 3 2 * *is then simplified to its result, 6.*

Shown below (and provided electronically) is a program that gets an infix expressio
expression into separate elements, and converts the order of the elements from in
code and try to understand what is happening in the program, before attempting t

```
operators = ['+', '-', '*', '/']

def getElements(expression):
    elements = []
    element = ""
    expression = expression.strip()

    for i in expression:
        if i != " ":
            element = element + i
        else:
            elements.append(element)
            element = ""
    valid = checkExpression(elements)
    if valid:
        return elements
    else:
        print("Invalid expression given!")
        return[]

def checkExpression(elements):
    lastElement = None
    for element in elements:
        if isInt(element):
            element = int(element)
        else:
            isOperator = False
            for operator in operators:
                if element == operator:
                    isOperator = True
            if isOperator == False:
                return False
        if isInt(lastElement):
            if isInt(element):
                return False
        else:
```

```
37              if not isInt(element):
38                  return False
39          lastElement = element
40      return True
41
42  def isInt(value):
43      try:
44          int(value)
45          return True
46      except:
47          return False
48
49  def infixToRPN(elements):
50      stack = []
51      opStack = []
52      for el      i elements:
53          if  (element):
54              stack.append(element)
55          else:
56              if opStack != []:
57                  lastOp = opStack[-1]
58              if opStack == [] or element == '(' or ((lastOp ==
                lastOp == '-') and (element == '/' or element == '*
59                  opStack.append(element)
60              elif element == ')':
61                  operator = None
62                  while operator != '(' and opStack != []:
63                      operator = opStack.pop()
64                      if operator != '(':
65                          stack.append(operator)
66              else:
67                  if lastOp != '(':
68                      stack.append(
69                      opStack[
70                  else:
71                      stack.append(element)
72      for i      ge(len(opStack)):
73          stac  append(opStack.pop())
74      return stack
75
76  elements = getElements(input("Enter an expression: "))
77  elements = infixToRPN(elements)
78  print(elements)
79  input()
```

Section A

A 1 Give a line number from the program that contains a comparison oper

.....

A 2 Give a line number from the program that contains a substring operat

.....

A 3 Explain how the `isPrint` function determines whether or not the given

.....

.....

.....

A 4 Write the RPN form of the following infix expression: (3 + 2) * (4 - 1) /

.....

.....

.....

A 5 Write the infix form of the following RPN expression: 4 5 + 3 2 1 / - *

.....

.....

A 6 The program currently does not always add the final operand given to

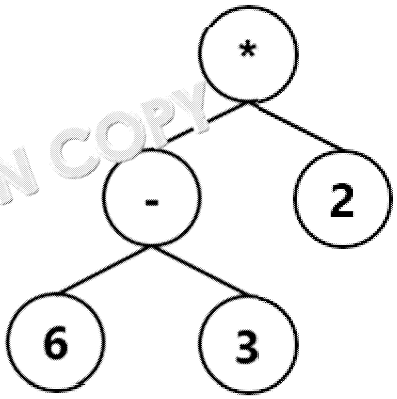
Explain the cause of this error.

.....

.....

A 7 Mathematical expressions can be represented as a binary tree, where a
produce the RPN expression, and an inorder tree traversal will produce

Write the RPN expression produced by the following binary tree:



.....

.....

A 8 Write the infix expression produced by the binary tree in A7.

.....

.....

A 9 Draw the binary tree that is created by the following infix expression: 8

A 10 Draw the binary tree that is created by the following RPN expression: 4

Section B

B 1 Modify the `getElements` function so that it always adds the final op
elements that it returns.

Program updated ☐

B 2 Modify the program so that it continu... as the user for input until

Program updated ☐

B 3 Modify t... so that the `checkExpression` function accepts
...ets as operators, providing a ')' operator is only placed after an o
...tors does not exceed the number of '(' operators at any point wh
left to right, and the final expression contains an equal number of '(' o

Program updated ☐

B 4 Modify the program to add a `Node` class with the attributes `value`, `le`
constructor that sets `value` to be equal to a given value, and sets the
`None`. Also add an `ExpressionTree` class with the attribute `root`, w
of elements for an RPN expression and uses this list to set `root` to be
`createTree` function. The `ExpressionTree` class should contain a
returns the root node of the binary tree created from a given list of el
and a `showTree` procedure that performs a breadth-first search of th
tree on a separate line. The main program procedure should be modif
result of the call to `infixToRPN` and then call `showTree` for that tree

Program updated ☐

B 5 Modify the `Expres...` class to add an `RPN` function that trave
traversal) an... the RPN expression that the tree represents as a
...ed... should be modified to display the result of calling `RPN` for t

...n updated ☐

B 6 Modify the `ExpressionTree` class to add an `infix` function that tra
traversal) and returns the infix expression that the tree represents as a
placed around each operand in the expression, e.g. the expression "(4
"(((4) + (2)) * (3))". The main program procedure should be modifie
`infix` for the created tree.

Program updated ☐

B 7 Modify the program to add a `removeExtraBrackets` function whi
removes any brackets that only surround a single integer, and the set
entire expression, e.g. "(((4) + (2)) * (3))" wo... become "(4 + 2) * 3
The main program procedure sho... ...ined to display the result
`removeExtraBrackets` f... on the result of the call to `infix`

Program updated ☐

B 8 **...enge Question (no solution provided).** Can you change the wa
...ered so that they are flexible with spaces (i.e. no spaces required a

Answers

When studying the suggested answers for Section B tasks, it is important to remembe of achieving the same outcome, and credit should be given for alternative solutions.

EXERCISE 1 – SEARCHING ALGORITHMS

A 1

1 mark for giving a suitable example:

Line 11 / Line 13 / Line 24

A 2

1 mark for giving a suitable example:

Line 1

A 3

1 mark for explaining that binary search is more efficient / faster than

Binary search is usually more time efficient (takes less time to run) th

A 4

1 mark for explaining that binary search can only be performed on so

The list might be unsorted – a binary search requires the list to be so

A 5

1 mark for explaining the cause of the error:

There are no quotes surrounding *Value not found* on line 7 and line the individual words as variables instead of the sentence as a string.

A 6

1 mark for explaining the cause of the err

The `linearSearch` function returns the value that has been found location of that valu

A 7

Up to 2 marks for explaining the cause of the error:

`binarySearch` function discards the values that come before/a checked (if the value is higher/lower than the search value), but not th means that when there are only two elements at the end of the list le checked, no elements are discarded from the list, and the function re

A 8

2 marks (1 mark for explaining that time complexity describes numbe taken to run, 1 mark for explaining how time complexity relates to var

The time complexity of an algorithm is a description of the number o takes to complete in relation to the size of the input given to the alg

A 9

2 marks (1 mark for stating the time complexity of linear search; 1 ma of binary search):

Linear search has a time complexity of $O(n)$, binary search has a time

A 10

Up to 2 marks for explaining why recursion may not be suitable. For e

Recursion may not be suitable for searching large lists because each new stack frame which includes a copy of the list (or part of the list), registers and local variables to be created, which could lead to the memory to store the lists. Additionally, many language/OS combinat limits for this reason. All of this will generally make recursive solution iterative solutions.

B 1

1 mark available for modifying the code as shown (or equivalent code)

```
    return  "Value not found"
...
    return "Value not found"
```

B 2

1 mark available for modifying the code (as shown below or equivalent)

```
def linearSearch(searchList, searchVal):
    for index, i in enumerate(searchList):
        if i == searchVal:
            return index
    return "Value not found"
```

B 3

1 mark available for modifying the code (as shown below or equivalent)

```
def binarySearch(searchList, searchVal):
    start = 0
    end = len(searchList) - 1
    while start <= end:
        mid = (start + end) // 2
        if searchList [mid] == searchVal:
            return mid
        elif searchList [mid] < searchVal:
            start = mid + 1
        elif searchList [mid] > searchVal:
            end = mid - 1
    return "Value not found"
```

B 4

5 marks available for modifying the code (as shown below or equivalent)

Marks could be awarded for:

- creating a `recursiveBinarySearch` function that takes a input
- returning the correct index (in the original list) when the el
- returning "Value not found" if the element is not in the list
- recursively calling `recursiveBinarySearch` if another sta
- modifying the main program procedure to display the resul

```
def recursiveBinarySearch(searchList, searchVal,
    mid = len(searchList) // 2
    if searchList [mid] == searchVal:
        return mid + startIndex
    elif len(searchList) == 1:
        return "Value not found"
    elif searchList [mid] < searchVal:
        return recursiveBinarySearch(searchLis
            startIndex + mid + 1)
    elif searchList [mid] > searchVal:
        return recursiveBinarySearch(searchList
            startIndex)
...
searchList = [1,2,3,4,5,6,7,8,9,10]
print(linearSearch(searchList, x))
print(binarySearch(searchList, x))
print(recursiveBinarySearch(searchList, x, 0))
input()
```

B 5

4 marks available for modifying the code (as shown below or equivale

Marks could be awarded for:

- creating a `getVal` function that repeats until a valid input i
- handling (but not accepting) invalid input
- using appropriate messages
- returning the input value as an integer
- modifying the main program proc... to use `getVal` to s

```
def getVal():
    number = 1
    while number <= 0:
        number = input("Please enter a positi
        try:
            number = int(number)
            assert number > 0
        except:
            print("Error -",number, "is not a
            number = -1
    return number
...
searchList = [1,2,3,4,5,6,7,8,9,10]
x = getVal()
print(linearSearch(searchList, x))
```

B 6

2 marks available for modifying the code (as shown below or equivale

Marks could be awarded for:

- creating a `generateList` function that correctly generate 1 to a given length
- modifying the main pr... edure to use `generateL`

```
def gener...i...length):
    ... [x for x in range(1,length+1)]
...
searchList = generateList(getVal())
x = getVal()
print(linearSearch(searchList, x))
```

B 7

5 marks available for modifying the code (as shown below or equivale

Marks could be awarded for:

- correctly counting and returning the number of steps made
- correctly counting and returning the number of steps made
- creating a `test` function that returns the average number o algorithm on a list of a given length
- modifying the main program to use `test` to perform 1,000 1,000, 10,000 and 100,000
- modifying the main program to display how many more op on average in comparison to `binarySearch` for each of th

```
from random import randint

def linearSearch(searchList, searchVal):
    count = 0
    for index, i in enumerate(searchList):
        count = count + 1
        if i == searchVal:
            return count
        return count
...
def binarySearch(searchList, searchVal):
    start = 0
    count = 0
    end = len(searchList) - 1
    while start <= end:
        count = count + 1
        mid = (start + end) // 2
        if searchList [mid] == searchVal:
            return count
        elif searchList [mid] < searchVal:
            start = mid + 1
        elif searchList [mid] > searchVal:
            end = mid - 1
        return count
...
def test(searchList, tests):
    linearSteps = 0
    binarySteps = 0
    for i in range (tests):
        x = randint(1,len(searchList))
        noOfSteps = linearSearch(searchList, x
        linearSteps = linearSteps + noOfSteps
        noOfSteps = binarySearch(searchList, x
        binarySteps = binarySteps + noOfSteps
    return linearSteps/tests, binarySteps/tes

for i in range(5):
    n = 10 ** (i + 1)
    linear, binary = test(generateList(n), 10
    print("Lists of size " + str(n) + ":"
    print("Linear search took " + str(lin
    longer than binary search")
    print()
input()
```

Preview of Answers Ends Here