Computer Science AS & A Level | OCR | H046 & H446

Python Exercises

for AS & A Level OCR Computer Science

Download support files from zzed.uk/productsupport

zigzageducation.co.uk

POD 9944

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 OCR 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.

NB. Exercise 10 introduces the concept of postfix (Reverse Polish) notation which, while not required by the OCR specification, is included in this resource because it gives students the chance to develop their understanding of creating, manipulating and traversing trees, and to practise the other important programming skills that are reinforced throughout the resource. The skeleton code provided for this exercise is more expansive than the skeleton code for other exercises, and so this may be considered an additional activity for more talented students who relish the extra level of challenge.

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**

OCR 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
2.2.1 – Programming constructs	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2.2.1 – Recursion	✓	✓				✓				✓
2.2.1 – Global and local variables	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2.2.1 – Modularity	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2.2.1 – Debugging programs	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2.2.1 – Object-oriented techniques			✓	✓	✓	✓	✓	✓		✓
2.2.2 – Divide and conquer	✓	✓				✓				✓
2.3.1 – Algorithm efficiency	✓	✓							✓	
2.3.1 – Measuring efficiency	✓	✓								
2.3.1 – Comparing algorithm complexity	✓	✓							✓	
2.3.1 – Data structures			✓	✓		✓	✓		✓	✓
2.3.1 – Standard algorithms	✓	✓					✓		✓	

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 below and is provided electr to understand what is happening in the program, before attempting the questions

```
x = 4

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

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
        elif searchList [mid] > searchVal:
            end = mid
    return Value not found

searchList = [1,2,3,4,5,6,7,8,9,10]
print(linearSearch(searchList, x))
print(binarySearch(searchList, x))
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 preferabl

....

....

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

....

....

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 eve 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 consecutive 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 i

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 `l` `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 lo took in comparison to `binarySearch` on average for lists of the give

Program updated □

Exercise 2 – Sorting Algorithms

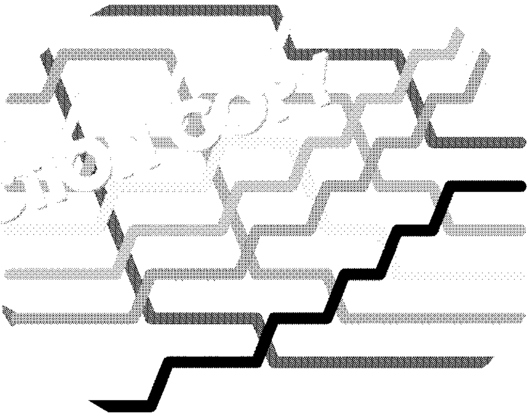
This is a simple program that provides two functions that, when given an integer li ascending order. The first function uses a bubble sort algorithm to sort the given li uses a merge sort algorithm.

A program designed to test these functions is shown below and is provided electr to understand what is happening in the program, before attempting the questions

```
def bubbleSort(sortList):
    sorted = False
    length = len(sortList)
    while !sorted:
        for i in range(length - 2):
            if sortList[i] > sortList[i+1]:
                sortList[i] = sortList[i+1]
                sortList[i+1] = sortList[i]
                sorted = False
    return sortList

def mergeSort(sortList):
    mid = len(sortList) // 2
    leftHalf = sortList[:mid]
    rightHalf = sortList[mid:]
    if len(sortList) > 1:
        mergeSort(leftHalf)
        mergeSort(rightHalf)

numList = []
for i in range(5):
    print("Add an integer number to the list: ")
    numList.append(int(input()))
print("Bubble sort given:")
print(numList)
print("Bubble sort returns")
print(bubbleSort(numList))
input()
```



SECTION A

A 1 Give a line number from the program that contains a call by reference.

....

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

....

A 3 Define [illegible].

....

....

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 `bubbl` 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 [illegible] the time complexity of the bubble sort and merge sort algorith

....

....

....

....

A 10 Another method of sorting a list of numbers is known as an insertion
Describe how the insertion sort algorithm works.

....

....

....

....

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 t

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 when they have entered a non-integer number and keep asking the u valid integer. The input should terminate once they enter a blank (jus

Program updated □

B 5 Currently, the code that asks the user to enter the numbers in the list 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 p 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. [[illegible], 4, [illegible], 14, 12]) to give their e number individually. The user should still have the option to enter nu again ending with a blank.

Program updated □

B 7 The `bubbleSort` function checks every element of the list for each p 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

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 [illegible] 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 m 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 placed in 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 try to understand what is happening in the program, before attempt

```
class Tower():
    def __init__(self, number, startingDiscs):
        self.__towerNumber = number
        self.__discs = []
        for disc in startingDiscs:
            self.__discs.append(disc)

    def checkTower(self):
        return self.__discs

    def removeDisc(self):
        return self.__discs.pop(-1)

    def addDisc(self, disc):
        self.__discs.append(disc)

class Game():
    def __init__(self, noOfDiscs):
        discs = [disc for disc in range(noOfDiscs,0,-1)]
        self.towerOne = Tower(1, discs)
        self.towerTwo = Tower(2)
        self.towerThree = Tower(3)

    def move(self, startTower, endTower):
        disc = startTower.removeDisc()
        endTowerDiscs = endTower.checkTower()
        if not endTowerDiscs == []:
            endTopDisc = endTowerDiscs[len(endTowerDiscs) - 1
        if (not disc == None) and disc < endTopDisc:
            endTower.addDisc(disc)
            print("Disc moved!")
            print()

    def getMove(self):
        print("Which tower would you like to remove a disc f
        startTower = input()
        print()
        print("Which tower would you like to move this disc t
        endTower = input()
        print()
        if startTower == "1" or startTower.lower == "one":
            startTower = self.towerOne
        elif startTower == "2" or startTower.lower == "two":
            startTower = self.towerTwo
        elif startTower == "3" or startTower.lower == "three"
            startTower = self.towerThree
        if endTower == "1" or endTower.lower == "one":
            endTower = self.towerOne
        elif endTower == "2" or endTower.lower == "two":
            endTower = self.towerTwo
        elif endTower == "3" or endTower.lower == "three":
            endTower = self.towerThree
        self.move(startTower, endTower)

game = Game(5)
while True:
    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 [illegible] can only 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
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 if the player tries to take and move a disc to a
Explain the cause of this 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
towerTwo and towerThree.

Program updated □

B 2 Modify the program so that it accepts "ONE", "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 p
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
to the tower from which it was taken (if a disc was taken) and a mess
that they have entered an invalid move.

Program updated □

B 7 Modify the getMove procedure so that the `move` procedure is only ca
to [illegible] 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 t
successfully completed the game, or otherwise returns *False*. The ma
so that it uses this function to end the game once it has been won, a
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 automati
complete the game in the minimum number of moves. The main pro
`autoSolve` instead of `getMove`, and, once the game is completed, t
should be displayed, along with a message that states whether or no
made.

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 linked list 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 reproduced below and is provided electronically) and try to u the program before attempting the questions that follow.

```
class Node():
    def __init__(self, value, index):
        self.value = value
        self.index = index
        self.nextNode = None

class Queue():
    def __init__(self):
        self.__startNode = None

    def addValue(self, value):
        if self.startNode == None:
            newNode = Node(value, 0)
        else:
            currentNode = self.__startNode
            while not currentNode.nextNode == None:
                currentNode = currentNode.nextNode
            newNode = Node(value, currentNode.index + 1)
            currentNode.nextNode = newNode

    def displayQueue(self):
        currentNode = self.__startNode
        while currentNode != None:
            print("Node " + str(currentNode.index + 1) +
            print("Value: " + str(currentNode.value))
            print("Index: " + str(currentNode.index))
            print()
            currentNode = currentNode.nextNode

queue = Queue()
queue.addValue(1)
queue.addValue(4)
queue.addValue(2)
queue.addValue([illegible])
queue.addValue(7)
queue.displayQueue
input()
```

Section A

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

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

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 trie `startNode` attribute.

Explain the cause of this error.

A 5 The `displayQueue` 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 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 valu

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 `previousNode` and `nextNode` attributes of each node in the queue should be upda `main` 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 sets up a game of draughts is shown below (and is provided and try to understand what is happening in the program, before attempting the qu

```
class Board():
    def __init__(self):
        self.__board = [[None]*8 for i in range(8)]
        self.setUp()

    def display(self):
        firstLine = "    "

        for c in range(8):
            firstLine += ("| " + c + "  ")

        firstLine += "|"
        print(firstLine)
        print("-"*((5*8)+4))
        for r in range(8):
            print(" " + str(r) + " ", end="")
            for x in self.__board[r]:
                if x == None:
                    y = "     "
                else:
                    if x.king:
                        y = x.getColour() + "(K)"
                    else:
                        y = " " + x.getColour() + "  "
                print("|" + y, end="")
            print("|")
            print("-"*((5*8)+4))
        print()

    def setUp(self):
        for c in range(8):
            for r in range(8):
                colour = ""
                if r == 0 or r == 1 or r == 2:
                    colour = "R"
                elif r == 5 or r == 6 or r == 7:
                    colour = "B"
                if r % 2 == 0 and c % 2 == 0 or r % 2 ==
                    if not colour == "":
                        self.__board[r][c] = Piece(colour)

class Piece():
    def __init__(self, colour):
        self.__colour = colour
        self.king = False

    def getColour(self):
        return self.__colour

board = Board()
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 No tokens 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 on the board when the

Program updated □

B 3 Modify the program to add a `size` attribute to the `Board` class. The
set to *8* in the constructor, and the `size` attribute should be used in
throughout the `Board` class.

Program updated □

B 4 Modify the program to add a `pieceAt` function in the `Board` class th
(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
integers (a start position and an end position on the board) and a pla
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
can move in a straight diagonal line either one space (if that space is e
one space diagonally on from the start position contains an opposing
empty); non-king red pieces can only move down and non-king black
pieces can move either up or down; tokens cannot move to a positio

Program updated □

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

Program updated □

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

Program updated □

B 8 Modify the program to add a `checkWon` function that returns the pla
(if the game has been won), or returns an empty string if neither play
procedure should be modified to run in a loop that takes turns getting
of the players has won, displaying the start of the board before ea
they should be given another turn. At the end of the game, a m
say which player has won.

Program updated □

Exercise 6 – Tree Traversal

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

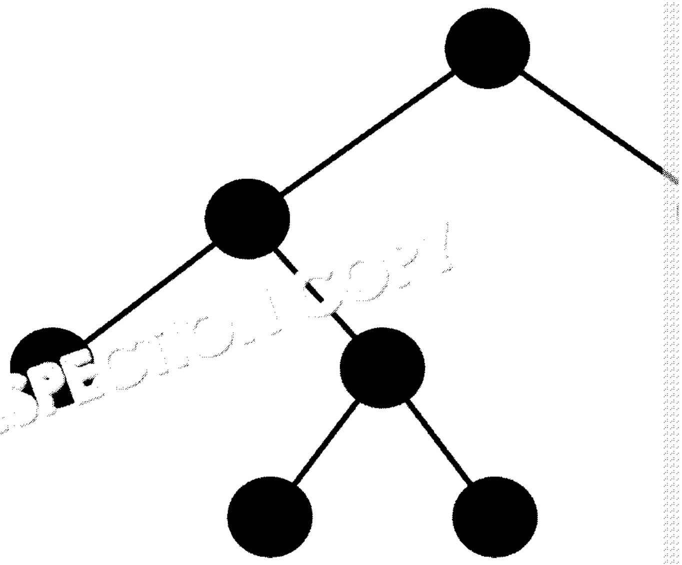
Study the code (shown provided below and is provided electronically) and try to u the program, before attempting the questions that follow.

```
class Node():
    def __init__(self, value, left = None):
        self.value = value
        self.left = left

class Tree():
    def __init__(self):
        self.__rootNode = self.createBalancedTree()

    def createBalancedTree(self):
        node1 = Node(1)
        node2 = Node(2, node1)
        node4 = Node(4)
        node3 = Node(3, node2, node4)
        node6 = Node(6)
        node8 = Node(8)
        node7 = Node(7, node6, node8)
        node5 = Node(5, 3, node7)
        return node5

tree = Tree()
print(tree.rootNode.value)
print(tree.rootNode.right.value)
print(tree.rootNode.left.value)
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 encounter an error when it tri node's right child node.

Program updated □

B 3 Modify the program so that it does not encounter an error when it tri node's left child node.

Program 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 `rootN` 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` va object. Each value should be displayed in the order in which it is chec

Program updated □

B 6 Modify the `Tree` constructor and `createBalancedTree` function s takes a sorted list of numbers as input and creates a balanced binary s `createBalancedTree` returning its root node. The main procedure the list `[1,2,3,4,5,6,7,8]` to construct a `Tree` object with this list, and breadth-first tree traversals on this tree.

Program updated □

B 7 Modify 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 b 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 child node. The main pro remove the value *2* from the tree, and then perform a binary search f

Program updated □

B 9 Modify the program to add an `addNode` procedure in the `Tree` class a [illegible] adds this to the tree in the correct place. Note that yo `lue` 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.

```
class Graph():
    def __init__(nodes, edges):
        self.nodes = nodes
        self.edges = edges

    def closestNode(self, currentNode, paths):
        startRow = ord(currentNode) - 65

        for edge in self.edges:
            if edge[0] == currentNode:
                endRow = ord(edge[1]) - 65
                if paths[endRow][1] == None or paths[end
                paths[startRow][1]:
                    paths[endRow][1] = edge[2] + paths[s
                    paths[endRow][2] = currentNode

        nextNode = ''
        shortestDistance = 0
        for edge in edges:
            shortestDistance = shortestDistance + edge[2
        for node in paths:
            if node[1] != None and node[1] < shortestDis
                shortestDistance = node[1]
                nextNode = node[0]

        return nextNode

nodes = ['A', 'B', 'C', 'D', 'E', 'F']
edges = ['A', 'B', 12], ['A', 'C', 6], ['A', 'E', 13],
        ['B', 'F', 1], ['C', 'D', 3], ['D', 'E', 2], ['
graph = Graph(nodes, edges)
paths = []
for node in nodes:
    paths.append([node, None, None])
startNode = 'A'
paths[ord(startNode) - 65][1] = 0
print(graph.closestNode(startNode, paths))
input()
```

SECTION A

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

....

A 2 Give a line number from the program that declares an attribute.

....

A 3 The attributes of the Graph class are public.

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

....

....

....

....

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: ....

....

Lines 9–1[illegible]: ....

....

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 inste

Explain the cause of this error.

....

....

....

A 7 The `closestNode` function doesn't return a node when given the in

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 the program to correct the error that means that the `closestNo`
given the input 'F'. The main procedure should be modified to make t

Program 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 `pathExists` function that takes tw
returns *True* when the given nodes both exist in the graph, or returns
that lists any nodes given that do not exist in the graph.

Modify the main procedure to call and print the result of the `pathEx`
`startNode` of 'A' and an `endNode` of 'F'.

Program 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 `visit`
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 `sho`

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 and try to understand what is happening in the program, before attempting the questions that follow.

```
class Board:
    def __init__(self, size, bombs):
        self.__size = size
        self.__bombs = bombs
        self.__board = [[None]*self.__size for i in range
        self.setUp()

    def setUp(self):
        for r in self.__size:
            for c in self.__size:
                self.__board[r][c] = Tile()

    def display(self):
        firstLine = "    "
        for c in range(self.__size):
            firstLine += ("| " + str(c) + " ")
        firstLine += "|"
        print(firstLine)
        print("-"*((4*self.__size)+4))
            for r in range(self.__size):
                print(" " + str(r) + "  ", end='')
                for x in self.__board[r]:
                    if x.isBomb:
                        y = " B "
                    else:
                        y = " " + str(x.adjBombs) + " "
                    print("|" + y, end="")
            print("|")
            print("-"*((4*self.__size)+4))
        print()

    def getMove(self):
        valid = False
        while not valid:
            print("Which tile would you like to reveal? (
            locStr = input()
            print()
            try:
                loc = []
                loc.append(int(locStr[0]))
                loc.append(int(locStr[2]))
                valid = True
            except:
                print("That is not a valid move.")
        return loc

class Tile:
    def __init__(self, bomb):
        self.isBomb = False
        self.adjBombs = 0
        self.revealed = False

board = Board(5, 5)
board.display()
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 that contains 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 [illegible]

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

....

....

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

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 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 during the `setUp` proce

Program updated □

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

Program 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) th 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` function that 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's main procedure should be modified to continually display the stat player to reveal a tile.

Program updated □

B 7 Modify the program to add a `gameWon` function that returns *True* if th except those containing bombs, and returns *False* otherwise. The ma modified so that the game ends when the player reveals all of the non the player reveals a bomb (and loses). An appropriate message shoulc 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 procedure 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 members' information is s electronically). Study the code and try to understand what is happening in the pro questions that follow.

```
size = 19

table = []
for i in range(size):
    table.append([])

def addMember(number, name, postcode, memberList):
    memberList.append([number,name,postcode])

members = [ [123,"Robin","AB4"],
            [124,"Nguyen","HD12"],
            [125,"Jev","L18"],
            [126,"Will","OX5"],
            [127,"Lily","CH3"],
            [128,"Jonny","YO12"],
            [129,"Clara","BS1"],
            [130,"Callum""BA1"]
          ]

addMember(131,"Kirsten","SE2",members)
for member in members:
    print(member[0],member[1],member[2])
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` proc manually 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 want 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 t 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 tak argument and returns a dictionary to be stored as `membersDiction` dictionary should have the membership number as the key for the dic another dictionary with the keys "name" and "postcode" for the remai program to call the function and store the result in the variable `memb` the dictionary to check that it is correct.

Program updated □

B 3 Create a `displayDictionary` procedure that takes two parameter `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 printi 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 `member` a hash value calculated by the formula: `memberNumber * memberN` return value of `getHash` for all of the membership numbers.

Program updated □

B 5 Change the data type of the membership number to string by adding name to the start of each membership number. Update the `members` to reflect this. Update the `getHash` function to a new algorithm that `(memberNumber * asciiValueOfMemberLetter) % size`. Prin `getHash` for all of the membership numbers.

Program updated □

B 6 Modify the program to add a `createHash` function that has `membe` 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 lo

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 <u>your new procedure</u> and then print out the has

Program updated □

B 8 Modify the program to add a `removeMemberHash` procedure that wil member number, and delete the entry from the hash table for that [illegible]" from the hash table using your new procedure, and then print o

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 as '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 com*
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

Understanding of Reverse Polish is <u>not</u> a requirement for your course, so you will ***no***

```
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:
            if not isInt(element):
                return False
        lastElement = element
    return True

def isInt(value):
    try:
        int(value)
        return True
    except:
        return False

def infixToRPN(elements):
    stack = []
    opStack = []
    for element in elements:
        if isInt(element):
            stack.append(element)
        else:
            if opStack != []:
                lastOp = opStack[-1]
            if opStack == [] or element == '(' or ((lastOp == '
            lastOp == '-') and (element == '/' or element == '*
                opStack.append(element)
            elif element == ')':
                operator = None
                while operator != '(' and opStack != []:
                    operator = opStack.pop()
                    if operator != '(':
                        stack.append(operator)
            else:
                if lastOp != '(':
                    stack.append(lastOp)
                    opStack[-1] = element
                else:
                    opStack.append(element)
    for i in range(len(opStack)):
        stack.append(opStack.pop())
    return stack

elements = getElements(input("Enter an expression: "))
elements = infixToRPN(elements)
print(elements)
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 `isInt` 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 / - *

....

....

....

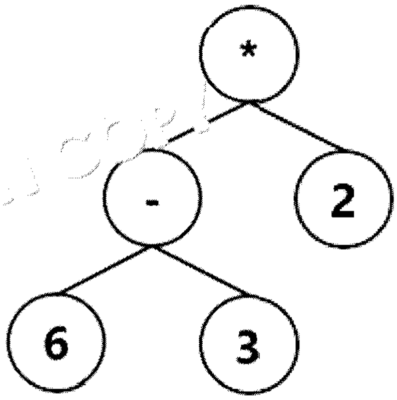
A 6 The program currently does not always add the final operand given to
Explain the cause of this error.

....

....

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

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:

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

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 continually asks the user for input until

Program updated □

B 3 Modify the program so that the checkExpression function accepts brackets as operators, providing a ')' operator is only placed after an o operators 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, l 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 modi result of the call to infixToRPN and then call showTree for that tre

Program updated □

B 5 Modify the ExpressionTree class to add an RPN function that trave traversal) and returns the RPN expression that the tree represents as a procedure should be modified to display the result of calling RPN for t

Program updated □

B 6 Modify the ExpressionTree class to add an infix function that tr 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 whic removes any brackets that only surround a single integer, and the set entire expression, e.g. "(((4) + (2)) * (3))" would become "(4 + 2) * 3

The main program procedure should be modified to display the result removeExtraBrackets function on the result of the call to infix f

Program updated □

B 8 **Challenge Question (no solution provided).** Can you change the wa entered 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 error:

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

A 7

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

The `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 va

The time complexity of an algorithm is a description of the number c 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 solutio 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 equivale

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

B 3

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

```
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 equivale

Marks could be awarded for:

- defining a recursiveBinarySearch function that takes a input
- returning the correct index (in the original list) when the ele
- 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 procedure 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 program procedure to use `generateL`

```
def generateList(length):
    return [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 equival

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

This is a limited inspection copy. Sample of answers ends here to stop students looking up answers to their assessments. See contents page for details of the rest of the resource.