MAXIMUM MARK: 75
1 (a) Mark as follows:

High ← 63
X = 0
High ← Middle - 1

One mark for each correct line

(b) (i) ordered / in order

(ii) 6

(iii) 0
item not present in array
non zero
position of the item in the array

(c) (i) e.g. in Python:

def BinarySearch(Low, High):
    global Found
    if Low>High:
        return
    Middle=int((High+Low)/2)
    if SearchData[Middle] == SearchItem:
        Found = Middle
    elif SearchData[Middle] < SearchItem:
        BinarySearch(Middle + 1, High)
    elif SearchData[Middle] > SearchItem:
        BinarySearch(Low, Middle - 1)
    return

(d) BinarySearch(1, 63)

[Total: 15]
### 2 (a)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditions</td>
<td><strong>Group 1 tests</strong></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td><strong>Group 2 tests</strong></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td><strong>Group 3 tests</strong></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Actions</td>
<td><strong>Accepted</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Repair</strong></td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Rejected</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

Correct column 1  
Correct columns 2 and 3  
Correct column 4  
Correct columns 5–8

### 2 (b)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditions</td>
<td><strong>Group 1 tests</strong></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td><strong>Group 2 tests</strong></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td><strong>Group 3 tests</strong></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>–</td>
</tr>
<tr>
<td>Actions</td>
<td><strong>Accepted</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Repair</strong></td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Rejected</strong></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

Correct column 1  
Correct column 2  
Correct column 3  
Correct column 4  
Correct column 5

### 2 (c)

```python
def Reject():
    if ((G1Tests() == True and G2Tests() == False and
         G3Tests() == False)or G1Tests() == False):
        return True
```

Correct function header  
Correct if statement  
Correct return statement

[Total: 12]
(a) Start Jon → Kellie → Scarlett 0

*Mark as follows:*
Three correct items \[1\]
Indication of correct order with start and termination \[1\]

(b) Type ListNode
   - Pointer as Integer
   - Name As String

*EndType*

*Mark as follows:*
Record structure definition \[1\]
Pointer field definition \[1\]
Node data definition \[1\]

(c) Dim NameList[1..50] As ListNode

*Mark as follows:*
Appropriate size of array \[1\]
Use of user defined record type \[1\]

(d) (i)

<table>
<thead>
<tr>
<th>HeadPointer</th>
<th>Name</th>
<th>Pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(49)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

*Mark as follows:*
HeadPointer \[1\]
FreePointer \[1\]
Pointer[50] \[1\]
(ii) FOR Index ← 1 TO 49
    NameList[Index].Pointer ← Index + 1
ENDFOR
NameList[50].Pointer ← 0
HeadPointer ← 0
FreePointer ← 1

Mark as follows:
Correct FOR loop
Correct setting of Pointer[50], HeadPointer and FreePointer

(e) (i) 01 PROCEDURE AddItem(NewItem)
02 //
03    NameList[FreePointer].Name ← NewItem
04    CurrentPointer ← HeadPointer
05 //
06    REPEAT
07      IF NameList[CurrentPointer].Name < NewItem
08      THEN
09        PreviousPointer ← CurrentPointer
10        CurrentPointer ← NameList[CurrentPointer].Pointer
11      ENDIF
12    UNTIL NameList[CurrentPointer].Name > NewItem
13 //
14    IF CurrentPointer = HeadPointer
15      THEN
16      NameList[FreePointer].Pointer ← HeadPointer
17      HeadPointer ← FreePointer
18    ELSE
19      NameList[FreePointer].Pointer
20      ← NameList[PreviousPointer].Pointer
21      NameList[PreviousPointer] ← FreePointer
22    ENDIF
23    FreePointer ← NameList[FreePointer].Pointer
24 ENDPROCEDURE

(ii) New item placed in node at head of Free List

(iii) Loop that repeats until position of new item located
Records current pointer and then updates current pointer

(iv) Check to see whether new item is first in linked list
If first item then place item at head of list
If not first item then adjust pointers to place it in correct position in list

[Total: 22]
4 (a)

1 mark for each correctly labelled activity – max 4 marks [max 4]

(b) (i) 1 – 2 – 3 – 4 – 5 [1]
(ii) 20 [1]

(c) (i) 8 [1]
(ii) 17 [1]

[Total: 8]
(a) Mark as follows:

noOfSeats declaration and associated show method in PassengerVehicle [1]
inheritance arrows [1]
constructor method in Coach [1]
seatBeltsFitted declaration and associated show method in Coach [1]

(b) e.g. in Python:

class PassengerVehicle():
    def __init__(self, regNo, noOfSeats):
        # Sets all the initial values
        self.__ regNo = regNo
        self.__ noOfSeats = noOfSeats

    def showRegNo(self):
        print("Registration No: ",self.__regNo)

    def showNoOfSeats (self):
        print("No of seats: ",self.__noOfSeats)

Mark as follows:

data declarations [1]
use of __ in identifiers to give “private” attribute [1]
use of ‘self’ parameter [1]
showRegNo function [1]
showNoOfSeats function [1]
e.g. in Visual Basic:
MustInherit Class PassengerVehicle
  Protected regNo As String
  Protected noOfSeats As Integer

  Public Sub showRegNo()
    Console.WriteLine(regNo)
  End Sub

  Public Sub showNoOfSeats()
    Console.WriteLine(noOfSeats)
  End Sub

End Class

Mark as follows:

MustInherit [1]
data declarations [1]
protected [1]
showRegNo function [1]
showNoOfSeats function [1]

(c) e.g. in Python:
class Bus(PassengerVehicle):
  def __init__(self, regNo,
               noOfSeats, maxStanding):
      super().__init__(regNo, noOfSeats)
      self.__maxStanding = maxStanding

      def showMaxStanding (self):
          print("No of standing passengers: ", self.__maxStanding)

Mark as follows:
inheritance [1]
__init__ function header [1]
use of __init__ from superclass [1]
initialisations in __init__ function [1]
showMaxStanding function [1]
e.g. in Visual Basic:

```vbnet
Class Bus
    Inherits PassengerVehicle
    Private maxStanding As Integer
    Public Sub New(ByVal regNoValue As String, ByVal noOfSeatsValue As Integer, ByVal maxStandingValue As Integer)
        regNo = regNoValue
        noOfSeats = noOfSeatsValue
        maxStanding = maxStandingValue
    End Sub
    Public Sub ShowMaxStanding ()
        Console.WriteLine(maxStanding)
    End Sub
End Class
```

Mark as follows:

- inheritance [1]
- private [1]
- Public Sub New header [1]
- Initialisations in Sub New [1]
- ShowMaxStanding function [1]

(d) (i) e.g. in Python:
```
pv1 = Bus("NBR 123", 51, 10)
```

- e.g. in Visual Basic:
```
Dim pv1 As Bus = New Bus("NBR 123", 51, 10)
```

(ii) e.g. in Python:
```
pv1.showRegNo()  
pv1.showNoOfSeats()  
pv1.showMaxStanding()
```

- e.g. in Visual Basic
```
pv1.showRegNo()  
pv1.showNoOfSeats()  
pv1.showMaxStanding()
```

[Total: 18]