4(a)

MP1 Count-controlled

MP2 The number of iterations required is known

4(b)

I	Key	J	Chars[J]	Chars [1]	Chars [2]	Chars [3]	Chars [4]
2				'D'	'T'	'H'	'R'
	'T'	1	'D'				
3							
	'H'	2	'T'			'T'	
		1	'D'				
					'H'		
4							
	'R'	3	'T'				'T'
		2	'H'			('R')	
						'R'	
5							

MP1 MP2 MP3 MP4 MP5 MP6

`Chars` array final values

'D'	'H'	'R'	'T'

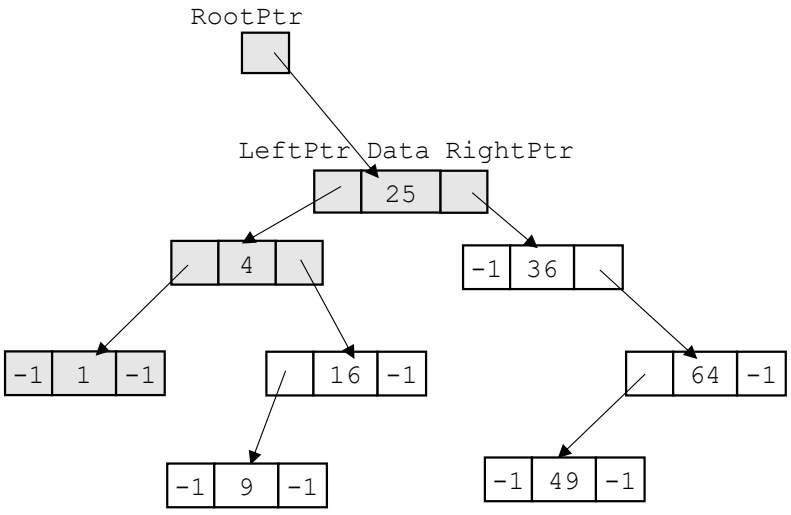
Mark as follows:

MP1, MP2, MP3, MP4, MP5
1 mark per enclosure

11(a)

One mark per mark point (**Max 4)**

MP1 Five additional nodes with correct data values
MP2 Correct null pointers in all nodes (7) – no extra null pointers where the arrow points to the next node
MP3 Correct arrows to represent pointers joining parent nodes to child nodes – must come from the correct left or right pointer not the middle
MP4 All nodes in correct order and no additional data in left/right pointer boxes.



1

11(b)

First **eight** rows of table (**Max 3**)
Three marks for all eight correct rows
Two marks for six or seven correct rows
One mark for three, four or five correct rows

Last row of table and FreePtr (**Max 1**)
One mark for correct `FreePtr` with last row of table completely blank

RootPtr
0

FreePtr
8

Index	LeftPtr	Data	RightPtr
0	1	25	2
1	3	4	4
2	-1	36	5
3	-1	1	-1
4	6	16	-1
5	7	64	-1
6	-1	9	-1
7	-1	49	-1
8			

11(c)

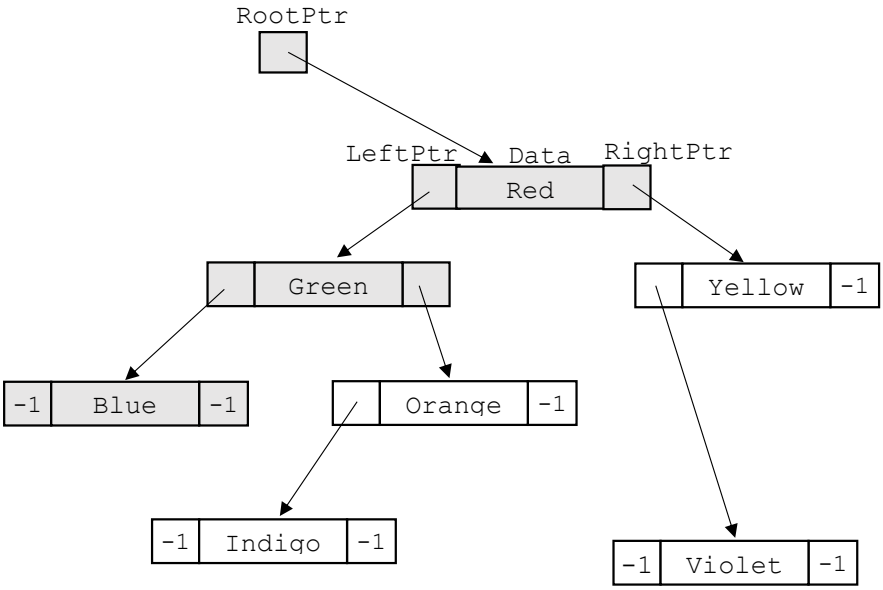
One mark for any correct row (**Max 4**)

```
FUNCTION SearchList(Item : INTEGER) RETURNS INTEGER
   NullPtr ← -1
   NowPtr ← RootPtr
   WHILE NowPtr <> NullPtr
      IF LinkList[NowPtr].Data < Item THEN
         NowPtr ← LinkList[NowPtr].RightPtr
      ELSE
         IF LinkList[NowPtr].Data > Item THEN
            NowPtr ← LinkList[NowPtr].LeftPtr
         ELSE
            RETURN NowPtr
         ENDIF
      ENDIF
   ENDWHILE
   RETURN NullPtr
ENDFUNCTION
```

11(a)

One mark per mark point (**Max 4)**

MP1 Four additional nodes with correct data values
MP2 Correct null pointers in all added nodes (6) with no extra null pointers where the arrow points to the next node
MP3 Correct arrows to represent pointers joining parent nodes to child nodes
MP4 All nodes in correct order and no extra data added to pointers.



11(b)

One mark per mark point (**Max 4)**

MP1 Correct `Red` and `Green` rows
MP2 Correct `Yellow` and `Blue` rows
MP3 Correct `Orange`, `Indigo` and `Violet` rows
MP4 Correct `FreePtr` with blank row 7

RootPtr
0

FreePtr
7

Index	LeftPtr	Data	RightPtr
0	1	Red	2
1	3	Green	4
2	6	Yellow	-1
3	-1	Blue	-1
4	5	Orange	-1
5	-1	Indigo	-1
6	-1	Violet	-1
7			

11(c)

One mark for any correct row (**Max 4**)

```
FUNCTION SearchTree(Item : STRING) RETURNS INTEGER
   NowPtr ← RootPtr
   WHILE NowPtr <> -1
      IF BinTree[NowPtr].Data > Item THEN
         NowPtr ← BinTree[NowPtr].LeftPtr
      ELSE
         IF BinTree[NowPtr].Data < Item THEN
            NowPtr ← BinTree[NowPtr].RightPtr
         ELSE
            RETURN NowPtr
         ENDIF
      ENDIF
   ENDWHILE
   RETURN NowPtr
ENDFUNCTION
```


1(a)

1 mark each to max 6:

- Function declaration (and close where appropriate)
- Declaration/use of an array (with space/initialised with 45 spaces/strings)
- Opening the file Data.txt for read and closing in an appropriate place
- Looping through all file contents/Looping 45 times and reading each line …
- … storing all items from file into array
- Returning the populated array
- Exception handling with suitable try, catch and output

e.g.

Python

```
def ReadData():
    Colours = []
    try:
        File = open("Data.txt")
        Colours = File.read().split("\n")
        File.close()
        return Colours
    except:
        print("No file found")
```

VB.NET

```
Function ReadData()
        Dim TextFile As String = "Data.txt"

        Dim Colours(45) As String
        Try
            Dim FileReader As New System.IO.StreamReader(TextFile)
            For x = 0 To 45
                Colours(x) = FileReader.ReadLine()
```

1(a)

```
            Next
            FileReader.Close()
        Catch ex As Exception
            Console.WriteLine("No file found")
        End Try
        Return Colours
End Function
```

Java

```
public static String[] ReadData(){
       String TextFile = "Data.txt";
       String Colours[] = new String[45];

       try{
           FileReader f = new FileReader(TextFile);
           BufferedReader Reader = new BufferedReader(f);
           for(Integer X = 0; X < 45; X++){
                try{
                     Colours[X] = Reader.readLine();
                }catch(IOException ex){}
           }
           try{
                Reader.close();
           }catch(IOException ex){}
           return Colours;
       }catch(FileNotFoundException e){
           System.out.println("File not found");
       }
       return Colours;
}
```

1(b)(i)

1 mark each

- Function header (and end where appropriate) taking (min) one parameter
- Looping through each parameter array element, concatenating with space and returning

Python

```
def FormatArray(DataArray):
    OutputText = ""
    for x in range(0, 45):
        OutputText = OutputText + DataArray[x] + " "
    return OutputText
```

VB.NET

```
Function FormatArray(DataArray)
    Dim OutputText As String = ""
    For X = 0 To 44
        OutputText = OutputText & DataArray(X) & " "
    Next
    Return OutputText
End Function
```

Java

```
public static String FormatArray(String[] DataArray){
       String OutputText = "";
       for(Integer X = 0; X < 45; X++){
           OutputText = OutputText + DataArray[X] + " ";
       }
       return OutputText;
}
```

1(b)(ii)	1 mark each:
	• Calling `ReadData()` and storing returned array …
	• … calling `FormatArray()` with returned array
	• Outputting return value from `FormatArray()`
	
	Python
	`Colours = ReadData() #string array`
	`print(FormatArray(Colours))`
	
	VB.NET
	`Dim Colours(45) As String`
	`Colours = ReadData()`
	`Console.WriteLine(FormatArray(Colours))`
	
	Java
	`String[] Colours = new String[45];`
	`Colours = ReadData();`
	`System.out.println(FormatArray(Colours));`

1(b)(iii)	1 mark for output showing all colours in one string e.g. beige green scarlet silver bronze slate yellow orange jade lavender magenta magnolia turquoise black grey russet mango maroon mint purple red pink white cream navy olive brown violet cyan amber aqua azure copper fawn fuschia gold indigo ivory mauve mulberry peach periwinkle plum rose sage

1(c)	1 mark each
	• Function header (and close where appropriate) taking (min) two parameters and returns a value in all cases
	• Looping through each character in each string parameter …
	• … return 1 when first parameter < second
	• … return 2 when first parameter > second
	
	e.g.
	Python
	`def CompareStrings(First, Second):`
	`  Count = 0`
	`  while True:`
	`  if First[Count] < Second[Count]:`
	`  return 1`
	`  elif First[Count] > Second[Count]:`
	`  return 2`
	`  else:`
	`  Count = Count + 1`
	
	VB.NET
	`Function CompareStrings(FirstS, SecondS)`
	`  Dim Count As Integer = 1`
	`  While (True)`
	`  If Mid(FirstS, Count, 1) < Mid(SecondS, Count, 1) Then`
	`  Return 1`
	`  ElseIf Mid(FirstS, Count, 1) > Mid(SecondS, Count, 1) Then`
	`  Return 2`
	`  Else`
	`  Count = Count + 1`
	`  End If`
	`  End While`
	`End Function`

1(c)

Java

```
public static Integer CompareStrings(String First, String Second){
       Integer Count = 0;
       while(true){
           if(First.substring(Count, Count + 1).compareTo(Second.substring(Count, Count +
1)) < 0){
           return 1;
           }else if(First.substring(Count, Count + 1).compareTo(Second.substring(Count,
Count + 1))>0){
                 return 2;
           }else{
                 Count++;
           }
       }
}
```

1(d)(i)

1 mark each

- Bubble sort function header taking array parameter and returns **sorted** array in all cases
- Comparing strings using `CompareStrings()` **and** correctly swapping values when needed
- Correct bubble sort that sorts the data correctly

Python

```
def Bubble(DataArray):
    ArrayLength = len(DataArray)

    for x in range(ArrayLength - 1):
        for y in range(0, ArrayLength - x - 1):
            Result = CompareStrings(DataArray[y], DataArray[y + 1])
            if Result == 2:
                DataArray[y], DataArray[y+1] = DataArray[y+1], DataArray[y]
    return DataArray
```

1(d)(i)

VB.NET

```
Function Bubble(DataArray)
        Dim ArrayLength As Integer = 45
        Dim Result As Integer
        Dim Temp As String

        For X = 0 To ArrayLength - 1
            For Y = 0 To ArrayLength - X - 2
                Result = CompareStrings(DataArray(Y), DataArray(Y + 1))
                If Result = 2 Then
                    Temp = DataArray(Y)
                    DataArray(Y) = DataArray(Y + 1)
                    DataArray(Y + 1) = Temp
                End If
            Next
        Next
        Return DataArray

End Function
```

Java

```
public static String[] Bubble(String[] DataArray){
       Integer ArrayLength = 45;
       Integer Result;
       String Temp;
       for(Integer X = 0; X < ArrayLength ; X++){
           for(Integer Y = 0; Y < ArrayLength - X - 1; Y++){
                 Result = CompareStrings(DataArray[Y], DataArray[Y+1]);
```

1(d)(i)

```
                if (Result == 2){
                    Temp = DataArray[Y];
                    DataArray[Y] = DataArray[Y+1];
                    DataArray[Y+1] = Temp;
                }
            }
        }
        return DataArray;
}
```

1(d)(ii)

1 mark each
- Calling `Bubble()` with array as parameter **and** using/storing return value
- Calling `FormatArray()` with **return value** from `Bubble()` **and** outputting return value

Python
```
BubbleSorted = Bubble(Colours)
print(FormatArray(BubbleSorted))
```

VB.NET
```
Dim BubbleSorted(45) As String
BubbleSorted = Bubble(Colours)
Console.WriteLine(FormatArray(BubbleSorted))
```

Java
```
String[] BubbleSorted = new String[45];
BubbleSorted = Bubble(Colours);
System.out.println(FormatArray(BubbleSorted));
```

1(d)(iii)

1 mark for sorted data

e.g.

beige green scarlet silver bronze slate yellow orange jade lavender magenta magnolia turquoise black grey russet mango maroon mint purple red pink white cream navy olive brown violet cyan amber aqua azure copper fawn fuschia gold indigo ivory mauve mulberry peach periwinkle plum rose sage
amber aqua azure beige black bronze brown copper cream cyan fawn fuschia gold green grey indigo ivory jade lavender magenta magnolia mango maroon mauve mint mulberry navy olive orange peach periwinkle pink plum purple red rose russet sage scarlet silver slate turquoise violet white yellow

1(a)

1 mark each to max 6:
- Function declaration (and close where appropriate)
- Declaration/use of an array (with space/initialised with 45 spaces/strings)
- Opening the file Data.txt for read and closing in an appropriate place
- Looping through all file contents/Looping 45 times and reading each line …
- … storing all items from file into array
- Returning the populated array
- Exception handling with suitable try, catch and output

e.g.
Python
```
def ReadData():
    Colours = []
    try:
        File = open("Data.txt")
        Colours = File.read().split("\n")
        File.close()
        return Colours
    except:
        print("No file found")
```

VB.NET
```
Function ReadData()
        Dim TextFile As String = "Data.txt"

        Dim Colours(45) As String
        Try
            Dim FileReader As New System.IO.StreamReader(TextFile)
            For x = 0 To 45
                Colours(x) = FileReader.ReadLine()
```

1(a)

```
            Next
            FileReader.Close()
        Catch ex As Exception
            Console.WriteLine("No file found")
        End Try
        Return Colours
End Function
```

Java

```
public static String[] ReadData(){
       String TextFile = "Data.txt";
       String Colours[] = new String[45];

       try{
           FileReader f = new FileReader(TextFile);
           BufferedReader Reader = new BufferedReader(f);
           for(Integer X = 0; X < 45; X++){
                try{
                     Colours[X] = Reader.readLine();
                }catch(IOException ex){}
           }
           try{
                Reader.close();
           }catch(IOException ex){}
           return Colours;
       }catch(FileNotFoundException e){
           System.out.println("File not found");
       }
       return Colours;
}
```

1(b)(i)

1 mark each

- Function header (and end where appropriate) taking (min) one parameter
- Looping through each parameter array element, concatenating with space and returning

Python

```
def FormatArray(DataArray):
    OutputText = ""
    for x in range(0, 45):
        OutputText = OutputText + DataArray[x] + " "
    return OutputText
```

VB.NET

```
Function FormatArray(DataArray)
    Dim OutputText As String = ""
    For X = 0 To 44
        OutputText = OutputText & DataArray(X) & " "
    Next
    Return OutputText
End Function
```

Java

```
public static String FormatArray(String[] DataArray){
       String OutputText = "";
       for(Integer X = 0; X < 45; X++){
           OutputText = OutputText + DataArray[X] + " ";
       }
       return OutputText;
}
```

1(b)(ii)	1 mark each:
	• Calling `ReadData()` and storing returned array …
	• … calling `FormatArray()` with returned array
	• Outputting return value from `FormatArray()`
	
	Python
	`Colours = ReadData() #string array`
	`print(FormatArray(Colours))`
	
	VB.NET
	`Dim Colours(45) As String`
	`Colours = ReadData()`
	`Console.WriteLine(FormatArray(Colours))`
	
	Java
	`String[] Colours = new String[45];`
	`Colours = ReadData();`
	`System.out.println(FormatArray(Colours));`

1(b)(iii)	1 mark for output showing all colours in one string e.g. beige green scarlet silver bronze slate yellow orange jade lavender magenta magnolia turquoise black grey russet mango maroon mint purple red pink white cream navy olive brown violet cyan amber aqua azure copper fawn fuschia gold indigo ivory mauve mulberry peach periwinkle plum rose sage

1(c)	1 mark each
	• Function header (and close where appropriate) taking (min) two parameters and returns a value in all cases
	• Looping through each character in each string parameter …
	• … return 1 when first parameter < second
	• … return 2 when first parameter > second
	
	e.g.
	Python
	`def CompareStrings(First, Second):`
	`  Count = 0`
	`  while True:`
	`  if First[Count] < Second[Count]:`
	`  return 1`
	`  elif First[Count] > Second[Count]:`
	`  return 2`
	`  else:`
	`  Count = Count + 1`
	
	VB.NET
	`Function CompareStrings(FirstS, SecondS)`
	`  Dim Count As Integer = 1`
	`  While (True)`
	`  If Mid(FirstS, Count, 1) < Mid(SecondS, Count, 1) Then`
	`  Return 1`
	`  ElseIf Mid(FirstS, Count, 1) > Mid(SecondS, Count, 1) Then`
	`  Return 2`
	`  Else`
	`  Count = Count + 1`
	`  End If`
	`  End While`
	`End Function`

1(c)

Java

```
public static Integer CompareStrings(String First, String Second){
       Integer Count = 0;
       while(true){
           if(First.substring(Count, Count + 1).compareTo(Second.substring(Count, Count +
1)) < 0){
           return 1;
           }else if(First.substring(Count, Count + 1).compareTo(Second.substring(Count,
Count + 1))>0){
                 return 2;
           }else{
                 Count++;
           }
       }
}
```

1(d)(i)

1 mark each

- Bubble sort function header taking array parameter and returns **sorted** array in all cases
- Comparing strings using `CompareStrings()` **and** correctly swapping values when needed
- Correct bubble sort that sorts the data correctly

Python

```
def Bubble(DataArray):
    ArrayLength = len(DataArray)

    for x in range(ArrayLength - 1):
        for y in range(0, ArrayLength - x - 1):
            Result = CompareStrings(DataArray[y], DataArray[y + 1])
            if Result == 2:
                DataArray[y], DataArray[y+1] = DataArray[y+1], DataArray[y]
    return DataArray
```

1(d)(i)

VB.NET

```
Function Bubble(DataArray)
        Dim ArrayLength As Integer = 45
        Dim Result As Integer
        Dim Temp As String

        For X = 0 To ArrayLength - 1
            For Y = 0 To ArrayLength - X - 2
                Result = CompareStrings(DataArray(Y), DataArray(Y + 1))
                If Result = 2 Then
                    Temp = DataArray(Y)
                    DataArray(Y) = DataArray(Y + 1)
                    DataArray(Y + 1) = Temp
                End If
            Next
        Next
        Return DataArray

End Function
```

Java

```
public static String[] Bubble(String[] DataArray){
       Integer ArrayLength = 45;
       Integer Result;
       String Temp;
       for(Integer X = 0; X < ArrayLength ; X++){
           for(Integer Y = 0; Y < ArrayLength - X - 1; Y++){
                 Result = CompareStrings(DataArray[Y], DataArray[Y+1]);
```

1(d)(i)

```
                if (Result == 2){
                      Temp = DataArray[Y];
                      DataArray[Y] = DataArray[Y+1];
                      DataArray[Y+1] = Temp;
                }
            }
        }
        return DataArray;
}
```

1(d)(ii)

1 mark each

- Calling `Bubble()` with array as parameter **and** using/storing return value
- Calling `FormatArray()` with **return value** from `Bubble()` **and** outputting return value

Python

```
BubbleSorted = Bubble(Colours)
print(FormatArray(BubbleSorted))
```

VB.NET

```
Dim BubbleSorted(45) As String
BubbleSorted = Bubble(Colours)
Console.WriteLine(FormatArray(BubbleSorted))
```

Java

```
String[] BubbleSorted = new String[45];
BubbleSorted = Bubble(Colours);
System.out.println(FormatArray(BubbleSorted));
```

1(d)(iii)

1 mark for sorted data

e.g.

beige green scarlet silver bronze slate yellow orange jade lavender magenta magnolia turquoise black grey russet mango maroon mint purple red pink white cream navy olive brown violet cyan amber aqua azure copper fawn fuschia gold indigo ivory mauve mulberry peach periwinkle plum rose sage

amber aqua azure beige black bronze brown copper cream cyan fawn fuschia gold green grey indigo ivory jade lavender magenta magnolia mango maroon mauve mint mulberry navy olive orange peach periwinkle pink plum purple red rose russet sage scarlet silver slate turquoise violet white yellow