CAMBRIDGE
International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/41**

Paper 4 Practical **October/November 2025**

MARK SCHEME

Maximum Mark: 75

Published

This document consists of **36** printed pages.

Generic Marking Principles

These general marking principles must be applied by all examiners when marking candidate answers. They should be applied alongside the specific content of the mark scheme or generic level descriptions for a question. Each question paper and mark scheme will also comply with these marking principles.

GENERIC MARKING PRINCIPLE 1:

Marks must be awarded in line with:

- the specific content of the mark scheme or the generic level descriptors for the question
- the specific skills defined in the mark scheme or in the generic level descriptors for the question
- the standard of response required by a candidate as exemplified by the standardisation scripts.

GENERIC MARKING PRINCIPLE 2:

Marks awarded are always **whole marks** (not half marks, or other fractions).

GENERIC MARKING PRINCIPLE 3:

Marks must be awarded **positively**:

- marks are awarded for correct/valid answers, as defined in the mark scheme. However, credit is given for valid answers which go beyond the scope of the syllabus and mark scheme, referring to your Team Leader as appropriate
- marks are awarded when candidates clearly demonstrate what they know and can do
- marks are not deducted for errors
- marks are not deducted for omissions
- answers should only be judged on the quality of spelling, punctuation and grammar when these features are specifically assessed by the question as indicated by the mark scheme. The meaning, however, should be unambiguous.

GENERIC MARKING PRINCIPLE 4:

Rules must be applied consistently, e.g. in situations where candidates have not followed instructions or in the application of generic level descriptors.

GENERIC MARKING PRINCIPLE 5: Marks should be awarded using the full range of marks defined in the mark scheme for the question (however; the use of the full mark range may be limited according to the quality of the candidate responses seen).
GENERIC MARKING PRINCIPLE 6: Marks awarded are based solely on the requirements as defined in the mark scheme. Marks should not be awarded with grade thresholds or grade descriptors in mind.

Annotations guidance for centres

Examiners use a system of annotations as a shorthand for communicating their marking decisions to one another. Examiners are trained during the standardisation process on how and when to use annotations. The purpose of annotations is to inform the standard isation and monitoring processes and guide the supervising examiners when they are checking the work of examiners within their team. The meaning of annotations and how they are used is specific to each component and is understood by all examiners who mark the component.

We publish annotations in our mark schemes to help centres understand the annotations they may see on copies of scripts. Note that there may not be a direct correlation between the number of annotations on a script and the mark awarded. Similarly, the use of an annotation may not be an indication of the quality of the response.

The annotations listed below were available to examiners marking this component in this series.

Annotations

Annotation	**Meaning**
BOD	Benefit of the doubt
λ	To indicate where a key word/phrase/code is missing
✗	Incorrect
FT	Follow through
〰	Indicate a point in an answer
Highlighted text	To draw attention to a particular aspect or to indicate where parts of an answer have been combined
I	Ignore
NAQ	Not answered question
NBOD	No benefit of doubt given
NE	No examples or not enough

Annotation	**Meaning**
≀	Not relevant or used to separate parts of an answer
Off-page comment	Allows comments to be entered at the bottom of the RM marking window and then displayed when the associated question item is navigated to.
REP	Repetition
SEEN	Indicates that work or a page has been seen including blank answer spaces and blank pages.
✓	Correct
TV	Too vague

Mark scheme abbreviations

- **Bold** in mark scheme means that idea is required.
- / in mark scheme means alternative.
- // in mark scheme means alternative solution that gains the same mark point.
- … at the end of one mark point without a … at the start of the next just means the sentence follows on. There is no dependency.
- … at the end of one mark point and … at the start of the next, this means the second cannot be awarded without the first.
- () means what is in the brackets is not required, or it is not required in some languages but may be required in others.

Question	Answer	Marks

1(a)	1 mark each
	• (Global) 1D array initialised with 30 null values
	• (Global) `TopofStack` initialised with `-1`
	
	Example program code
	
	Java
	```
	public static Integer[] Stack = new Integer[30];
	public static Integer TopOfStack;
	public static void main(String args[]){
	      for(Integer X = 0; X < 30; X++){
	            Stack[X] = null;
	      }
	      TopOfStack = -1;
	}
	```
	VB.NET
	```
	Dim Stack(29) As Integer
	 Dim TopOfStack As Integer
	Sub Main(args As String())
	   For x = 0 To 29
	       Stack(x) = Nothing
	   Next
	   TopOfStack = -1
	End Sub
	```
	Python
	```
	Stack = [None for x in range(30)]
	TopOfStack = -1
	```
		2

Question	Answer	Marks

1(b)	1 mark each
	• Function header (and end) taking one parameter, returning Boolean in all instances
	• Checking if stack is full (`TopOfStack = 29`) **and**, if it is, returning `FALSE`
	• Incrementing `TopofStack`
	• (Otherwise) Storing parameter in incremented `TopOfStack` position **and** returning `TRUE`
	
	Example program code
	
	Java
	`public static Boolean Push(Integer DataToPush){`
	`      if(TopOfStack < 29){`
	`            TopOfStack++;`
	`            Stack[TopOfStack] = DataToPush;`
	`            return true;`
	`      }`
	`      return false;`
	`}`
	
	VB.NET
	`Function Push(DataToPush)`
	`    If TopOfStack < 29 Then`
	`        TopOfStack = TopOfStack + 1`
	`        Stack(TopOfStack) = DataToPush`
	`        Return True`
	`    End If`
	`    Return False`
	`End Function`
		4

Question	Answer	Marks

1(b)	Python
	`def Push(DataToPush):`
	`    global Stack`
	`    global TopOfStack`
	`    if TopOfStack < 29:`
	`        TopOfStack = TopOfStack + 1`
	`        Stack[TopOfStack] = DataToPush`
	`        return True`
	`    else:`
	`        return False`	

Question	Answer	Marks

1(c)	1 mark each
	• Function header (and end) returning integer in all cases
	• Checking if stack empty (`TopofStack = -1`) **and** returning `-999` when true
	• (Otherwise) Accessing **and** returning item at `TopOfStack` (before it's decremented)
	• Decrementing `TopofStack`
	
	Example program code
	
	Java
	`public static Integer Pop(){`
	`      Integer DataReturn;`
	`      if(TopOfStack == -1){`
	`            return -999;`
	`      }`
	`      DataReturn = Stack[TopOfStack];`
	`      TopOfStack--;`
	`      return DataReturn;`
	`}`
	
	VB.NET
	`Function Pop()`
	`    If TopOfStack = -1 Then`
	`        Return -999`
	`    Else`
	`         Dim DataReturn As Integer = Stack(TopOfStack)`
	`         TopOfStack = TopOfStack - 1`
	`         Return DataReturn`
	`     End If`
	`End Function`
		4

Question	Answer	Marks

1(c)	Python
	`def Pop():`
	`    global Stack`
	`    global TopOfStack`
	`    if TopOfStack == -1:`
	`        return -999`
	`    else:`
	`        DataReturn = Stack[TopOfStack]`
	`        TopOfStack = TopOfStack - 1`
	`        return DataReturn`	

Question	Answer	Marks

1(d)	1 mark each
	• Looping 40 times
	• Generating random number between 0 and 1000 inclusive inside the loop
	• Calling `Push()` with each random number **and** storing/using return value … … if return value is `FALSE` output `Stack full` and breaking out of loop
	
	Example program code
	
	Java
	`for(Integer X = 0; X < 40; X++){`
	`      Pushed = Push(RandomNumber.nextInt(1001));`
	`      if(Pushed == false){`
	`            System.out.println("Stack full");`
	`            X = 40;`
	`      }`
	`}`
	
	VB.NET
	`For x = 0 To 39`
	`    Pushed = Push(RandomNumber.Next(0, 1000))`
	`    If Pushed = False Then`
	`       Console.WriteLine("Stack full")`
	`       x = 40`
	`    End If`
	`Next`
	
	Python
	`for x in range(40):`
	`    Pushed = Push(random.randint(0,1000))`
	`    if Pushed == False:`
	`        print("Stack full")`
	`        break`
		4

Question	Answer	Marks

1(e)	1 mark each
	• Procedure header (and end) **and** output of highest and lowest include appropriate messages
	• Calls `Pop()` until there are no items left in stack (`return value = -999 // TopOfStack  = -1`) **and** storing/using return values
	• Finds **and** outputs highest value from returned values
	• Finds **and** outputs lowest value from returned values
	
	Example program code
	
	Java
	`public static void FindValues(){`
	`      Integer Highest;`
	`      Integer Lowest;`
	`      Highest = Pop();`
	`      Lowest = Highest;`
	`      Integer ReturnValue = Highest;`
	`      while(ReturnValue != -999){`
	`            if(ReturnValue > Highest){`
	`                  Highest = ReturnValue;`
	`            }`
	`            if(ReturnValue < Lowest){`
	`                  Lowest = ReturnValue;`
	`            }`
	`            ReturnValue = Pop();`
	`      }`
	`      System.out.println("The highest value is " + Highest + " and the lowest value is " +`
	`Lowest);`
	`}`
		4

Question	Answer	Marks

1(e)	VB.NET
	`Sub FindValues()`
	`    Dim Highest, Lowest As Integer`
	`    Highest = Pop()`
	`    Lowest = Highest`
	`    Dim ReturnValue As Integer = Highest`
	`    While ReturnValue <> -999`
	`        If ReturnValue > Highest Then`
	`           Highest = ReturnValue`
	`        End If`
	`        If ReturnValue < Lowest Then`
	`           Lowest = ReturnValue`
	`        End If`
	`        ReturnValue = Pop()`
	`    End While`
	`    Console.WriteLine("The highest value is " & Highest & " and the lowest value is " & Lowest)`
	`End Sub`
	
	Python
	`def FindValues():`
	`    Highest = Pop()`
	`    Lowest = Highest`
	`    ReturnValue = Lowest`
	`    while(ReturnValue != -999):`
	`        if ReturnValue > Highest:`
	`            Highest = ReturnValue`
	`        if ReturnValue < Lowest:`
	`            Lowest = ReturnValue`
	`        ReturnValue = Pop()`
	`    print("The highest value is", Highest, "and the lowest value is", Lowest)`	

Question	Answer	Marks

1(f)(i)	1 mark for calling `FindValues()`
	
	Example program code
	
	Java
	`FindValues();`
	
	VB.NET
	`FindValues()`
	
	Python
	`FindValues()`
		1

1(f)(ii)	1 mark for a screenshot of output showing Stack full output **once** Lowest value output in an appropriate message Highest value output in an appropriate message Lowest and Highest must be 0–1000 inclusive	1

Question	Answer	Marks

2(a)(i)	1 mark each
	• Class header (and end)
	• Declaration of 2 private attributes with correct data types
	• Constructor header (and end) within class with 2 parameters …
	• … assigning parameters to attributes
	
	Example program code
	
	Java
	`class Train{`
	`      private String Number;`
	`      private Integer Route;`
	
	`      public Train(String pNumber, Integer pRoute){`
	`            Number = pNumber;`
	`            Route = pRoute;`
	`      }}`
	
	VB.NET
	`Class Train`
	`    Private TrainIDNumber As String`
	`    Private Route As Integer`
	
	`    Sub New(pNumber, pRoute)`
	`        TrainIDNumber = pNumber`
	`        Route = pRoute`
	`    End Sub`
	`End Class`
	
	Python
	`class Train():`
	`    def __init__(self, pNumber, pRoute):`
	`        self.__TrainIDNumber = pNumber #string`
	`        self.__Route = pRoute #integer`
		4

Question	Answer	Marks

2(a)(ii)	1 mark each
	• One get method header (and end) with no parameter …
	• … returning correct attribute
	• Second correct get method
	
	Example program code
	
	Java
	`public String GetTrainNumber(){`
	`      return Number;`
	`}`
	`public Integer GetRoute(){`
	`      return Route;`
	`}`
	
	VB.NET
	`Function GetTrainIDNumber()`
	`    Return TrainIDNumber`
	`End Function`
	
	`Function GetRoute()`
	`    Return Route`
	`End Function`
	
	Python
	`def GetTrainIDNumber(self):`
	`    return self.__TrainIDNumber`
	
	`def GetRoute(self):`
	`    return self.__Route`
		3

Question	Answer	Marks

2(b)	1 mark each
	• One instance of train with correct arguments **and** stored in a variable/structure
	• Remaining three instances correct
	
	Example program code
	
	Java
	`Train FirstTrain = new Train("12ADV", 134);`
	`Train SecondTrain  = new Train("33ART", 20);`
	`Train ThirdTrain  = new Train("9FKF", 3);`
	`Train FourthTrain  = new Train("21VBC", 24)`
	
	VB.NET
	`Dim FirstTrain As Train = New Train("12ADV", 134)`
	`Dim SecondTrain As Train = New Train("33ART", 20)`
	`Dim ThirdTrain As Train = New Train("9FKF", 3)`
	`Dim FourthTrain As Train = New Train("21VBC", 24)`
	
	Python
	`FirstTrain = Train("12ADV",134)`
	`SecondTrain = Train("33ART",20)`
	`ThirdTrain = Train("9FKF",3)`
	`FourthTrain = Train("21VBC",24)`
		2

Question	Answer	Marks

2(c)(i)	1 mark each
	• Class header (and end) with four private attributes with appropriate data types
	• Constructor header (and end) within class taking 2 parameters …
	• … assigning parameters to attributes, initialising `NumberTrains` to `0`, initialising `Trains` to an empty array
	
	Example program code
	
	Java
	`class Station{`
	`      private String StationID;`
	`      private Integer NumberPlatforms;`
	`      private Train[] Trains = new Train[10];`
	`      private Integer NumberTrains;`
	
	`      public Station(String pID, Integer pNumberOfPlatforms){`
	`            StationID = pID;`
	`            NumberPlatforms = pNumberOfPlatforms;`
	`            NumberTrains = 0;`
	`      }}`
	
	VB.NET
	`Class Station`
	`    Private StationID As String`
	`    Private NumberPlatforms As Integer`
	`    Private Trains(9) As Train`
	`    Private NumberTrains As Integer`
	
	`    Sub New(pID, pNumberOfPlatforms)`
	`        StationID = pID`
	`        NumberPlatforms = pNumberOfPlatforms`
	`        NumberTrains = 0`
	`    End Sub`
	`End Class`
		3

Question	Answer	Marks

2(c)(i)	Python
	`class Station():`
	`    def __init__(self, pID, pNumberOfPlatforms):`
	`        self.__StationID = pID #string`
	`        self.__NumberPlatforms = pNumberOfPlatforms #integer`
	`        self.__Trains = [] #train 10 elements`
	`        self.__NumberTrains = 0 #integer`	

Question	Answer	Marks

2(c)(ii)	1 mark each
	• Method header (and close) taking one `Train` parameter
	• Checking if all platforms are full **and** returning `FALSE`
	• (Otherwise) Storing parameter in array `Trains` …
	• … incrementing `NumberTrains` **and** returning `True`
	
	Example program code
	
	Java
	`public Boolean AddTrain(Train NewTrain){`
	`      if(NumberTrains >= NumberPlatforms){`
	`            return false;`
	`      }`
	`      Trains[NumberTrains] = NewTrain;`
	`      NumberTrains++;`
	`      return true;`
	`}`
	
	VB.NET
	`Function AddTrain(NewTrain)`
	`    If NumberTrains >= NumberPlatforms Then`
	`        Return False`
	`    End If`
	`    Trains(NumberTrains) = NewTrain`
	`    NumberTrains = NumberTrains + 1`
	`    Return True`
	`End Function`
	
	Python
	`def AddTrain(self, NewTrain):`
	`    if self.__NumberTrains >= self.__NumberPlatforms:`
	`        return False`
	`    else:`
	`        self.__Trains.append(NewTrain)`
	`        self.__NumberTrains += 1`
	`        return True`
		4

Question	Answer	Marks

2(c)(iii)	1 mark each
	• Method header (and close) **and** returning a string in all cases
	• Checking if no trains **and** returning `"There are no trains"`
	• (Otherwise) Looping through each train in the station …
	• … accessing train ID number and route number using get methods
	• … creating a string with ID number and route number for each train
	• … returning correctly formatted **string**
	
	Example program code
	
	Java
	`public String GetTrains(){`
	`      if(NumberTrains == 0){`
	`            return "There are no trains";`
	`      }`
	`      String OutputLine = "The trains at station " + StationID + " are: \n";`
	`      for(Integer x =0; x < NumberTrains; x++){`
	`            OutputLine = OutputLine + Trains[x].GetTrainNumber() + " on route number " +`
	`Trains[x].GetRoute() + "\n";`
	`      }`
	`      return OutputLine;`
	`}`
	
	VB.NET
	`Function GetTrains()`
	`    If NumberTrains = 0 Then`
	`        Return "There are no trains"`
	`    End If`
	`    Dim OutputLine As String = "The trains at station " & StationID & " are:" & vbNewLine`
	`    For x = 0 To NumberTrains - 1`
	`       OutputLine = OutputLine & Trains(x). GetTrainIDNumber() & " on route number " &`
	`Trains(x).GetRoute() & vbNewLine`
	`    Next`
	`    Return OutputLine`
	`End Function`
		6

Question	Answer	Marks

2(c)(iii)	Python
	```
	def GetTrains(self):
	    if self.__NumberTrains == 0:
	        return "There are no trains"
	    OutputLine = "The trains at station " + self.__StationID + " are: \n"
	    for x in range(self.__NumberTrains):
	        OutputLine = OutputLine + self.__Trains[x]. GetTrainIDNumber() + " on route number
	" + str(self.__Trains[x].GetRoute()) + "\n"
	    return OutputLine
	```

2(d)(i)	1 mark each
	• One instance of `Station` created with correct arguments and stored
	• Second correct instance and stored
	
	Example program code
	
	Java
	```
	Station SouthStation = new Station("STH", 2);
	Station NorthStation  = new Station("NTH", 1);
	```
	VB.NET
	```
	Dim SouthStation As Station = New Station("STH", 2)
	Dim NorthStation As Station = New Station("NTH", 1)
	```
	Python
	```
	SouthStation = Station("STH",2)
	NorthStation = Station("NTH",1)
	```
		2

Question	Answer	Marks

2(d)(ii)	1 mark each
	• Calling `AddTrain` for 3 correct trains for station STH **once**
	• Calling `AddTrain` for 1 correct train for station NTH **once**
	• Outputting `"Station is full"` if **any** return value is `FALSE`
	• Calling `GetTrains()` for both stations **and** outputting return values
	
	Example program code
	
	Java
	`Boolean ReturnValue  = SouthStation.AddTrain(FirstTrain);`
	`if(ReturnValue == false) {`
	`       System.out.println("Station is full");`
	`}`
	`ReturnValue = SouthStation.AddTrain(SecondTrain);`
	`if(ReturnValue == false) {`
	`      System.out.println("Station is full");`
	`}`
	`ReturnValue = SouthStation.AddTrain(ThirdTrain);`
	`if(ReturnValue == false) {`
	`      System.out.println("Station is full");`
	`}`
	
	`ReturnValue = NorthStation.AddTrain(FourthTrain);`
	`if(ReturnValue == false) {`
	`      System.out.println("Station is full");`
	`}`
	`System.out.println(SouthStation.GetTrains());`
	`System.out.println(NorthStation.GetTrains());`
	
	VB.NET
	`Dim ReturnValue As Boolean = SouthStation.AddTrain(FirstTrain)`
	`If ReturnValue = False Then`
	`    Console.WriteLine("Station is full")`
	`End If`
	`ReturnValue = SouthStation.AddTrain(SecondTrain)`
	`If ReturnValue = False Then`
		4

Question	Answer	Marks

2(d)(ii)	
	```
	   Console.WriteLine("Station is full")
	End If
	ReturnValue = SouthStation.AddTrain(ThirdTrain)
	If ReturnValue = False Then
	   Console.WriteLine("Station is full")
	End If

	ReturnValue = NorthStation.AddTrain(FourthTrain)
	If ReturnValue = False Then
	   Console.WriteLine("Station is full")
	End If
	Console.WriteLine(SouthStation.GetTrains())
	Console.WriteLine(NorthStation.GetTrains())
	```
	Python
	```
	ReturnValue = SouthStation.AddTrain(FirstTrain)
	if ReturnValue == False:
	    print("Station is full")

	ReturnValue = SouthStation.AddTrain(SecondTrain)
	if ReturnValue == False:
	    print("Station is full")

	ReturnValue = SouthStation.AddTrain(ThirdTrain)
	if ReturnValue == False:
	    print("Station is full")


	ReturnValue = NorthStation.AddTrain(FourthTrain)
	if ReturnValue == False:
	    print("Station is full")

	print(SouthStation.GetTrains())
	print(NorthStation.GetTrains())
	```

Question	Answer	Marks

2(d)(iii)	1 mark each, screenshot(s) showing:
	• **One** output of `"Station is full"`
	• Output of correct data for both stations (in correct format)
	
	e.g.
	
	`Station is full`
	`The trains at station STH are:`
	`12ADV on route number 134`
	`33ART on route number 20`
	
	`The trains at station NTH are:`
	`21VBC on route number 24`
		2

Question	Answer	Marks

3(a)	1 mark each
	• Class header (and end) **and** constructor header (and end) in class
	• Constructor takes two parameters **and** stores each in attributes
	
	Example program code
	
	Java
	`class Record{`
	`      public Integer Key;`
	`      public String Data;`
	
	`      public Record(Integer pKey, String pData){`
	`            Key = pKey;`
	`            Data = pData;`
	`      }`
	`}`
	
	VB.NET
	`Class Record`
	`    Dim Key As Integer`
	`    Dim Data As String`
	`    Sub New(pKey, pData)`
	`        Key = pKey`
	`        Data = pData`
	`    End Sub`
	`End Class`
	
	Python
	`class Record:`
	`    def __init__(self, pKey, pData):`
	`        self.Key = pKey #integer`
	`        self.Data = pData #string`
		2

Question	Answer	Marks

3(b)	1 mark each
	• 2D array of 100 × 10 elements of type `Record`
	• Procedure `InitialiseHashTable()` header (and end) **and** initialises each element in the 2D array to an empty/null record in procedure
	
	Example program code
	
	Java
	`public static Record[][] HashTable = new Record[100][10];`
	`public static void InitialiseHashTable(){`
	`      Record EmptyRecord = new Record(-1,"-1");`
	
	`      for(Integer X = 0; X < 100; X++){`
	`            for(Integer Y = 0; Y < 10; Y++){`
	`                  HashTable[X][Y] = EmptyRecord;`
	`            }`
	`      }`
	`}`
	
	VB.NET
	`Dim HashTable(99, 9) As Record`
	`Sub InitialiseHashTable()`
	`    Dim EmptyRecord As Record = New Record(-1, "")`
	`    For X = 0 To 99`
	`        For Y = 0 To 9`
	`            HashTable(X, Y) = EmptyRecord`
	`        Next`
	`    Next`
	`End Sub`
	
	Python
	`HashTable = []`
	`def InitialiseHashTable():`
	`    global HashTable`
	`    HashTable = [[Record(-1,"")]*10 for i in range(100)]`
		2

Question	Answer	Marks

3(c)	1 mark each
	• Function header (and end) taking one parameter **and** returning calculated hash
	• …. hash calculated correctly from parameter
	
	Example program code
	
	Java
	`public static Integer Hash(Integer TheKey){`
	
	`      return(TheKey % 100);`
	`}`
	
	VB.NET
	`Function Hash(Key)`
	`    Return Key Mod 100`
	`End Function`
	
	Python
	`def Hash(Key):`
	`    return Key % 100`
		2

Question	Answer	Marks

3(d)	1 mark each
	• Procedure header (and end) taking one `Record` parameter
	• Calling `Hash()` using key from parameter **and** storing/using return value
	• Accessing `HashTable[return][0]` **and** storing parameter if no collision … … if collision: iterating through 2nd dimension to find empty index **and** store parameter in that position
	
	Example program code
	
	Java
	`public static void InsertData(Record RecordData){`
	
	`      Integer HashValue = Hash(RecordData.Key);`
	
	`      for(Integer X = 0; X < 10; X++){`
	
	`            if(HashTable[HashValue][X].Key.equals(-1)){`
	`                  HashTable[HashValue][X] = RecordData;`
	`                  X = 10;`
	`            }`
	`      }`
	`}`
	
	VB.NET
	`Function InsertData(RecordData)`
	`   Dim HashValue As Integer = Hash(RecordData.Key)`
	
	`    For X = 0 To 9`
	`        If HashTable(HashValue, X).Key = -1 Then`
	`            HashTable(HashValue, X) = RecordData`
	`            X= 10`
	`        End If`
	`    Next X`
	`End Function`
		4

Question	Answer	Marks

3(d)	Python
	`def InsertData(RecordData):`
	`    global HashTable`
	
	`    HashValue = Hash(RecordData.Key)`
	
	`    for X in range(0, 10):`
	`        if HashTable[HashValue][X].Key == -1:`
	`            HashTable[HashValue][X] = RecordData`	

Question	Answer	Marks

3(e)	1 mark each to max 5
	• Procedure header (and end), opening file **and** closing file (in appropriate place)
	• Iterating through each line in file // reading each line in from file
	• Splitting each line read in by comma …
	• … creating `Record` object with **each** key and data as arguments …
	• … calling `InsertData()` with each **object**
	• Try, catch with appropriate output and all file access within try
	
	Example program code
	
	Java
	`public static void ReadData(){`
	    `String[] Data = new String[3];`
	    `Integer NewKey;`
	    `Integer NewItem1;`
	    `Integer NewItem2;`
	    `Record TheRecord;`
	    `try{`
	        `FileReader File = new FileReader("HashTableData.txt");`
	        `try{`
	            `BufferedReader Reader = new BufferedReader(File);`
	            `String Line= Reader.readLine();`
	            `while (Line != null){`
	                `Line = Line.replace("\n","");`
	                `Data = Line.split(",");`
	                `TheRecord = new Record(Integer.parseInt(Data[0]), Data[1]);`
	
	                `InsertData(TheRecord);`
	                `Line= Reader.readLine();`
	            `}`
	            `Reader.close();`
	        `}catch(IOException ex){}`
	
	    `}catch(FileNotFoundException e){System.out.println("File not found");}`
	`}`
		5

Question	Answer	Marks

3(e)	VB.NET
	`Sub ReadData()`
	`    Dim Line As String`
	`    Dim Data(3) As String`
	`    Dim TheRecord As Record`
	
	`    Dim FileReader As New System.IO.StreamReader("HashTableData.txt")`
	`    While Not FileReader.EndOfStream`
	`        Line = FileReader.ReadLine()`
	`        Data = Split(Line, ",")`
	`        TheRecord = New Record(Integer.Parse(Data(0)), Data(1))`
	`        InsertData(TheRecord)`
	`    End While`
	`    FileReader.Close()`
	
	`End Sub`
	
	Python
	`def ReadData():`
	`    global HashTable`
	`    File = open("HashTableData.txt")`
	`    for Line in File:`
	`        Data = Line.strip()`
	`        Data = Line.split(",")`
	`        InsertData(Record(int(Data[0]), Data[1]))`
	`    File.close()`	

Question	Answer	Marks

3(f)	1 mark each
	• Function header (and end) taking one parameter **and** returning string in all cases
	• Calling `Hash()` with parameter **and** storing/using return value
	• Iterating through 2nd dimension at `HashTable[return value]` **and** comparison to parameter …
	• … returning data if found/equal
	• … returning "`Not found`" if not found by the end of the dimension
	
	Example program code
	
	Java
	`public static String GetRecord(Integer Key){`
	`      Integer HashValue = Hash(Key);`
	`      for(Integer X = 0; X < 10; X++){`
	`            if(HashTable[HashValue][X].Key.equals(Key)){`
	`                  return(HashTable[HashValue][X].Data);`
	`            }`
	`      }`
	`      return "Not found";`
	`}`
	
	VB.NET
	`Function GetRecord(Key)`
	`    Dim HashValue As Integer = Hash(Key)`
	
	`    For X = 0 To 9`
	`        If HashTable(HashValue, X).Key = Key Then`
	`            Return HashTable(HashValue, X).Data`
	`        End If`
	`    Next X`
	`    Return "Not found"`
	`End Function`
		5

Question	Answer	Marks

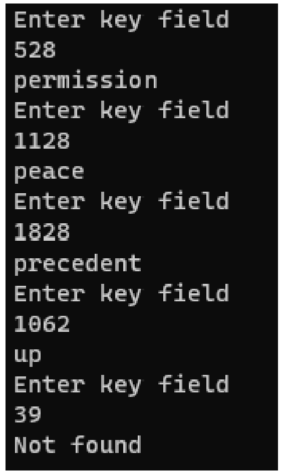
3(f)	Python
	`def GetRecord(Key):`
	`    global HashTable`
	
	`    HashValue = Hash(Key)`
	`    for X in range(0, 10):`
	`        if HashTable[HashValue][X].Key == Key:`
	`            return HashTable[HashValue][X].Data`
	
	`    return "Not found"`	

Question	Answer	Marks

3(g)(i)	1 mark each
	• Calling `InitialiseHashTable()` then `ReadData()`
	• Taking five (integer) inputs
	• Calling `GetRecord()` with **each** input **and** outputting return value
	
	Example program code
	
	Java
	`public static void main(String args[]){`
	`      InitialiseHashTable();`
	`      ReadData();`
	`      Scanner scanner = new Scanner(System.in);`
	`      for(Integer X = 0; X < 5; X++){`
	`            System.out.println("Enter key field");`
	`            System.out.println(GetRecord(Integer.parseInt(scanner.nextLine())));`
	
	`      }`
	`}`
	
	VB.NET
	`Sub Main(args As String())`
	`    InitialiseHashTable()`
	`    ReadData()`
	
	`    For X = 0 To 5`
	`        Console.WriteLine("Enter key field")`
	`        Console.WriteLine(GetRecord(Console.ReadLine()))`
	`    Next`
	`End Sub`
		3

Question	Answer	Marks

3(g)(i)	Python
	`InitialiseHashTable()`
	`ReadData()`
	
	`for x in range(5):`
	`    Key = int(input("Enter key field "))`
	`    print(GetRecord(Key))`	

3(g)(ii)	1 mark each for screenshot(s) showing
	• Input of the 4 integers and matching word output
	`528   permission`
	`1128  peace`
	`1828  precedent`
	`1062  up`
	• Input of `39` and output of `Not found`
	
	e.g.
	
	`Enter key field`
	`528`
	`permission`
	`Enter key field`
	`1128`
	`peace`
	`Enter key field`
	`1828`
	`precedent`
	`Enter key field`
	`1062`
	`up`
	`Enter key field`
	`39`
	`Not found`
		2