Question No. 1

9	To trap (some) **runtime** errors To prevent a program halting unexpectedly To produce meaningful error messages for these errors Example divide by zero // end of file // file not found	**4**

Question No. 2

1(a)	**1 mark** per reason to **max 3** e.g. • Division by zero • Array out of bounds • File does not exist • Stack overflow • Memory leakage • Hardware fault/failure	**3**

1(b)	**1 mark** per bullet point
	
	• Check if file exist
	• Reporting appropriate exception
	
	Python
	`try:`
	`  file = open('MyData.txt')`
	`except:`
	`  print "No file found"`
	
	Visual Basic (.net)
	`Try`
	`  Dim fileReader As New System.IO.StreamReader("MyData.txt")`
	`Catch ex As Exception`
	`  console.writeline("No file found")`
	`End Try`
	
	Pascal
	`try`
	`  Readln("MyData.txt")`
	`except`
	`  Writeln("No file found")`
	`end;`
		2

Question No. 3

<table>
<tr><td>5 (a) (i)</td><td>
<table>
<tr><th>Mark</th><th>Description</th><th>Expected result (Grade)</th></tr>
<tr><td></td><td>Normal</td><td>FAIL/PASS/MERIT/DISTINCTION</td></tr>
<tr><td></td><td>Abnormal</td><td>Error</td></tr>
<tr><td></td><td>Extreme/Boundary</td><td>FAIL/PASS/MERIT/DISTINCTION</td></tr>
</table>
3 × (mark + matching grade)
for abnormal data accept negative values, non-integer values, Expected Result: Error
0 and marks above 100 are still acceptable values
Do not accept FAIL in expected result column for Abnormal data
</td><td>3</td></tr>
<tr><td>(ii)</td><td>(The programmer is) concerned only with the input (i.e. the mark) to the function and monitoring the expected output (i.e. the grade)
// can compare expected result and actual result</td><td>1</td></tr>
<tr><td>(b)</td><td>Exception:
1. situation causing a crash / run-time error / fatal error 1

Exception handling:
2. code which is called when a run-time error occurs 1
3. … to avoid the program terminating/crashing 1</td><td>3</td></tr>
<tr><td>(c)</td><td>1 Open a non-existent file
2 Directory path does not exist
3 Attempt to read past the end of the file // attempt to read an empty file
4 Array subscript is out of range
5 Non-integer value / corrupt data read
6 File already open in a different mode // wrong file permissions</td><td>Max 3</td></tr>
</table>

Question No. 4

1(a)	It is an unplanned event // an event not wanted	**1**
1(b)	**1 mark** per example to **max 3** e.g. • Division by zero • Invalid array index • File does not exist • Run-time error • Invalid input • Invalid argument/value • Stack overflow • Memory leakage • Hardware failure/error	**3**
1(c)	**1 mark** per bullet point to **max 2** • The program will not crash // more robust // program will continue • Result does not cause further errors/problems later • Appropriate error messages/result • Exceptional conditions are identified • Improve readability	**2**

Question No. 5

6(b)	1 mark per completed statement
	
	PROCEDURE StoreRecord(NewData : **CustomerData**)
	  HashValue ← CustomerHash(NewData.CustomerID)
	  Filename ← "CustomerRecords.dat"
	  OPENFILE Filename FOR **RANDOM**
	  SEEK Filename, **HashValue**
	  PUTRECORD Filename, **NewData**
	  CLOSE Filename
	ENDPROCEDURE
		5

Question No. 6

7(a)	1 mark per bullet point to max 2 • To stop the program crashing … • To stop a run-time error … • … to make sure the input is the correct data type // other reasonable example	**2**

7(b)	1 mark per bullet point
	• Using try (and close where appropriate) followed by the input
	• Catching exception
	• Outputting appropriate message (built-in or otherwise)
	
	Example program code:
	
	VB.NET
	`Try`
	`  Dim Value As Integer`
	`  Console.WriteLine("Enter a number")`
	`  Value = Console.ReadLine()`
	`Catch ex As Exception`
	`  Console.WriteLine(ex.Message)`
	`End Try`
	
	Python
	`try:`
	`    Value = int(input("Enter a number"))`
	`except:`
	`    print("Invalid number")`
	
	Pascal:
	`begin`
	`try`
	`  readln(Value);`
	`except`
	`  On E : Exception do writeln("Invalid number");`
	`end;`
		3

7(c)	1 mark per example • Check file exists • No input • No data in file • Array out of bounds • Calculation / division by 0	**2**

Question No. 7

2(a)	1 mark per bullet point to max 4:
	
	• declaration of type Book
	• Title, Author and ISBN as String
	• Fiction as Boolean
	• LastRead as Date
	
	For example:
	
	`TYPE Book`
	`  DECLARE Title : String`
	`  DECLARE Author : String`
	`  DECLARE ISBN : String`
	`  DECLARE Fiction : Boolean`
	`  DECLARE LastRead : Date`
	`ENDTYPE`
		4

2(b)	1 mark per bullet point to max 4:
	
	• **Function** header
	• … taking ISBN as parameter
	• Converting ISBN to integer
	• Calculating Hash (ISBN mod 2000 + 1)
	• Returning the calculated Hash
	
	Examples:
	
	Python:
	
	`def Hash(ISBN):`
	`  ISBNint = int(ISBN)`
	`  Hash = (ISBNint % 2000) + 1`
	
	VB.NET:
	
	`Function Hash (ISBN As String) As Integer`
	`  ISBNint = convert.toInt32(ISBN)`
	`  Hash = (ISBNint MOD 2000) + 1`
	`End Function`
	
	Pascal:
	
	`function Hash(ISBN : String) : Integer`
	`  begin`
	`  ISBNint = StrToInt(ISBN)`
	`  Hash = (ISBNint MOD 2000) + 1`
	`  end;`
		4

2(c)	1 mark per bullet point to max 8: • Procedure FindBook declaration **and** prompt **and** input ISBN • Validate data input has 13 characters • … and are all numeric • ..loop until valid • Call Hash() with input data **and** store return data • Open MyBooks.dat for reading as random file **and** close • Finding the record using return value Hash() • Get the data for the record • …store in variable of type Book • …output all the data for the record	8

Question No. 8

(a)

MembershipFile

Address	MemberID	other member data
0	0	
1	1001	
2	7002	
3	0	
4	0	
5	3005	
6	0	
7	0	
8	0	
:	:	
:	:	
96	4096	
97	0	
98	2098	
99	0	

1001 and 7002 and 3005 1
4096 and 2098 1 [2]

(b) (i)

```
10 // generate record address
20 NewAddress ← Hash(NewMember.MemberID)
30 // move pointer to the disk address for the record
40 SEEK NewAddress
50 PUTRECORD "MembershipFile", NewMember
```

[4]

(ii)

```
01 TRY
02    OPENFILE "MembershipFile" FOR RANDOM
03 EXCEPT
04    OUTPUT "File does not exist"
05 ENDTRY
```

[2]

(iii) collisions/synonyms
The previous record will be overwritten [2]

(iv) Create an overflow area
The 'home' record has a pointer to others with the same key
OR
Store the overflow record at the next available address
in sequence
OR
Re-design the hash function ….
to generate a wider range of indexes // to create fewer collisions [2]

(v)

```
41 GETRECORD "MembershipFile", CurrentRecord
42 WHILE CurrentRecord.MemberID <> 0
43    NewAddress ← NewAdress + 1
44    IF NewAddress > 99 THEN NewAddress ← 0
45    SEEK NewAddress
46    GETRECORD "MembershipFile", CurrentRecord
47 ENDWHILE
```

[max. 4]

Question No. 9

(a) 1 mark for structure header/ending
1 mark for each field correct, take away 1 mark for additional fields
Python answers will use a class

Python

```
class StockItem :
    def __init__(self) :
        self.ProductCode = ""     # = 0
        self.Price = 0.0          # = 0
        self.NumberInStock = 0
```

(b) **(i)**

```
01 TRY
02    OPENFILE "StockFile" FOR READ/RANDOM   // ignore "
03 EXCEPT
04    OUTPUT "File does not exist"
05 ENDTRY
```

[2]

(ii) (Line 01) alerts system to check for possible run-time errors (exception)
(Lines 03, 04) handle the exception without the program crashing // keeps program running// provide alternative statements to execute to avoid run-time error

Accept "exception handling" for 1 mark [Max 2]

(c)

```
WHILE NOT EOF("StockFile")
   READFILE "StockFile", ThisStockItem // accept reading separate
fields
   OUTPUT ThisStockItem.ProductCode
   OUTPUT ThisStockItem.NumberInStock
ENDWHILE
```

1 mark for loop (accept REPEAT)
1 mark for EOF("StockFile") // StockFile.Peek <> -1 / NONE/"
1 mark for READ record
1 mark for OUTPUT of 2 fields

Ignore opening and closing file [4]