higher education & training

Department:
Higher Education and Training
REPUBLIC OF SOUTH AFRICA

MARKING GUIDELINE

NATIONAL CERTIFICATE (VOCATIONAL)

PRINCIPLES OF COMPUTER PROGRAMMING

(Second paper)

NQF LEVEL 3

22 November 2024

This marking guideline consists of 10 pages.

Copyright reserved

INSTRUCTIONS TO MARKERS

1. Mark all the questions, using red ink only.

2. Read all the answers provided in the source code carefully.

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

4. Put your ticks neatly and legibly.

5. Write down the question number next to the source code being marked.

6. Each page of the printed source code must be initialled by the marker.

7. Markers are required to work out the question paper before the marking guideline is discussed.

8. This marking guideline is to be used as the basis for the marking session. All markers are required to attend a rigorous standardisation meeting to ensure that the guideline is interpreted consistently applied in the marking of candidates' work.

9. Note that candidates who provide an alternate correct solution to that given as an example of a solution in the marking guideline will be given full credit for the relevant solution, unless the specific instructions in the paper were not followed or the requirements of the question were not met.

10 Marks are awarded when candidates clearly demonstrate what they know and can do.

11. Marks are not deducted for errors and for omissions.

12. The candidate's total mark is written on the EXAMINATION FOLDER cover next to the corresponding question number.

13. If a candidate handed in a PRINTOUT, but achieved 0 for the question, then 0 (zero) is written on the cover next to the corresponding question number.

14. If a candidate did not hand in a PRINTOUT, a line is drawn through the block next to the corresponding question number.

15. If a candidate handed in a PRINTOUT without a printed EXAMINATION NUMBER and a consistent WATERMARK, the question must be marked and reported as an irregularity.

16. All irregularities are marked, and marks are transferred to the cover page. The word 'Irregularity' is written next to the percentage and an irregularity report is completed.

17. If a candidate handed in duplicate printouts, a line must be drawn across the duplicate with a notation 'Duplicate' written across it.

Copyright reserved

QUESTION 1: MARKING GRID - GENERAL PROGRAMMING

QUESTION	NUMBER	DESCRIPTION	MARK	MAX MARKS

1.1		Change font size.
		Change text property of text box.
		Change read-only property to True.
		Change back colour of text box to Yellow.
		
		`Private Sub btnQ1_1_Click(…) Handles btnQ1_1.Click`
		`  txtQ1_1.Font = New Font("Consolas", 12)`
		`  txtQ1_1.Text = "Hello Mzansi"`
		`  txtQ1_1.ReadOnly = True`
		`  txtQ1_1.BackColor = Color.Yellow`
		`End Sub`
			✓
			✓
			✓
			✓
				(4)

1.2	1.2.1	cboQuestion1_2_DropDown() –
		Correct dropdown event procedure.
		
		Clear combo box.
		
		Populate combo box with all twelve (12) months.
		
		`Private Sub cboQuestion1_2_DropDown(…) Handles`
		
		`cboQuestion1_2.DropDown`
		
		`  cboQuestion1_2.Items.Clear()`
		
		`  cboQuestion1_2.Items.Add("January")`
		`  cboQuestion1_2.Items.Add("February")`
		`  cboQuestion1_2.Items.Add("March")`
		`  cboQuestion1_2.Items.Add("April")`
		`  cboQuestion1_2.Items.Add("May")`
		`  cboQuestion1_2.Items.Add("June")`
		`  cboQuestion1_2.Items.Add("July")`
		`  cboQuestion1_2.Items.Add("August")`
		`  cboQuestion1_2.Items.Add("September")`
		`  cboQuestion1_2.Items.Add("October")`
		`  cboQuestion1_2.Items.Add("November")`
		`  cboQuestion1_2.Items.Add("December")`
		
		NOTE TO MARKERS:
		If candidate placed code in *SelectedIndex* event procedure, mark question out of two marks.
			✓
			
			✓
			
			✓
				(3)

	1.2.2	Enable button control. `  btnQuestion1_2.Enabled = True`	✓	(1)

<table>
<tr><th>QUESTION</th><th>NUMBER</th><th>DESCRIPTION</th><th>MARK</th><th>MAX MARKS</th></tr>
<tr><td></td><td>1.2.3</td><td>Determine if the combo box control has been selected.
Remove the month that has just been selected from the combo box control.

<pre>Private Sub btnQuestion1_2_Click(…) Handles
                          btnQuestion1_2.Click
   If cboQuestion1_2.SelectedIndex <> -1 Then
   cboQuestion1_2.Items.RemoveAt
              (cboQuestion1_2.SelectedIndex)
End If</pre></td><td>✓✓

✓✓</td><td>(4)</td></tr>
<tr><td>1.3</td><td colspan="2">Clear list box.
Assign symbol from combo box to string variable.
Assign number of rows from text box to integer variable.
Use of loop to create the pattern row and then display this row in the list box.

<pre>Private Sub btnQuestion1_3_Click(…) Handles
btnQuestion1_3.Click
 ' Clear the ListBox
   lstQuestion1_3.Items.Clear()

 ' Read the selected symbol from a ComboBox
   Dim strSelectedSymbol As String =

cboQuestion1_3.SelectedItem.ToString()
 'Determine the rows
   Dim intRows As Integer = CInt(txtQuestion1_3.Text)

 ' Loop through the rows
   Dim strRowPattern As String = ""
   For i As Integer = 1 To intRows

   ' Format pattern for the current row
    strRowPattern &= New String(strSelectedSymbol(0), i)
    OR strRowPattern = strRowPattern & strSelectedSymbol

   ' Add the row pattern to the ListBox
    lstQuestion1_3.Items.Add(strRowPattern)
   Next
End Sub</pre></td><td>✓
✓
✓
✓✓✓✓</td><td>(7)</td></tr>
<tr><td>1.4</td><td>1.4.1</td><td>Check if any item has been selected in the source list box.
Add the selected item from source list box to destination list box.
Remove the selected item from source list box ONLY.</td><td>✓

✓

✓</td><td>(3)</td></tr>
</table>

Copyright reserved

QUESTION	NUMBER	DESCRIPTION	MARK	MAX MARKS
		Private Sub btnQuestion1_4_1_Click(…) Handles btnQuestion1_4_1.Click ' Check if any item is selected in ListBox1 If lstQuestion1_4_a.SelectedItem IsNot Nothing Then ' Add the selected item to ListBox2 lstQuestion1_4_b.Items.Add (lstQuestion1_4_a.SelectedItem) ' Remove the selected item from ListBox1 lstQuestion1_4_a.Items.Remove (lstQuestion1_4_a.SelectedItem) End If End Sub		
	1.4.2	Loop through all the items in the source list box. Move item from source list box to destination list box. Remove the item. Private Sub btnQuestion1_4_2_Click(…) Handles btnQuestion1_4_2.Click ' Loop through all items in ListBox1 For i As Integer = lstQuestion1_4_a.Items.Count - 1 To 0 ' Add all items to ListBox2 lstQuestion1_4_b.Items.Add (lstQuestion1_4_a.Items(i)) ' Remove all items from ListBox1 lstQuestion1_4_a.Items.RemoveAt(i) Next End Sub	✓ ✓ ✓	(3)
1.5	ONE mark if ALL comment lines.		✓	(1)
				[26]

QUESTION 2: MODULAR PROGRAMMING (ARRAYS & PROCEDURES)

QUESTION	NUMBER	DESCRIPTION	MARK	MAX MARKS
2.1	2.1.1	Created and initialized two (2) constants. Const decWeightCharge As Decimal = 2.8D Const decTransportCharge As Decimal = 1.1D	✓	(1)
	2.1.2	Created three (3) parallel, single dimension arrays. Dim intArrBoxNumber() As Integer Dim dblArrBoxWeight() As Double Dim decArrAmount() As Decimal	✓	(1)
	2.1.3	Created integer counter variable. Dim intCount As Integer = 0	✓	(1)

2.2		Function procedure (DetermineCapacity)
		Function definition header.
		Function return type. (Integer)
		If/Elseif logic
		Return value.
		
		Private Function DetermineCapacity() As Integer
		Dim intCapacity As Integer
		
		If rad500Kg.Checked Then
		intCapacity = 500
		ElseIf rad750Kg.Checked Then
		intCapacity = 750
		ElseIf rad1000Kg.Checked Then
		intCapacity = 1000
		ElseIf rad1250Kg.Checked Then
		intCapacity = 1250
		Else
		intCapacity = 1500
		End If
		Return intCapacity
		End Function
			
			✓
			✓
			✓✓
			✓
				(5)

2.3		Function procedure (CalcWeightCost). Function definition header. Correct Parameter. Function return type (Decimal). Return value. Private Function CalcWeightCost(ByVal Weight As Double) As Decimal Return CDec(Weight) * decWeightCharge End Function	 ✓ ✓ ✓ ✓✓	(5)

QUESTION	NUMBER	DESCRIPTION	MARK	MAX MARKS

2.4		Function procedure (CalcDistanceCost)
		Function definition header.
		Correct Parameter.
		Function return type (Decimal).
		Return value.
		
		`Private Function CalcDistanceCost(ByVal Distance As Integer) As Decimal`
		`Return CDec(Distance) * decTransportCharge`
		`End Function`
			
			✓
			✓
			✓
			✓✓
				(5)

2.5	2.5.1	Button – Question 2_5
		
		Resized/preserved the THREE arrays.
		`ReDim Preserve intArrBoxNumber(intCount)`
		`ReDim Preserve dblArrBoxWeight(intCount)`
		`ReDim Preserve decArrAmount(intCount)`
			
			✓✓✓
				(3)

	2.5.2	Determine if capacity of van is exceeded.
		Create pop-up dialog box with appropriate message.
		Clear text box contents.
		Call sub-procedure.
		Terminate procedure.
		
		`If ((CInt(txtWeight.Text) + intVanCapacity) > intMaxVanCapacity) Then`
		`MessageBox.Show("Not enough space for this parcel" & vbCrLf & vbCrLf & "There is only space enough for " & intMaxVanCapacity – intVanCapacity & " kgs", "You can try a smaller parcel", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)`
		`txtWeight.Clear`
		`Call ConfirmAdd()`
		`Exit Sub`
		`End If`
			✓
			
			✓
			✓
			✓
			✓
				(5)

	2.5.3	Populate Number and Weight arrays.
		
		`intArrBoxNumber(intCount) = intCount`
		`dblArrBoxWeight(intCount) = CDbl(txtWeight.Text)`
			✓✓	(2)

	2.5.4	Populate Cost Array from Function. `decArrAmount(intCount) = CalcWeightCost(CDbl(txtWeight.Text))`	✓✓	(2)
2.6	ONE mark if ALL comment lines.		✓	(1)
				[31]

QUESTION 3: FILE AND ERROR HANDLING

QUESTION	NUMBER	DESCRIPTION	MARK	MAX MARKS
3.1	Sub-procedure (ReadFile) Try/Catch block with Application termination {both}. Determine if the file Students.csv exists. Display Message dialog box. Try Using fileStream As New FileStream("Students.txt", FileMode.Open, FileAccess.Read) End Using Catch ex As FileNotFoundException MessageBox.Show("The file does not exist." & vbCrLf & vbCrLf & "This application will now terminate!", "File not found", MessageBoxButtons.OK, MessageBoxIcon.Warning) Application.Exit() 'OR Me.Close() End Try **NOTE:** Do NOT penalise if candidate used File.Exists() method.		✓ ✓ ✓	(3)
3.2	Exception Handler Try Write to file using StreamWriter Using writer As New StreamWriter("Students.txt", True) writer.Write(txtName.Text & "," & cboCourse.SelectedItem.ToString & "#" & txtClass.Text & vbCrLf) Confirm write with Messagebox MessageBox.Show("File saved successfully")		✓ ✓✓ ✓✓✓ ✓	(7)
3.3	ONE mark if ALL comment lines.		✓	**(1)**
				[11]

QUESTION 4: DATABASE

4.1 Renaming the database. No mark allocation (0)

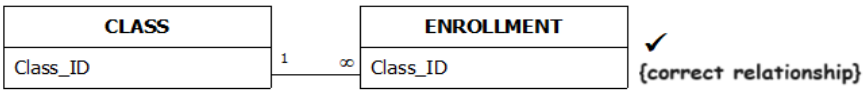
4.2

Columns

Name	Type	Size
Stu_ID	Short Text	13
Class_ID ✓	Long Integer ✓	4 ✓
Enrol_Date	Date With Time	8
Enrol_Status	Short Text	25

Relationships

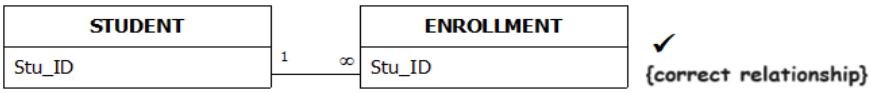
CLASSENROLLMENT



✓ {correct relationship}

Attributes: Enforced ✓
RelationshipType: One-To-Many ✓

STUDENTENROLLMENT



✓ {correct relationship}

Attributes: Enforced ✓
RelationshipType: One-To-Many ✓

Print the database table structures for ENROLLMENT. (9)

4.3

ENROLLMENT			
Stu_ID	**Class_ID**	**Enrol_Date**	**Enrol_Status**
0001105010089	1	2023/11/01	Enrolled
0002250145086	3	2023/11/02	Waiting List
0012251235081	3	2023/11/02	Waiting List
0012255221087	6	2023/10/18	Enrolled
0102141245084	2	2023/11/06	Enrolled
0102141245084	5	2023/11/21	Waiting List
0111253215081	4	2023/11/13	Waiting List

✓✓✓✓ (4)

4.4

Field:	Stu_ID	Name: Stu_Name	Surname: Stu_Surname	Class: Class_Name
Table:	ENROLLMENT	STUDENT	STUDENT	CLASS
Sort:			Ascending	
Show:	☐	☑	☑	☑
Criteria:				
or:				

Hide Stu_Id field ✓
All fields selected ✓
Sorted Surname {Ascending} ✓✓
Rename fields ✓
Printout ✓ (6)

4.5

Field:	Student ID: Stu_ID	Name: Stu_Name	Surname: Stu_Surname	Class: Class_Name	Status: Enrol_Status
Table:	ENROLLMENT	STUDENT	STUDENT	CLASS	ENROLLMENT
Sort:				Descending	
Show:	☑	☑	☑	☑	☑
Criteria:					"Waiting List"
or:					

All fields selected ✓
Criteria {"Waiting List"} ✓✓
Sorted Surname {Descending} ✓
Rename fields ✓
Printout ✓ (6)

4.6

ENROLLMENT - EXAMINATION NUMBER

Class	Student ID	Name	Surname	Status
Artificial Intelligence				
	0111253215081	Patsy	Walker	Waiting List
Database Systems				
	0012251235081	Harvey	Cobblepot	Waiting List
	0002250145086	David	Lee	Waiting List
Web Development				
	0102141245084	Peter	Pakade	Waiting List

All fields selected ✓
All records ✓
Examination number added to label ✓✓
Grouped by Class ✓✓
Printout ✓ (7)

[32]

TOTAL: 100

Copyright reserved