Place candidate's barcoded sticker here.

This sheet must be stapled to the front of each candidate's submission.

I.T. PRACTICAL EXAMINATION 2016

Version:	**FINAL Marksheet (October 2016)**	**Prog. Lang.**	
Candidate:			
Marker:		**120**	**0**
Checksum:	0000-0000-0120	**Total**	**Pupil**

	SQL		
1.1	SELECT * FROM tblItems WHERE Retailer = “Home Pride”	**3**	
1.2	SELECT Description, Price FROM tblItems WHERE Price >= 350 AND Price <= 500 OR: WHERE Price BETWEEN 350 AND 500	**4**	
1.3	SELECT BrideName & " & " & GroomName AS CoupleName, MONTH(WeddingDate) AS MonthOfWedding FROM tblCouples ORDER BY MONTH(WeddingDate)	**4**	
1.4	UPDATE tblGifts SET Available = True WHERE RequestedQuantity > BoughtQuantity; (Alternatively: UPDATE tblGifts SET Available = RequestedQuantity > BoughtQuantity)	**4**	
1.5	SELECT Retailer, AVG(Price) AS AvgItemPrice FROM tblItems WHERE Description LIKE '*Platter*' (%Platter% will not work in Access. Do not award marks) GROUP BY Retailer	**5**	
1.6	SELECT SUM(Price) AS TotalPrice FROM tblItems WHERE ItemID NOT IN (SELECT ItemID FROM tblGifts) OR SELECT SUM (Price) AS TotalPrice FROM tblItems LEFT JOIN tblGifts ON tblItems.ItemId = tblGift.ItemID WHERE tblGifts.itemID IS NULL (If the join is incorrect award no marks. If the ON clause is correct then award 1 mark).	**5**	
1.7	SELECT BrideName, Description, RequestedQuantity - BoughtQuantity AS QuantityAvailable FROM tblItems, tblGifts, tblCouples WHERE tblItems.ItemID = tblGifts.ItemID AND tblCouples.CoupleID = tblGifts.CoupleID AND Available = True ORDER BY BrideName , RequestedQuantity - BoughtQuantity (Accept WHERE … RequestedQuantity > BoughtQuantity instead of Available = True or any other working condition)	**9**	
1.8	INSERT INTO tblGifts (ItemID, CoupleID, RequestedQuantity, BoughtQuantity, Available) SELECT ItemID, 9, RequestedQuantiy, 0 , True (Hard-coded values) (Field names in select) FROM tblGifts WHERE CoupleID = 2 (Only award for correct nested SELECT)	**6**	
	Programming		
2.1	**Wedding Class:** Class header is correct	**1**	
2.2	**Fields/Properties:** attributes declared private name correctly and correct type	**3**	
2.3	**Constructor:** header is correct with correct parameters and values assigned to correct parameters	**3**	
2.4	**Accessor Methods (getWeddingDate, getVenue, getGuests):** named correctly return type is correct and returns the correct attribute value	**3**	
2.5	**equals:** method header correct (accept integer for inVenue paramenter) weddingDate attribute compared with parameter and venue attribute compared with parameter (accept == for comparison of inVenue) and true or false returned (Accept any attempt to convert between integer and string with …)	**4**	
2.6	**toString**: method header correct correct attributes returned with the correct formatting (& need not have spaces before and after)	**3**	

IEB Copyright © 2016

3.1	**BookingDay Class:** Class header is correct	1	
3.2	**Attributes:** bookingDate declared correctly garden and lake BOTH attributes declared as Wedding objects	3	
3.3	**Constructor:** inBDate parameter is of the correct type garden and lake attributes are Wedding objects inBDate assigned to correct attribute and garden and lake assigned to correct attributes (Penalise for incorrect garden and lake types again)	4	
3.4	**getOverBooked:** method header correct perform checks in two separate if statements (do not accept if …else) that garden or lake properties are not null check the number of guests does not exceed 160 and 130 respectively; if so, return a string stating overbooked in correct format	5	
3.5	**toString:** method header correct check if there is a garden and lake wedding (deduct this markeven if already deducted in 3.4) and return toString of respective object(s). Check if neither venue is booked and return appropriate message. Return overbooking information	5	
4.1	**VenueManager:** Class header is correct	1	
4.2	**PROPERTIES:** Private (penalise again for non-private properties) array of wedding objects and array of BookingDay objects declared correctly and both of correct size (50 and 30 respectively)	4	
4.3	**Constructor:** Accepts filename as a parameter Display error if file does not exist Open file for reading Loop through every line in the file (while or for loop accepted) Read a line for the wedding information and tokenize. Add a new Wedding object to the array at the correct position called constructor with correct parameters	10	
4.4	**listAllWeddings:** method header correct, appropriate for loop, concatenate the toString of each object with a newline	5	
4.5	**getWeddingsOnDay:** Method header (including parameters) correct return null if no wedding is found loop through the array of objects use the wedding objects' equals method (or any other method) to compare objects return the wedding object if found	5	
4.6	**processBookings:** Loop through the array of BookingDay objects Generate each day of the month Allowance made for days with a single digit Search for lake weddings on that day Search garden weddings on that day Create a new BookingDay object at the correct position using the date and the two wedding objects as parameters Return the all BookingDay objects toString's with each on a new line	8	
5.1	Create an interface/form/class/unit called **WeddingUI**	1	
5.2	Instantiate a **VenueManager Object** (Must have filename parameter, -1 if not penalised in 4.3)	1	
5.3	Methods called and output displayed	2	
6.1	**displayCalendar:** Method header is correct Days of the week as column headings Leading spaces before the 1st generated using a loop Displays all 30 days in the month generated using a loop Days without leading zeros are displays with the correct spacing using an if statement or tabs New line after each Sunday checked using some kind of calculation within the loop (Give maximum 2 marks for a purely hard-coded solution with 7 if statements and no loops or other if statements) (Accept any arrays or data structures that are populated using loops and if statements and then just displayed. E.g. 2D Arrays)	7	
6.2	Method called in interface and output displayed	1	

IEB Copyright © 2016