**Lab Questions (9 @5 points each for a total of 45)**Your answers for each question should fill at least 1/2 page. You may use any drawing application to create the designs and then copy/paste the pictures into this document. If you use Visio, just use the flowchart shapes; you may be able to use the Word flowchart auto shapes; you might need to scan your drawing to a .jpg or .bmp or whatever and then paste it here. **---------------------------------------------------------------------  
 A grocery chain needs a database designed in order to keep track of inventory and sales; the scenario to model is as follows:  
 A customer goes to one of the stores to make a purchase, which contains one or more products. The employee ID of the cashier will be stored for the purchase; if the customer is a member of the loyalty program, his membership ID will be associated with the purchase as well (although this is not required).  
 Each product will have a current price and stock inventory count for that store; the purchase amount will be the total of all product prices times their quantity. After the customer pays, the purchase should be marked as paid, and the stock inventory count for all items in the purchase should be reduced accordingly.   
---------------------------------------------------------------------**

**Q1**. Read through the above scenario and determine which entities are involved (identify all of the *things* and decide whether they are entity classes or attributes). Create a simpleERD to show only the **entities** and the **relationships** between them (see the first lecture example). Name the relationships (e.g. “employs”, “schedules”, etc) but do not show the relationship cardinalities nor any of the attributes. It is expected and perfectly OK to have at least one **many-to-many** and/or **ternary** relationship here, because these will be changed in the next step.



**Q2.** Expand the ERD from the previous question by adding the minimum and maximum relationship cardinalities to the model; eliminate any ternary and/or many-to-many relationships from the previous ERD by creating an intersection entity between them. Show optional/mandatory cardinalities and use crows-feet notation to display the “many” sides. Include cardinality for **all** relationships between two entities.   
**A2.**



**Q3.** Taking each pair of related entities at a time, write one sentence describing the relationship cardinalities (minimum and maximum) in each direction. For example, this relationship



...may be stated: *An employee builds zero to many stoves / A stove is built by exactly one employee.*

**A3. A CUSTOMER can have 0 to many PURCHASE(s) / A PURCHASE has exactly one CUSTOMER**

**A CUSTOMER can be a LOYALTY\_PROGRAM member / A LOYALTY\_PROGRAM member must be a CUSTOMER**

**An EMPLOYEE can have 0 to many PURCHASES / A PURCHASE has exactly one EMPLOYEE**

**A PURCHASE has 1 to many PURCHASE\_LINE\_ITEMS / A PURCHASE\_LINE\_ITEM has exactly one PURCHASE**

**A STORE has 1 to many EMPLOYEES / An EMPLOYEE has exactly one STORE**

**A STORE has 1 to many PRODUCTS / A PRODUCT has exactly one STORE**

**A PRODUCT can be 0 to many PURCHASE\_LINE\_ITEMs / a PURCHASE\_LINE\_ITEM has exactly one PRODUCT**

**Q4.** Turn four of your entities into relations. Select an attribute(s) to represent the primary key; display this first, underlined. Include other attributes you would expect to find for this entity. Lastly, include any foreign keys which reference other entities; display these in italics.

Example from the FiredUp database:

**STOVE** (SerialNumber, Type, Version, DateOfManufacture, Color, *FK\_EmpID*)

**A4.**

**CUSTOMER (CustID, Name)**

**PURCHASE (PurchaseID, Date, *CustID, EmpID*)**

**EMPLOYEE (EmpID, Name, Position, Salary, *StoreID*)**

**STORE (StoreID, Location)**

**Q5.** What are some questions you may have to ask the user in order for you to continue the design process? Include at least **five** questions. Be sure to address any ambiguities in the scenario that might affect your design.

**A5. The first thing I would ask would be if we were needing to keep track of employees as an entity. If employees (cashier's as described in the text) do not need to be kept track of in this system--besides just being kept track of in the PURCHASE table--then they do not need to be included as an entity (but it is probably a good idea, which is why I included it as an entity in my diagram).  
 Another aspect I would want clarified would be whether or not loyalty\_program needs to be an entity. I could just have an attribute in the PURCHASE table that holds a boolean, bit, or some simple value to determine if a purchase was made from a loyalty\_program customer. Also, that field will need to be in the customer table, to determine whether or not that customer is involved in the loyalty program.**

**Q6.** A weak entity is an entity whose existence depends upon another entity. Examine the ERD for the FiredUp database and identify two entities that can be considered to be weak (note: this ERD does *not* use rounded-corner representation of weak entities; you will have to identify them logically). Explain why each of these entities is weak.

**A6. PHONE and EMAIL are the weakest entities that I can spot. They rely on the primary key of the CUSTOMER table as a foreign key. I can tell that they are weak because they rely solely on the existence of a CUSTOMER entity. If a customer was deleted then some values of the phone and email table may be left as orphans, children entities without a parent entity. I believe this is what determines a weak entity--one that can't exist without relying on the existence of another.**

**Q7.** Again, examine the FiredUp ERD. Identify two intersection entities (i.e. entities which were created to break apart a **N:M** relationship). Explain what tells you that these are intersection entities.

**A7. You can tell they are intersection entities because they are on the many side of several 1:N relationships. The first one that stands out is PO\_LINE\_ITEM, for it is the table that is an intersection relationship for the PART and PURCHASE\_ORDER tables. There are many purchase orders involving many parts, which constitutes a N:M relationship. PO\_LINE\_ITEM is used to bridge the gap to help make the database a relational database. I believe this intersection relationship is binary for it involves joining 2 entities in accordance with the relational model. INVOICE\_LINE\_ITEM is another one of these intersection entities, used to join the INVOICE, PART, and STOVE tables (because many invoices involve many parts which involve many stoves).**

****

**Q8.** Examine the data in the relations in Figure 3-17, above. Explain why we can say that Buyer determines Department, but Department does not determine Buyer.

**A8. A Buyer determines the Department because Buyer seems as though it is used as a unique identifier for department. Department does not determine the Buyer however, because there are multiple Buyer values to a single Department value (i.e. Nancy Meyers and Pete Harrison both order from the Water Sports department--so each one of them determines the Water Sports department but not the other way around).**

**Q9.** Explain why SKU -> (SKU\_Description, Department, Buyer).

**A9. In the last question it was determined that the Buyer field is used to uniquely identify the Department value. If we are to look at the big table at the bottom of figure 3-17, one can see that SKU is a unique identifier that determines the SKU\_Description and the Buyer. Transitively if SKU determines Buyer, and Buyer determines Department, the SKU determines Department as well.**

**IF**

**SKU -> (SKU\_Description, Buyer)**

**AND**

**BUYER -> (Department)**

**THEREFORE TRANSITIVELY:**

**SKU -> (SKU\_Description, Department, Buyer)**