Debugging
I will provide as a download an SQL “create database” script and a VS2008 project at the beginning of the exam. You are to run the SQL script and copy the project files to your local VS2008 directory. Start the project. Locate and fix the error. Write a comment in the code indicating where the error was and how you fixed it.

Programming
For the project and database I provided at the beginning of the exam, be prepared to write code for functionality I specify. For example, I might ask you to add a page that includes a gridview with sorting. You may use any resources you wish during the exam except for each other (or a live person--no life lines please). If you think you will listen to some of my podcasts, please bring earbuds so you do not disturb others.

My goal is for programming portion to be able to be completed in 30-45 minutes. You have the entire hour and 15 minutes. But my goal is for the programming portion to be easy if you thoughtfully completed all three exercises.

Written portion (50%) March 10, 6:00 FCE 17

Be able to explain the SQL syntax from the course project. Note: explain does not mean restate in English. I can read as well as you can… For example,

Sample Question: A colleague calls you over to his/her workstation, points to the SQL syntax given below and asks “Hey, could you help me for a minute? I am trying to use the sproc you wrote and I’m not sure that I understand how it works or how to use it. Thanks.”

```
CREATE PROC InsertRoleInfo
  @RoleName NVARCHAR(50),
  @RoleDescription NVARCHAR(250)
AS
BEGIN TRY
  BEGIN TRANSACTION
  INSERT INTO tblRole (RoleName, RoleDescription) VALUES
      (@RoleName, @RoleDescription)
  COMMIT TRANSACTION
  PRINT 'Role successfully inserted...'
END TRY
BEGIN CATCH
  DECLARE @ErrorMessage VARCHAR(500)
  SET @ErrorMessage = ERROR_MESSAGE() + ' Rolledback transaction: insert failed.'
  ROLLBACK TRANSACTION
  RAISERROR (@ErrorMessage, 16,1)
END CATCH
END
```
**Bad answer:**
“The statement CREATE PROC creates a stored procedure named InsertRoleInfo. @RoleName is data type NVARCHAR(50) and @RoleDescription is data type NVARCHAR(250). There is a BEGIN and an END for the sproc. There is a BEGIN TRY and BEGIN CATCH for the INSERT statement. The INSERT statement inserts RoleName and RoleDescription into tblRole. The sproc prints the message “Role successfully inserted” and does COMMIT TRANSACTION or else it sets an error message, does a ROLLBACK, and uses RAISERROR (@ErrorMessage, 16, 1).”

[Don’t laugh… I have and will get answers like this.]

**Good answer:**
“Sure, let me explain how it works. When you invoke this stored procedure, you have to use the name following the CREATE PROC statement. This is more-or-less the sproc’s signature. What follows the CREATE statement is the list of input parameters. Sprocs can also have output parameters. You can tell because the key word OUTPUT will follow the data type declaration. The way this sproc is written, both parameters are required (There is not ^=NULL following the parameter declaration). NVARCHAR is how SQL Server defines string data. Both attributes should be cast as string data in your application when you call it. You should also note that the number in the parenthesis indicates the maximum string length. One more thing, in SQL Server, the @ sign means the parameter is used as a local variable in the sproc. Global or system variables are prefixed with @@. I have written this sproc with exception handling using a try catch block (I think you are familiar with a try catch block?). If an exception is encountered when the new row is inserted into tblRole (this sproc creates a new row in tblRole and inserts the values passed in the input parameters) I capture the error message thrown by SQL Server “ERROR_MESSAGE()” in the local variable @ErrorMessage so that I can pass it along to the calling method using the RAISERROR function. If you are using the SqlException class in .Net, you can capture this error with the try catch block in your code. The ROLLBACK statement reverses the insert operation and returns tblRole back to its previous state. It is as if the insert never happened. Otherwise, the insert is committed to the database, that is, the change become final. The PRINT statement is just for debugging. As you can see below in the script, I have some test cases where you can run the sproc and verify its behavior.”

[Note, if you have not had ISC 561, I would not necessarily expect you to know this about ROLLBACK and COMMIT but you should from your undergraduate class…]

Be able to explain C# code from our course project. For example,

*Sample question:* You are at a code review session with your development team. You are the lead person on data access tier development. You are currently reviewing the C# code listed below. One of your teammates raises his hand and says “you know, I’ve never seen variables defined that way, “@Email” and so on. What do they refer to? In fact, I looked at this code this morning and wasn’t sure I understood at all how this is working…” As the leader of the data access development you respond…
public void ValidatePerson(string Email, string Password, string ConnectionString)
{
    SqlParameter[] arParms = new SqlParameter[3];
    arParms[0] = new SqlParameter("@Email", SqlDbType.NVarChar);
    arParms[0].Value = Email;
    arParms[1] = new SqlParameter("@Password", SqlDbType.NVarChar);
    arParms[1].Value = Password;
    arParms[2] = new SqlParameter("@IsFound", SqlDbType.Bit);
    arParms[2].Direction = ParameterDirection.Output;
    pTransactionSuccessful = true;
    try
    {
        SqlHelper.ExecuteNonQuery(ConnectionString, CommandType.StoredProcedure, "ValidatePerson", arParms);
    } catch
    {
        throw new Exception("Error executing stored procedure");
    }
}

Very bad answer:

@Email is the name of the SqlParameter in the first element of arParms. ValidatePerson is a public void method. It accepts as parameters email, password, and connectionString. arParms is a SqlParameter that is created as a new SqlParameter with three elements. arParms[0] is a new SqlParameter with @Email with a data type of NVarChar. The second line for arParms[0] is set to equal Email. pTransactionSuccessful is set to true. The method ExecuteNonQuery is passed ConnectionString, a stored procedure, the name of the method, and arParms.

Good answer:

First let me answer your question about @Email. @Email is an input parameter for the stored procedure that is being called by the data access method. Notice that it is in "" marks. It is the name of the parameter and in SQL Server, stored procedure parameters (as well as local variables) begin with "@". Now I will give you a brief overview of the syntax. This is a public method meaning there are no scope restrictions on access to this method. The name "ValidatePerson" will pop up in the intellisense dropdown box when you instantiate an instance of the data access class in the business or presentation tier. Void, as I am sure you know, just means this method does not return a value—isn’t a function). In this case (though not shown), it simply sets a property value. The methods input parameters are in (). These must be provided when the method is called by another tier in the application. Again, these will show up in the intellisense window when type “(“ after method name. These values are meant to be passed to a stored procedure. The way we pass them is to place them in an array and then we pass that array to the sqlhelper class which abstracts much of the data access logic common to this tier. The helper class will traverse the array and send these parameters to the stored procedure. Notice there are three elements in the array [0,1,2]. The first line of each element specifies the name of the parameter and its data type. In .Net there is a strongly typed data type specifically for Sql Server called “SqlDbType. These data types map directly to the datatypes in a stored procedure. The second line of each element specifies what the value will be. In this case, it is the value of the input parameter. The last parameter is different in that it is an output parameter. That is, it is assigned a value once the command has been executed. In this case it is a yes/no value whether the person was found in the database. pTransactionSuccessful is property that indicates whether there was an exception at the data tier. This property can be checked by the calling method to handle the exception. The “Try” is the first part of the try catch block. A try catch block is intended to capture and recover from unexpected results or behavior in the application. “ExecuteNonQuery” is an overloaded method (meaning it has multiple signatures…) that executes an SQL expression that does not return a result. In this case, the stored procedure
simply sets an output parameter “@IsFound.” The connection string includes credentials necessary for the ASP.net application to login into the SQL Server instance and execute some command. In this case, the command is a store procedure “CommandType.StoredProcedure”. The name of the stored procedure is listed next followed by the parameter array.

[Your answer does not have to be as long as mine, but it should “explain” not just restate the syntax. Which answer would you want to receive from your colleague?]

Be able to draw and explain the n-tier architecture as we have used it in this class.
Be able to explain how we implemented the n-tier architecture in Visual Studio.
Be able to explain the purpose of each tier in our architecture (including the helper class).
Be able to explain what is meant by “role based security” and give an example from your project.
Be able to answer multiple choice and short answer questions from my presentation of:
   .Net overview
   ASP.Net overview
   Classic ASP versus ASP.Net
   SQL Injection attack
   ADO.Net overview
   [If there are terms you don’t understand from my presentations, you should look them up on the Web…]

Be able to answer multiple choice and short answer questions from the following Microsoft Patterns and practices documents. (See questions embedded in podcasts for examples.)

Chapter 1: What is Software Architecture?
   What is Software Architecture?
   Why is Architecture Important?
   The Goals of Architecture
Chapter 2: Key Principles of Software Architecture
   Overview
   Key Design Principles
Chapter 3: Architectural Patterns and Styles
   What Is An Architectural Style?
   Combining Architectural Styles
   Client/Server Architectural Style
   Layered Architectural Style
   N-Tier / 3-Tier Architectural Style
Chapter 4: A Technique for Architecture and Design
   Application Overview
   Be able to relate this overview to Cesar’s architecture
   Cesar’s architecture (focus on podcast content)
Chapter 5: Layered Application Guidelines
   Overview
   Logical Layered Design
   Design Steps for a Layered Structure
Chapter 8: Data Layer Guidelines
Overview
Data Format
Queries
Stored Procedures
Stored Procedures vs. Dynamic SQL