1. Universal Usability (2 * (5 + 2) = 14 pts)

Computer users are extremely diverse in their background and circumstance; Universal Usability is a principle that recognizes this diversity, and stresses the importance of accommodating the needs of all users, not just users with special needs such as children, older adult or the physically disabled.

i. Explain two other circumstances where Universal Usability can be applied (i.e. for users without special needs such as the above).

ii. Describe an example design implementation of each.

Examples:

People of different skill levels, not only of computer skills, but also of the domain skill of the application. Ideally, an application can cater to a large range of skill levels, from novice to intermediates to advanced users by providing multiple levels of functionality, e.g. through layered interfaces. For example, drawing applications commonly provide a tool palette containing large icons of brushes and paint-buckets, which are easily identifiable and used by beginners; while at the same time, providing keyboard shortcuts for advanced users.

People with access to computing devices of varying capabilities, e.g. in terms of screen size. For screen-size, interfaces should be designed to be usable at an acceptable common denominator (e.g. 640-by-480), yet be scalable to larger screen sizes such that the additional space can be utilized effectively. In some applications, this may require two (or more) different interfaces to be designed for specific screen-sizes, e.g. one for cell-phones, and another for PCs. An example of the latter would be Google’s search engine, where in addition to the standard web-interface they have designed a specific version for cell-phones: results are shown as a list of very short (< ten words) descriptions optimized for small screens.

Others such as language, cultural background, usage environment (e.g. mobile as well as stationary devices).

2. Golden Rules (4 * 2 = 8 pts)

Name four of the eight Golden Rules, and provide one example of each.

Strive for consistency: use consistent font/colors/layout/sequence of actions to achieve a task, e.g. the consistency between the Open and Save dialog boxes.

Cater to universal usability: (explained in part 1) e.g. provide easily available translations into common languages.

Offer informative feedback: e.g. when dragging (moving) a file from one folder to another on a desktop file manager, provide an outline of the file icon that follows under the mouse cursor.
Design dialogs to yield closure: e.g. an online retailer’s check-out should include, at the very end, a confirmation receipt that clearly indicates the end of the check-out process.

Prevent errors: e.g. instead of using a text-box for entering dates, provide a calendar widget where users can click on to select a date. (Note: there is no possible way a user can enter an invalid date! This isn’t the same as an error message.)

Permit easy reversal of actions: e.g. in MS Word, a multi-level undo is provided to easily revert edits in reverse chronological order.

Support internal locus of control: e.g. a web-browser should immediately indicate when the network is unavailable if possible, rather than delaying until after a 2 minute timeout before doing so.

Reduce short-term memory load: e.g. avoid having a multi-page form; if not possible, repeat important information (such as short instructions) across the pages. (Note: recalling passwords across sessions doesn’t involve this principle: short-term memory doesn’t last that long.)

3. Object-Action Interface (OAI) (8 + 4 + 4 = 16 pts)

OAI provides a useful model for understanding, as well as designing for human-computer interaction.

i. Describe the OAI model with examples from the e-mail client application depicted in Figure 1.

ii. Explain the importance of separating tasks, interfaces, objects and actions.

iii. Draw a simple (abstract) hierarchy to illustrate these distinctions.

The OAI model is based on the way humans solve problems by decomposing them hierarchically into smaller problems. OAI separates the design problem into tasks (e.g. managing company mail), which corresponds to the intentions of the user; and interfaces (e.g. Outlook Express), which corresponds to the representation of the task in the HCI applications. Both tasks and interfaces are further decomposed into object hierarchies, which organize objects into their components/compositions (e.g. servers->mailboxes->emails); and action hierarchies, which decompose actions into smaller steps (e.g. replying to an email can be broken into reading the original/writing a reply/sending the reply). In a well-developed OAI model, the task and interface parallels each other.

Separating tasks, interfaces, objects and actions is advantageous to both developers and users. For developers, the OAI model forms the plan for designing and building the application, since it mirrors common software engineering practices (e.g. object-oriented programming). For users, a well-developed OAI model allows them to more easily understand how an application works as the interface they see would correspond to their expectations of how it works.

(see pp. 97 of the textbook for the hierarchy)
A group of bright UMD undergraduates, frustrated with the UM-Shuttle system, have approached the UM-Transportation Director with a proposal to create an interactive Shuttle Routes and Schedule application for commuters. They have even created a prototype Flash application where users are able to select and see one or more routes on a map, and display the schedule as well. As a member of this group, you have decided to pursue this project more formally using the LUCID methodology. For the following stages:

1. Discovery: Imagine that you are a team of 3 students with two weeks to complete this phase. How will you propose to the Director to identify the characteristics and needs of your target users? Also provide a sample timeline that identifies the stages of your study. (15 pts)

The important points are:

i. Provide a timeline for the Discovery stage! This can include brainstorming, interviews, surveys, observations, etc. to collect data for your project. You can also include time for doing the statistics or preparing a report, but it’s not important for the problem. Do not include any design/prototyping yet!

ii. Identify the characteristics of your target users. Are they frequent commuters (probably not, if you think about it, since they would already know the routes)? Students? Children? The style of your interview/survey should incorporate this either with appropriate questions (e.g. “Describe your occupation/age/etc."), or by pre-targeting a set of users, with reasonable explanations (e.g. UMD students; not children, since they can’t commute alone! drivers, depends on your explanation). The latter part is important to keep your project logistics within practical limit.

iii. Identify the needs of your users. What would your target find useful? You can study existing systems, have surveys with open-ended questionnaires (“What do you if you don’t know which bus to take?”), observe people at bus stops looking at timetables, and so on. Riding the shuttles isn’t terribly helpful: the application isn’t about improving the shuttles, it’s about looking up which shuttle to take to begin with.
2. Design: Imagine that at this point, you have created a new-improved design based on the discovery study. With the same team of three students and another two weeks, how will you propose to the Director to evaluate your new design? Describe briefly the setup of your study, including the details you will be evaluating, the users you intend to study, as well as example interpretation of possible results. (15 pts)

   The important points are:

   i. The type of study: is it an experiment? An expert review? A Cognitive walkthrough? A qualitative study?

   ii. What are you measuring (e.g. how long it takes for the user to find a route? How many mistakes?)? If a questionnaire, what sort of questions will you ask to elicit the measurement (e.g. “What do you find difficult?”)? If an expert review, what guidelines will you be using (e.g. Golden Rules)?

   iii. Who will you study? They will have been identified in the Discovery stage, but may be slightly different/limited for your study (e.g. students paid to participate). However, you should not include every type of user you can think of: that defeats the purpose of having a Discovery stage to begin with!

   iv. Example interpretation: e.g. if the participants are spending too much time looking at the screen (without interacting), perhaps it takes too long to understand the interface; if the expert review notes that font usage is inconsistent, well then font usage is inconsistent (and maybe too much text); if your participants whiz through the experiment in seconds, either your interface is perfect (rather unlikely) or your tasks may be flawed (excessive guidance/too trivial).

5. Tools (2 * 6 = 12 pts)

When designing a user interface, there are several considerations when choosing a tool to build the user interface with. What would motivate you to choose a graphically oriented tool such as Adobe Flash or PowerPoint, over a programming-oriented tool such as Java or C++? Vice-versa?

   The primary consideration would be when in the stage of development is the project at. I would choose to use Flash or PowerPoint in the very early stages of development when the user interface design is still being undergoing changes and superficial evaluation, since it is far less costly (in terms of time, resources and effort) to make change the interface graphically in Flash or PowerPoint. Also, if the application doesn’t require too much customization (and can be deployed over a web-based platform), the entire application can be developed and deployed in Flash.

On the other hand, if the project requires a substantially complex user interface such as 3D graphics (say, for a flight simulator), it would be far better to develop the application in a programming-tool such as Java or C++ (with an appropriate 3D toolkit). Also, when an interface design has been settled on, it may also be useful to convert the implementation to Java/C++ to provide better (interactive) performance or support a wider range of computing devices (universal usability).
Essentially, the choice between a graphically oriented tool over a programming-oriented tool is a trade-off between the speed of development against the flexibility of the tool. Familiarity, availability of resources, etc. are other somewhat less important considerations.

6. Direct Manipulation ($6 + 14 = 20$ pts)

1. Name and describe in one sentence each the three principles of Direct Manipulation. (6 pts)
   
   *I didn’t really look at the description for this, as long as you named (important parts of) the principles, you’d get the points.*

   i. **Continuous representation** of the objects and actions of interest with meaningful visual metaphors—that provides a consistent and familiar (to real-life) experience of manipulating objects.

   ii. **Physical actions** or presses of labeled buttons, instead of complex syntax—to mimic the way objects are interacted with in the real-world

   iii. **Rapid, incremental, reversible actions** whose effects on the objects of interest are visible immediately—such that user’s actions are immediately reflected as feedback.

2. The Figure 2 depicts a graphical computer desktop that is typical today. (14 pts)
   
   i. Explain two examples of how it applies the principle of Direct Manipulation.

   It uses a “Desktop” metaphor, organizing object/documents into files and folders (represented by symbolic icons), just as would be expected on a real-life desktop.

   Moving files between folders involves dragging (clicking on the mouse, and holding it down while moving the mouse cursor to a different location), which mimics the motion of carrying a physical file.

   ii. Explain one example of how it violates the principle of Direct Manipulation.

   Copying a file does not conform to Direct Manipulation. It requires either a context-menu (right-click), or some shortcut (shift-dragging on Windows), both of which are syntactical. (*I believe the Xerox Star had a DM implementation, by providing a “copier” icon where you can drag documents onto).*

   On some Linux desktops, there isn’t a “trash” icon; deleting a file involves a menu item or using a shortcut.

   There are two distinct representations of folders: as (closed) icons, and as open windows; they are one and the same, yet they can appear in two different places simultaneously, and support distinctly different operations (e.g. you can’t drag an open folder window to the trash). (*Mac OS up till version 9 did distinguish open and closed folder icons: open folder icons are blurred out; there can only be one window per open folder*)

   In Figure 2, it’s not clear what the buttons on the title bar mean (especially the one on the top-right; that’s completely unintuitive if you’ve ever used OS X). There isn’t any visual metaphor that can aid here.
Your answer may be based on your personal experiences.

Figure 2: Graphical Desktop Interface