The objectives of this exercise:

- **Computational:**
  - Decomposition: Breaking down a comprehensive description of an object into detailed descriptions of (1) its function(s), (2) the need(s) it fulfills and (3) its physical attributes.
  - Abstraction: Describing a generic example of an everyday object by focusing on its essential or typical functions and physical attributes without regard to trivial variations (such as color other variations).
  - Evaluation: Logically, methodically and completely describing an everyday object in sufficient detail and in clear, non-technical language such even if the name of the object is omitted that any reader could recognize the object and understand how it works.
  - Learning about the description and design process for modular programming by describing an everyday object in detail including why the object is needed and how the object functions
  - Learning about abstraction and function characterization by identifying properties of an everyday object
  - Learning about specifying input, output, and function of a module or object clearly
  - Learning about hiding details of the inner workings of an object without sacrificing the functionality of the object

- **Creative:**
  - Surrounding: Looking at an everyday object in new ways, using all of your senses to understand how it’s made and how it functions.
  - Capturing: Using written language to describe all the different details and characteristics of this everyday object so you can work with it in new ways.
  - Challenging: Describing the operations of an everyday object with words and also as a computer program.
  - Broadening: Imagining that this everyday object doesn’t exist and acting like its inventor and trying to fulfill a need by creating something new and useful.

- **Collaborative:**
  - Being open to all points of view and resolving group conflicts in a constructive way.
Giving and receiving thoughtful and constructive feedback in order to develop your group project.
Meeting group deadlines, including completing your individual work in a timely manner.
Contributing substantively to the group process, using your skills, knowledge and experience.
Working together as a team to achieve a common goal; being able to both compete against and cooperate with other teams.

**Problem Description**

For the next two weeks, you will be using language to try to clearly and thoroughly describe the functions of an ordinary object that you might use every day. You will be acting like the inventor of that object, imagining that it does *not* yet exist and trying to describe what need would be fulfilled by your (new) object and how (specifically) it will function.

Each group will set up a wiki page on [agora.unl.edu](http://agora.unl.edu). The name of this page should be: “Everyday Object by <Course> Group <Name>” where <Course> is the course abbreviation and <Name> is your group name (e.g., Everyday Object by CSCE 155A Group Awesome).

*Any member may create the group page. Note that there should be only one page created per group. Before you create a new page, make sure that one doesn’t already exist.*

Your group will **choose** a common, functional, everyday object from the list in Appendix A. Your challenge is to imagine that this object does not exist and to describe in written language (1) the mechanical function of your object, (2) what need is fulfilled by this object, and (3) the physical attributes of your object.

You must describe the object’s function, the need it will fulfill and its physical attributes in clear, non-technical language that any user could understand. Your description must be specific enough so that someone who had never seen the object could recognize it and understand how it works and understand what benefits it provides.
This description process is very important for developing algorithms in computer science. An algorithm consists of the series of steps necessary to solve a given problem. By using algorithms, we can solve problems without having to constantly “reinvent the wheel” and spend the time, money, etc. to figure out each step ourselves. However, if any of these steps are unclear, we can have difficulty following the algorithm, which can lead to serious repercussions. For example, if the formulation algorithm used to mix the concrete for a road or bridge is unclear, workers may make a mistake during pouring leading to reduced service life. Or, if the business plan algorithm for a new company is confusing, venture capitalists may be reluctant to invest leading to failure of the business. To avoid these repercussions, the developer should make every effort to make the algorithm’s description as clear as possible for all steps. In other words, characterization of processes is key; it allows us to abstract a process and then convert it into a formal problem or solution.

1. **WEEK ONE [20 POINTS]**

1.1. **WRITTEN DESCRIPTION**

Over the course of the first week, generate your written description of your object. Your description must include the following:

1. The mechanical **function(s)/use(s)** of the object (E.g., “This object, which I call a “hammer” is used to drive nails into wood or other materials . . .)
2. What **need(s)** the object fulfills (E.g., Instead of using a brick to drive nails, the hammer . . .)
3. The **physical attributes** of the object. These include:
   - components or parts (E.g., “The hammer has a handle and a head. The head may have a curved claw like end so that nails can be removed . . .”)
   - shape or materials (E.g., “The head is metal. The handle may be wood or metal and may have rubber padding . . .”)
   - general dimensions (E.g., “The hammer may range in length from . . .”)
This written description is very important for writing functions in computer science. Functions are blocks of code written to perform a discrete task. With functions we do not need to repeat the same blocks of code multiple times in the same program. This makes the source code more organized and also makes future changes to the code easier (and less error prone) since we only have to change a function once rather than updating each block separately. When you first start programming, you can get away with writing functions in an ad hoc manner while coding. However, for larger programs, and when working with a group, a written description is critical to making sure all the functions are written correctly. For example, imagine writing all the functions necessary for the F-22 Raptor jet fighter which consists of about 1.7 million lines without starting from a detailed description.

Your description should start with the name of the object and must have a minimum of 150 words. You must have at least one function, at least one need, and a minimum of 6 physical attributes for your object to receive full credit. Keep in mind that physical attributes may involve all of your senses. Remember, to receive any points you must have contributed to the description of the object by writing or editing the description in the body of on the wiki page (not the Discussion area).

2. **WEEK TWO**
2.1. **ANALYSIS AND REFLECTION [20 POINTS]**

Post your Analysis and Reflection responses in the Discussion area of your page, NOT in the body of the page.

You are expected to discuss these analysis and reflection questions among your group. One member must start a new topic for EACH Analysis or Reflection by selecting “New Comment.” In the Topic area, type “Analysis” or “Reflection” and in the Comment area, paste in the Analysis or Reflection questions. Using the Analysis or Reflection questions as prompts, each member will post his or her responses as a reply to the original comment. This process will keep the group’s Analysis and Reflection in separate threads and make it easier to follow the development of your answers.

You will be graded individually based upon your contributions to the group Analysis or Reflection. In order to receive individual credit for Week 2, each group member must contribute to the answers to these questions. **Group members who do not contribute to the Analysis or Reflection Discussion will not receive points.**
2.2. **ANALYSIS [10 POINTS]**

Respond to these questions: (1) Considering your object as a computer program, draw a diagram that shows all its functions as boxes (name them), and for each function, its inputs and outputs. Are there shared inputs and outputs among the functions? (2) Looking at the list of physical attributes, organize these such that each is declared as a variable with its proper type. Can some of these attributes/characteristics be arranged into a hierarchy of related attributes/characteristics?

2.3. **REFLECTION [10 POINTS]**

Respond to these questions: (1) Considering your response to Analysis 1, are there functions that can be combined so that the object can be represented with a more concise program? Are there new functions that should be introduced to better describe your object such that the functions are more modular? (2) Have you heard of abstraction? How does abstraction in computer science relate to the process of identifying the functions and characteristics as you have done in this exercise?

This diagraming process is important for problem analysis in computer science particularly and in all problem solving in general. Just as we have organized similar blocks of code using functions, we can organize functions with similar inputs and outputs together. This process provides a “big picture” view of the program which is vitally important for initial development of the code and future changes. For example, software for large insurance companies may contain many similar functions used for different insurance plans all of which need to be updated after a law is changed.

This abstraction process is used in many programming languages to allow similar functions to be written more concisely, and to be more easily understood in a conceptual way. The basic idea is to write the source code completely for only one function in such a group. The rest of the functions use this function as a baseline adding only the source code necessary for their specific tasks. In this way, source code common to multiple functions needs to be written only once. Again, the main advantage is in terms of organization—including defining the relationships between functions—and making updates to the functions. For example, in a simulation game, you could have hundreds of functions for customizing character appearance. By using abstraction, you can avoid having to update all hundreds of functions when you change the common source code on character appearance.
DEADLINES AND HAND-IN

Week 1 Deadline – [XXX, 11:59 p.m.]: You should have completed the description of the object by the Week 1 deadline above. This description should be posted to the body of your group’s wiki page.

Week 2 Deadline – [XXX, 11:59 p.m.]: Your analysis and reflection responses are due by the Week 2 deadline above. Your individual Analysis and Reflection comments must be posted in the Discussion area of your group’s page.

GRADING

Week 1. Object description posted in the body of the group wiki page. Each group member must contribute to the description by writing or editing to receive points. The description must include at least one function, one need and six physical attributes for full credit.

Week 2. Analysis and Reflection: graded individually. Each member must post in the Discussion with a minimum of 3-5 coherent, relevant sentences for full credit.

Late work will not be graded.

APPENDIX A. LIST OF OBJECTS

List of objects:

- zipper
- mechanical pencil
- binder clip
- ziploc bag
- scissors
- tape measure
- stapler
- nail clippers
- umbrella
- flashlight
- can opener
- clothespin
- sticky notes (Post-Its)
- toilet paper holder
- revolving door
- computer mouse
- pliers
- ball point pen
mousetrap
screwdriver
pocket calculator
sundial
belt
solid air freshener

**APPENDIX B. EXAMPLE OF PATENT DESCRIPTION FOR SCOTCH TAPE**
The US Patent Database, uspto.gov, has examples of how common objects were described for patent purposes.

You can view the original patent application for masking tape and its extension, Scotch tape (Patent 1,760,820; May 27, 1930) at http://patimg1.uspto.gov/.piw?Docid=1760820&idkey=NONE

Your described object should be able to meet the requirements of a “utility patent.” That is it is new, useful, functions as described, is non-obvious, and is not simply a combination of other existing inventions or a remaking of an existing object.

END OF EXERCISE