CS 247: Web and Distributed Programming
Project 2 (Adventure Game)

Requirements
For this project you will work in teams of two. Please pick your team by next week and send me an email with both team member names. If you don't pick your team by the end of next week, I will assign the team myself.

Your team will design and implement a graphic adventure game using HTML5, CSS3 and JavaScript. In this type of game, the player traverses a set of "rooms", which can represent any location (e.g. an actual room, a corridor, outdoor space, etc.). This is typically done by displaying an image of the room along with a textual description and having the user type in different commands that allow for navigating the environment (e.g. "go north"). In each room, the player may perform certain actions that allow him/her to interact with the objects in the room. The typical actions are: "take" and "drop". Taking an object inserts an item into the player's inventory and dropping the item removes it. The player receives 1 point for dropping one item into a specially designated room (the repository). The goal of the game will be to achieve the maximum score (i.e. take all the items in all the rooms and drop them into the repository room). In order to prevent cheating, you need to make sure the player cannot pick anything up in the repository room. The actual number of rooms and items and hence the maximum score, will be up to you, but you should have at minimum 7 rooms and 4 items to collect.

Before you start coding, design the game:
1. Think of the setting, the plot, and a title for your adventure game. The setting could be any location. For example, you could place the adventure in the Science Center of Lewis University. The plot could be about a student that has left his stuff in different labs and now needs to collect it and get out of the building. Try to be as creative as possible.
2. Decide on the room structure. How many rooms (at least 7)? How are they connected? What will be the textual descriptions? What items will they have (at least 4)? Which room will be the repository? Draw a map of the room structure as a reference.
3. Either: take photos of each location that represents a room or draw a picture for each room. You will have to display the corresponding image for each room. Make sure that the file sizes are small by either compressing the images or resizing them to a small size (e.g. 300x400 pixels).
4. Design the web page layout (see below) and think of how it will be styled.

The web page should contain the following:
1. Header text (your names, project#, course name, semester)
2. Title of your adventure game
3. An image of the "room"
4. Text with the room description
5. A list of items that are available to be picked up in the room
6. A list of items in the player's inventory
7. Score display
8. A textbox where the player can type in commands (*prompt textbox*)

All of the elements in the web page should be styled using CSS3. You will be graded based on the styling.

Event Processing
Upon typing something in the prompt textbox and pressing enter, a JavaScript function should be executed which will do the following:

1. Parse the text in the prompt (i.e., find out which command was entered).

2. Execute the following depending on the command:
   `"go north"`, `"go south"`, `"go west"`, `"go east"`
   Switch the current room to be the room connected to the north/south/west/east, correspondingly. If there is no room connected to that direction, alert the player that the move cannot be made.
   `"take <item>"`
   Check if the item is currently in the room. If not, alert the user that the item s/he wants to take doesn't exist in this room. If it does exist, delete the item from the list of items for this room and add it to the player's inventory. Also, prevent the user from taking anything from the repository room.
   `"drop <item>"`
   Check if the item is currently in the player's inventory. If not, alert the user that s/he doesn't have the item. Otherwise, delete the item from the player's inventory and add it to the room's item list. If the room where the item was dropped is the repository room, add 1 to the player's score.
   Any other command
   Alert the user that the command he entered is invalid

3. Refresh the display (update all values).

4. Check if the game is over. If it is over, redirect the browser's window to a game over page, which will display the text "Game Over. You Won!".

Additional requirements:
- Your page should display correctly when using the Google Chrome browser.
- You need to validate both your HTML5 file and your CSS3 file using the W3C validator service. The link for the validator service is provided on blackboard. Your files should have no errors and validate successfully as HTML5 and CSS3, respectively.
- Your JavaScript should produce no errors. Check this using the developer tools in the browser.
- All of your code needs to follow proper programming style. This includes the use of whitespace (line breaks and indents) to separate sections of your code and comments to document your code.
- At the top of each file, put a block comment containing your name, course name, project number, and semester.
- You need to have the HTML, CSS, and JavaScript code in separate files. The filename of each file should include your last names and contain the appropriate extension (.html for the HTML file, .css for the CSS file, .js for JavaScript). For example:

  MainPage-Jones-Smith.html or EventHandlers-Jones-Smith.js

- Include a text file (named `win.txt`) that contains the game winning sequence of commands, e.g.:
  ```
  go north
  take paper
  go west
  take pencil
  …
  drop paper
  drop pencil
  ```

  Test and make sure that it works.

HINTS

- Make sure you initialize the values (score, inventory list, starting room, etc.). You should do that after the page loads.
- Make use of Arrays to hold information about room items, inventory, room descriptions, etc.
- To capture the pressing enter event: first add an event handler to the textbox (onkeypress event) that calls a JavaScript function (assuming you use the DOM 0 event model). Then implement this function in JavaScript (in a separate file) with the following code:

```
function newCommand() {
   // Get the code for the character that was pressed
   var x;
   if(window.event) // IE8 and earlier
   {
          x=event.keyCode;
   }
   else if(event.which) // IE9/Firefox/Chrome/Opera/Safari
   {
          x=event.which;
   }

   // Check if "enter" was pressed
   if (x==13) {
      // *** YOUR CODE GOES HERE ***

      // Stop event propagation
      if(!e) var e = window.event;
      e.cancelBubble = true;
      e.returnValue = false;
      if (e.stopPropagation) {
          e.stopPropagation();
          e.preventDefault();
      }
   }
}
```

- For each room, you should have a list of 4 values that will indicate what room is accessible in each direction. A negative value may indicate that the move is not possible. For example, [1, 3, 2, -1] may indicate that from this room, you can go to room 1 if you go north, room 3 if you go south, room 2 if you go east, and that you cannot go west. You can implement this using 2 dimensional Arrays (i.e. an Array of Arrays).

What you Need to Know

- HTML, CSS
- Pattern matching using regular expressions (for recognizing commands)
- Processing events using DOM0 and DOM2 models
- JavaScript basics: arrays, functions, etc.

WHAT TO TURN IN

You should turn in a single .zip file, containing all of your files (including HTML, CSS, JavaScript, and images) via BlackBoard. The name of your file should be in this format:

CS247-P2-<last name 1>-<last name 2>.zip