Marissa Golden (mjg2238)
Carolyn Fine (crf2133)
Michelle Levine (mal2291)

Virtual Classroom (VC)

Introduction

We were inspired by codeacademy to create a programming language designed for teachers. Our goal is to make it easy for a teacher to program interactive lessons for students to learn. While codeacademy is limited to teaching programming languages, we hope to expand this functionality to teach math, language, history and more. This language will be used both to teach students and to test them on the material they have just learned.

Language Overview:

Our language, VC, allows a programmer to easily create an interactive lesson in any subject he/she chooses. A whole course can be developed, complete with practice exercises at the end of each lesson and a graded test to check the student's work. Practice questions can be programmed to allow the student to receive a hint and/or try to answer a question multiple times. The gradable test at the end of a lesson will be composed of a variety of multiple choice, true/false, and fill in the blank questions. The teacher will have the option to have the test increase or decrease in difficulty based on the student's responses. Once the teacher writes his course, he can easily distribute it to his students and allow them to work through the lessons at their leisure. When a student starts a course (i.e. runs the program), a GUI will display the lesson passages and questions.

Types:

- lesson = a class that contains the tutorial, accompanying questions, and test evaluation
- Q = question class (can be either a practice or test question) - contains different methods specifying type of expected response:
    - MC = multiple choice
        - programmer can determine how many choices are available by listing all options in an array
        - '*' before one of the answers will denote the correct solution
    - fill_in = fill in the blank

        - user will have to input the word missing from the statement
        - programmer will have the option to accept more than one input as the correct (predefined) answer
    - TF= true/false
        - programmer sets bool value of correct answer
    - response = response to the question
        - stores the user's response in a string
    - correct_me = programmer writes code/a statement/a problem/etc which needs correcting
        - object has a incorrect and correct component
            - may be more than one correct answer

    ➢ Additional features the programmer can utilize:
        - numTries= define number of guesses allowed
            - perhaps unlimited for practice questions and 1-2 submissions for test questions
        - hint = a description set by the programmer
            - can bind a hint to a question
        - after a specified number of question submissions, teacher can select different question tracks depending on number of correct/incorrect submission (easy, medium, hard)
            - e.g. after 10 questions, student answered <= 3 correctly, select from easy array. 3<correct<=6, select from medium array. >6 correct, select from hard array.
- // = single line comment
- /* */= multi-line comment

Operators

- +: addition or string concatenation
- -: subtraction
- *: multiplication when used in a mathematical expression, when in front of an element of the MC array it indicates the correct answer
- /: division

- =: mathematical comparison, and string comparison
- ==: logical comparison
- !=: not equal, logical comparison
- &&: logical and
- ||: logical or

Control Flow

- if else = check logical condition, if evaluated to true, perform the block of code that follows, otherwise perform block of code that follows the else statement
- while = check a condition, and while it evaluates to true, perform that block of code
- for = loop that starts with an initial value, checks the condition and increments/decrements that value, until the condition reaches false.
- break = allows you to break a while or for loop because it finishes evaluating the statement

Built-in Functions

- record_answer = stores student's answer as the response to its corresponding answer
    - record_answer marks the end of an instruction block
    - the program execution will be paused until the student selects "next" on his/her screen
- print = prints the string that follows in the UI
    - automatically prints a new line after the string
- page_break = ends a page block and adds a next button

Example Program

```
Lesson PEMDAS {
    // Passages and questions will be printed sequentially
    print "One of the keys to solving any math equation is to know the order of
        operations, commonly called PEMDAS.  Today you'll learn the
        tools needed to solve any arithmetic equation you find!";
    // Questions are automatically printed when defined
    Q acronym = "What does PEMDAS stands for?";
```

```
// Programmer defines fill in answers
acronym.fill_in = [“parenthesis”, “exponent”, “multiplication”, “division”,
        “addition”, “subtraction”];
/* set the # of attempts to infinity for practice question
 * if user presses “next” and his answer is not correct, the UI will ask if
 * the student wants to try again or move on
 */
acronym.numTries = inf;

// records answer from the user and ends the block, from this point on
programmer can use the response to check answers
record_answer;

if ( acronym.response == acronym.fill_in) {
        print “Amazing!”;
}
else {
        print “Not quite.  It’s “ + acronym.fill_in;
}

print “If you’re looking for an easy way to remember the order, just think:
        Please Excuse My Dear Aunt Sally!”;

page_break;

print “Ok, let’s practice!”;

Q q1 = “What is (1+1)/2 ?”;
q1.MC = (*1,2,3,4);
/* if user is incorrect, he will continue trying until he gets the correct answer or
 * uses up all of his attempts
 */
```

```
        q1.numTries = 3;

        Q q2 = “True or False: 6^2 / (1 - 4) = -12”;
        q2.TF = True;
        // if numTries is not specified, the default setting is 1 try

        record_answer;
}
```