Objectives Students will be able to…

* **Write** a game that plays Rock-Paper-Scissors.

Assessments Students will...

* **Submit** a program at the end of 2 or 3 class periods.

Homework Students will...

* **Outline** Chapter 5 (up to and including HW 5.3)

# Materials & Prep

* **Projector and computer**
* **Whiteboard and** **markers**
* **Classroom copies** of WS 3.16
* **Link** to rock-paper-scissor game (<http://tinyurl.com/bubyvtu>)
* **Poster** 3.16.1
* **Poster** 3.16.2

Truth tables are an important tool, especially for some AP test questions. If you are not familiar with truth tables, watch this 5 minute tutorial online (<http://tinyurl.com/mw8ohof>). We recommend instructors draw out the &&, ||, and ! truth tables, and maybe do an intermediate example of a two-operator expression, before getting into the De Morgan's law example later in class.

# Pacing Guide: Day 1

|  |  |
| --- | --- |
| Section | Total Time |
| Attendance & student play | 5min |
| Introduction to Boolean Logic  *Add 5-10 minutes if using truth tables* | 10min |
| Student programming activity | 40min |

# Pacing Guide: Day 2

|  |  |
| --- | --- |
| Section | Total Time |
| Attendance & outline collection | 5min |
| Whole-group troubleshooting and discussion | 10min |
| Student programming activity | 40min |

# Procedure

*In place of bell-work, invite students to warm up for class by visiting the online Rock Paper Scissor game at the link above. After a quick review of Boolean logic and variables, students will be asked to build their own Rock Paper Scissor game. This programming project should take between 2 and 3 55-minute class periods to complete*

## Attendance & Student Play [5 minutes]

## Introduction to Boolean Logic [10 minutes]

1. Students should have already reviewed this material as part of last nights’ homework assignment. Before moving on to purely mathematical examples, start with a real-life example of how we apply logic. Be sure to change P and Q to statements that are relevant to your students.

* + - * P: It is a holiday.

Q: My family is having dinner together.

!(p || q) ß It is not the case that (It is a holiday OR my family is having dinner together)

!p && !q ß It is not a holiday and my family is not having dinner together.

* + - * Review && , ||, and !, including non-examples:

AND (4 == 4) && (2 > 1) Evaluates to: True

OR (1 < 2) || (2 < 1) Evaluates to: True

NOT !(2 < 1) Evaluates to: True

if (**q == 1 || 2 || 4**){ *This doesn’t work, you have to use && or ||*

statement; *to combine full Boolean expressions.*

statement;

}

if (q == 1 || q == 2 || q == 4){

statement;

statement;

}

2. As a special note on negating Boolean expressions, review De Morgan’s law (poster 3.16.1). Have students write De Morgan’s law on their Tricky Code Cheat Sheet.

* + - * If you feel confident working with truth tables, work through the following illustration of De Morgan’s laws. On the board or projector, only write table headers as you go (putting them all up at once may lead to panic/distraction for some students).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| p | q | p ||q | !(p || q) | !p | !q | !p && !q |
| F | F | F | T | T | T | T |
| F | T | T | F | T | F | F |
| T | F | T | F | F | T | F |
| T | T | T | F | F | F | F |

* + - * + Have students help you fill out every possible combination of Boolean values for p and q.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| p | q | p ||q | !(p || q) | !p | !q | !p && !q |
| F | F | F | T | T | T | T |
| F | T | T | F | T | F | F |
| T | F | T | F | F | T | F |
| T | T | T | F | F | F | F |

* + - * + Ask students to evaluate the logical expression for each value of p and q.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| p | q | p ||q | !(p || q) | !p | !q | !p && !q |
| F | F | F | T | T | T | T |
| F | T | T | F | T | F | F |
| T | F | T | F | F | T | F |
| T | T | T | F | F | F | F |

* + - * + Now have students negate all of the values from the previous column.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| p | q | p ||q | !(p || q) | !p | !q | !p && !q |
| F | F | F | T | T | T | T |
| F | T | T | F | T | F | F |
| T | F | T | F | F | T | F |
| T | T | T | F | F | F | F |

* + - * + Ask students to complete the values for !p and !q, referring to the values from the first column.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| p | q | p ||q | !(p || q) | !p | !q | !p && !q |
| F | F | F | T | T | T | T |
| F | T | T | F | T | F | F |
| T | F | T | F | F | T | F |
| T | T | T | F | F | F | F |

* + - * + Now have students apply the && operator to !p and !q.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| p | q | p ||q | !(p || q) | !p | !q | !p && !q |
| F | F | F | T | T | T | T |
| F | T | T | F | T | F | F |
| T | F | T | F | F | T | F |
| T | T | T | F | F | F | F |

* + - * Point out to your students that these two columns are the same. Whenever two columns of a truth table are the same, we say that the expressions (column headings) are equivalent, or interchangeable.
      * In the illustration above, we showed that !(p && q) is equivalent to !p && !q. Invite students to show the equivalence of !(p && q) and !p || !q

3. Review operator precedence on your classroom poster 3.16.2 (or projected overhead, if you’re having a student make the poster for you during class).

4. Check for student understanding by having students complete Practice-It self-check questions assertions1 and assertions3.

## Student Programming Activity [40 minutes]

1. On the projector, board or as a handout (WS 3.16), give students the following programming prompt. A link to the NY Times article about rock paper scissors is included in the Materials section of this lesson plan.

**PROGRAMMING ACTIVITY**

*Exercise 1*

Write a game that plays many rounds of Rock Paper Scissors. The user and computer will each choose between three items: rock (defeats scissors, but loses to paper), paper (defeats rock, but loses to scissors), and scissors (defeats paper, but loses to rock). If the player and computer choose the same item, the game is a tie.

* A good program will prompt for user input, compare input to a computer counter-move, then output a verdict (user loses, wins, or ties), prompt for another round or exit.
* An excellent program will do all that a satisfactory program does, but will use different algorithmic strategies for choosing the best item. Teacher’s note: there is no superior strategy—the focus here is to get students to try different approaches and conclude that on their own.

*Exercise 2*

You and your partner should test out the game by playing it at least 3 times each. Keep record of how many moves it took before you won or lost the game.

*Exercise 3*

Write a program that compares 2 players. Your program should prompt for each player (1) the number of times they played the game, (2) the number of times they won the game, (3) the number of moves for each game. Have the program report which player performs better on the basis of their reported statistics. Test your program by inputting your and your partner’s results from Exercise 2.

*Exercise 4*

Write a program that plays the dice game Pig. Pig is a 2-player game where the players take turns repeatedly rolling a single 6-sided die. A player repeatedly rolls the die until one of the two events occurs: (1) either the player chooses to stop rolling, in which case the sum of that player’s rolls are added to his/her total points, or (2) if the player rolls a 1 at any time, all points from that turn are lots and the turn ends immediately. The first player to reach a score of at least 100 points wins.

**To get full credit on this assignment, you must include a structure diagram and/or pseudocode explaining your strategy.**

*Before you begin, take a moment to decide how your computer will pick rock, paper, or scissor. Should the computer pick randomly? Should it pick the same item always? Should it repeat the same item for a time, then switch strategies? Read through the New York Times article on Rock Paper Scissor, and any other online sources you choose to help you draft a plan for your program.*

2. Allow students to work in pairs, and encourage pairs to test out each others’ programs, look at each others code (to check for errors), etc. If students appear to be working too closely, remind them that each team is responsible for writing their own code.

3. Start grading student note-books in small batches (so students are not without their notebooks for too long!)

# Accommodation and Differentiation

Invite your artistic students to create posters 3.16.1 and 3.16.2 for your classroom. If needed, work through the 2 Practice-It questions as a whole class.

For your more advanced students, you might encourage them to create more complex algorithms, or more advanced interaction with the user. If they are interested in AI and machine learning, invite them to research the topic and experiment with different techniques on their Rock Paper Scissor program.