```
/*
Source for Words dataset: Code.org
*/

// Fill wordList with words from the dataset and wordPartList with corresponding parts of speech
var wordList = getColumn("Words", "Word");
var wordPartList = getColumn("Words", "Part of Speech");

// Create a filtered word list and a list for the corresponding parts of speech
var filteredWordList = [];
var filteredWordPartList = [];

// Create list of the indexes of words in the problems
var chosenWordIndexes = [];

// Create variable to track the problem number
var problemNumber = 0;

// Create list to track user's answers
var userAnswers = [];

// If the dropdown is changed to an invalid value, a warning message is displayed
onEvent("letterNumberDropdown", "change", function( ) {
  setText("homeWarningLabel", "");
  if (getText("letterNumberDropdown") == "Pick a number") {
    setText("homeWarningLabel", "Please choose a valid input.");
  }
  // console.log(filteredWordList);
  // console.log(filteredWordPartList);
});

// If homeNextButton is clicked, words are chosen to match the parameter and the screen is changed
onEvent("homeNextButton", "click", function( ) {
  if (getText("letterNumberDropdown") == "Pick a number") {
    setText("homeWarningLabel", "Please choose a valid input.");
  } else {
    filterList(getText("letterNumberDropdown"));
    chooseWords();
    // console.log(chosenWordIndex);
    setText("letterNumberDropdown", "Pick a number");
    userAnswers = [];
    nextProblem();
    setScreen("problemScreen");
  }
});

// filters the wordList and wordPartList lists by the length of the words in wordList
// wordLength {integer} - the length of the words chosen by the user
function filterList(wordLength) {
  if (wordLength == "any") {
    filteredWordList = wordList;
    filteredWordPartList = wordPartList;
  } else {
```

```
    filteredWordList = [];
    filteredWordPartList = [];
    for (var i = 0; i < wordList.length; i++) {
      if (wordList[i].length == wordLength) {
        appendItem(filteredWordList, wordList[i]);
        appendItem(filteredWordPartList, wordPartList[i]);
      }
    }
  }
}

// chooses distinct words from filteredWordList and appends their indexes to chosenWordIndexes
function chooseWords() {
  chosenWordIndexes = [];
  if (filteredWordList.length <= 10) {
    chosenWordIndexes = [0, 1];
  } else {
    for (var i = 0; i < 10; i++) {
      var isSame = 0;
      var chosenWordIndex = randomNumber(0, filteredWordList.length - 1);
      for (var j = 0; j < chosenWordIndexes.length; j++) {
        if (chosenWordIndexes[j] == chosenWordIndex) {
          i--;
          isSame = 1;
        }
      }
      if (isSame == 0) {
        appendItem(chosenWordIndexes, chosenWordIndex);
      }
    }
  }
}

// If the Dropdown is switched to an invalid value, a warning message is displayed
onEvent("wordPartDropdown", "change", function( ) {
  setText("problemWarningLabel", "");
  if (getText("wordPartDropdown") == "Pick a part of speech") {
    setText("problemWarningLabel", "Please choose a valid input.");
  }
});

// If problemNextButton is clicked, the next problem will appear
// If the screen shows the final problem, it will switch to resultsScreen
onEvent("problemNextButton", "click", function( ) {
  setText("problemWarningLabel", "");
  if (getText("wordPartDropdown") == "Pick a part of speech") {
    setText("problemWarningLabel", "Please choose a valid input.");
  } else {
    appendItem(userAnswers, getText("wordPartDropdown"));
    nextProblem();
  }
});

// resets the UI and switches to the next problem
// switches to the results screen if it is the last problem
function nextProblem() {
```

```
  setText("wordPartDropdown", "Pick a part of speech");
  if (problemNumber == chosenWordIndexes.length) {
    setText("problemOutput", "");
    problemNumber = 0;
    createResults();
    setScreen("resultsScreen");
  } else {
    problemNumber++;
    setText("problemTitleLabel", "Problem " + problemNumber);
    setText("problemOutput", filteredWordList[chosenWordIndexes[problemNumber - 1]]);
  }
}

// locates problems that were answered incorrectly and the corresponding correct parts of speech
function createResults() {
  var incorrectProblems = [];
  var incorrectProblemAnswers = [];
  for (var i = 0; i < chosenWordIndexes.length; i++) {
    if (userAnswers[i] != filteredWordPartList[chosenWordIndexes[i]]) {
      appendItem(incorrectProblems, i+1 + ")" + " " + filteredWordList[chosenWordIndexes[i]]);
      appendItem(incorrectProblemAnswers, i+1 + ")" + " " + filteredWordPartList[chosenWordIndexes
    }
  }
  var resultsWordText = "";
  var resultsAnswerText = "";
  for (var j = 0; j < incorrectProblems.length; j++) {
    resultsWordText += incorrectProblems[j]+ "\n" + "\n";
    resultsAnswerText += incorrectProblemAnswers[j]+ "\n" + "\n";
  }
  setText("resultsWordOutput", resultsWordText);
  setText("resultsCorrectOutput", resultsAnswerText);
}

// If resultsHomeButton is clicked, the screen switches to the Home screen
onEvent("resultsHomeButton", "click", function( ) {
  setScreen("homeScreen");
});
```

PDF document made with CodePrint using Prism