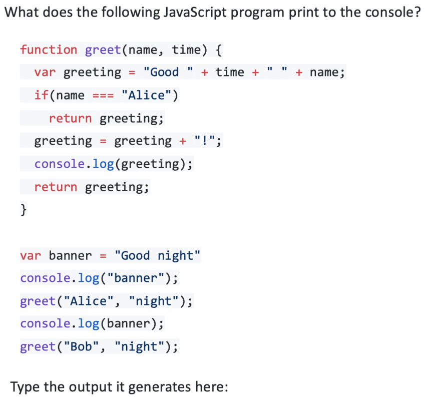
Programming Strategies Replication Guide

1 Before the Study Evaluation Test

Users who are interested to participate in our study should have a minimum knowledge of programming debugging and design. In order to filter out the very inexperienced users, participants should have pass a pre evaluation test in which they answer a couple of questions. For each participant, creat a copy of the pre-test PowerPoint. tell them that they have 5 minutes answering both questions, but wait for them at most 8 minutes to answer 2 questions. The timer started when they started reading the first question. If a participant is still working on the 2 questions after 8 minutes, stop them.

1.1 Pretest content

The content of the shared PowerPoint is shown in figure 1 and 2. The pdf version of the document is also available as ”Programming Strategies Pre evaluation tests” at http://programmingstrategies.org/ExplicitStrategiesStudyMaterials/Material.html



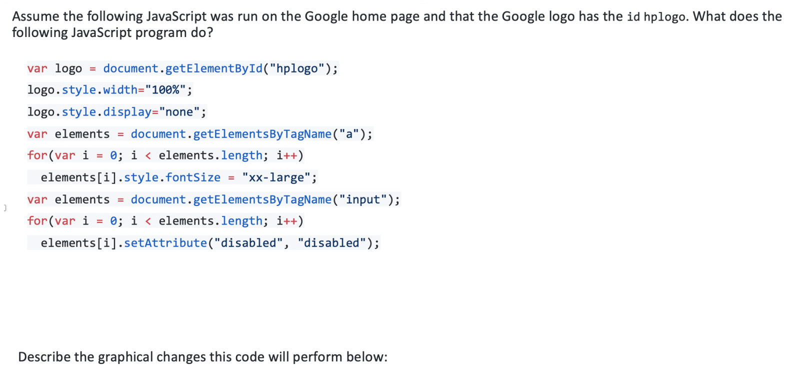
What does the following JavaScript program print to the console?

```
function greet(name, time) {
  var greeting = "Good " + time + " " + name;
  if(name === "Alice")
    return greeting;
  greeting = greeting + "!";
  console.log(greeting);
  return greeting;
}

var banner = "Good night"
console.log("banner");
greet("Alice", "night");
console.log(banner);
greet("Bob", "night");
```

Type the output it generates here:

Figure 1: Pretest evaluation question 1



Assume the following JavaScript was run on the Google home page and that the Google logo has the `id` `hplogo`. What does the following JavaScript program do?

```
var logo = document.getElementById("hplogo");
logo.style.width="100%";
logo.style.display="none";
var elements = document.getElementsByTagName("a");
for(var i = 0; i < elements.length; i++)
  elements[i].style.fontSize = "xx-large";
var elements = document.getElementsByTagName("input");
for(var i = 0; i < elements.length; i++)
  elements[i].setAttribute("disabled", "disabled");
```

Describe the graphical changes this code will perform below:

Figure 2: Pretest evaluation question 2

1.2 Pretest correct answers and grading policy

The correct answer of the pretest questions are:

- Task1
  - Banner
  - Good night
  - Good night Bob!
- Task2
  - Set width of logo to 100%
  - Hide the logo
  - Set all anchor elements font size to xxlarge
  - Disable all input elements

In the first task for each line of correct answer, participants received 1 point. The maximum grade of 3 could be gained in the first task. In the second task, if participant did not answer the the first question's correctly but answered the second question correctly, give them two points for the second answer. In both tasks, for each extra line in the output that is not correct, reduce the score by 1.

Participants passed the pretest if they have an overall score of 4 or greater on the sum of both tasks' scores. The maximum possible score is 7. If a participant pass the pretest, generate an id and record it in the participants' list (e.g., p17). Store a record of the pretest time and scores in data spreadsheet as well. Then schedule a study session for the participant. Participants who wanted to run the study online should install WebStorm on their local machine before the study session.

2 Study Session

2.1 Before the Study Session

To prepare for the study, ensure that WebStorm is installed either on your local machine for the in-person user studies or on the user local machine for online user study participants. A clone of the GitHub repository containing the experimental tasks should be created with the same name as the participant Id. The code is available online at ExplicitStrategiesStudyMaterials web page. Copy all the directories of Debugging Task, Design Task, Hello World, Jasmine Task, and Lib is copied to the new directory for the participant. Then, commit the newly created directories to Git. This will enable using a diff to identify the changes that participant makes, if any, during the study session. Next step is assigning the participant to either the control or experimental condition. Record this under the Condition column in the data spreadsheet. Generate a second participant identifier based on the condition (e.g., C3) and record this in the Data spreadsheet.

Create a new Note document with the name corresponding to the participant identifier. To take notes during the study, you will need another laptop or other device.

2.1.1 StrategyTracker Live Demo

In the experimental studies, after the participant finished reading our tool tutorials in the slides, you need to show a 5-minutes demo of how to use StrategyTracker and GitMerge strategy to fix a merge conflict in GitHub. For this purpose, you should have set up a small project with 2 branches (Master and second) with a conflict in one line. Since the demo should not take more than 5 minutes, it's better to have the changes committed in both branches before the study session. You can see the GitMerge strategy in the Strategies document online.

2.1.2 In-person vs Remote participants

In-person participants will do the study on the your laptop. Remote participants will participate by setting up a Google Hangouts or Skype connection. They should share their screen or (for the StrategyTracker Live demo), you will share their screen. For a remote participant, before the study session, you should clone the GitHub repository containing the experimental tasks by creating a directory in the repository with same name as the participant Id. At the beginning of the study session, you should ask participant to ensure WebStorm is installed. Participants can install a trial version of Web storm using the installer located at: https://www.jetbrains.com/webstorm/download. You should also ask the participant to clone the GitHub repository containing the experimental tasks.

2.2 During the Study Session

Give participants the informed consent form. Answer any questions they may have. Start the study session by starting Web Storm with the participant project directory. You should make sure participant does not access any of the folders other than contained in their participant directory. Based on the study condition, controlled or experimental, you should open either 'Programming Strategies Control Condition Study Material' or 'Programming Strategies Experimental Condition Study Material' slides(copy of pdf docs available online.) As participants begin the task, prompt participants to think aloud. Start an audio recorder for the whole study session. Have Notes document open throughout and record observations on the participant, specifically noting critical incidents where participants diverge from correct path.

2.2.1 Background Survey

In the second slide as the participant open the survey(Programming Strategies Survey google form available in the web page), give them their participant assigned Id. Ask participants to fill out the survey about their experience in designing and debugging skill. The survey answers are used to further make a record of the participants' expertise in each task. The total score they gained in each skill is calculated and normalized. Then for each skill, specify a threshold for ranking the participants as familiar or unfamiliar, which familiar stands for having good experience in task, and unfamiliar stands for having basic knowledge in related programming task.

2.2.2 Web Storm Tutorial

Users would go through a couple of slides about the Web Storm IDE. If participants got stuck, you provide whatever help is necessary for them to understand how to work with Web Storm. Ask online participants to open the shared repository in their local IDE.

2.2.3 Hello world Tutorial

Ask participants to write a hello word program in Web Storm and run it. They are provided with the basic class implementation code in their study repository. Make sure they write the code in main.js and check the output in console rather than HTML. Answer any questions they have. They should finish this part correctly to be able to go to the next slide.

2.2.4 Jasmine tutorial

Ask participants read a couple of slides about unit testing with Jasmine. This part is required as the design task is about using the Jasmine unit testing framework to write unit tests. After the Tutorial, ask to write some test cases for a piece of code. Answer any questions participants might have about how to use unit test framework in this part. Participant should not advance until successful task completion. They are allowed to use google.

2.2.5 Strategy Tracker tutorial [Experimental condition]

This section is just for the Experimental conditions, which are supposed to use a strategy in the StrategyTracker tool. They go through a couple of slides about the features of the tool and how they can use it. After they finish reading all the slides about strategy tracker(slides 9-20 in Programming Strategies Experimental Condition Study Material in the web page), give them the live demo on slide 21, on how to use GitMerge strategy to merge two branches in GitHub and fix the conflict. For the online participants, share your screen with the participants for the demo.

To warm up, participants should test a strategy in StrategyTracker to solve the problem of tower of Hanoi. The tower of Hanoi is a simple puzzle game. The objective is to move all of the discs while following some rules. We provided two links to the user, one for the related strategy to solve tower of Hanoi problem in StrategyTracker, and the other for an online puzzle game of tower of Hanoi at https://www.mathsisfun.com/games/towerofhanoi.html.

When participants start, note the time. If participants do not place StrategyTracker and tower of Hanoi side by side immediately as they are opening them, ask them to do so. If participant actions ever diverge from StrategyTracker instructions, give participants feedback immediately. If participants assess a condition in StrategyTracker incorrectly, give feedback immediately. The strategy used for this part is available as Programming Strategies at http://programmingstrategies.org/ExplicitStrategiesStudyMaterials/index.html.

3 Tasks

3.1 Design Task

Users are allowed to google. Do not answer questions about the task. Answer any question about the tool itself. If participant tries to run the code in TDD task by running the JS file rather than the HTML file, prompt the participant, “You can run the program by going to the HTML file and clicking the Chrome button in the top right.” Do not describe what the task is asking them to do. Do not answer any question about how they should start or approach the task. After they finished run the test cases on their code repository. Count the number of test cases passed and record them in the data sheet. The second measure of progress on the design task is the maturity of the solution’s verification infrastructure, which both guided and self-guided participants wrote. We counted each test that did not have syntax errors and that had a purpose related to a requirement, independent of a corresponding implementation. This resulted in an ordinal scale ranging from 0 to an observed maximum of 5 tests. Record these scores in the data sheet.

3.2 Debugging Task

In slide 26, before the user starts the task, give a demo of the correct behaviour of the game. Users are allowed to google. Do not answer questions about the task. Answer any question about the tool itself. If user asks about clarification in the statement, just guide them to read the description. Have a record of the statement of confusion in the Note records. Have all the think aloud words transcribed. Record every action participant is taking. If user forget to think aloud, remind them just one time to think aloud. Check which parts of the code the user is meeting. Check if they got close to find the fault or not. score their closeness to solve the problem with a range of 0-2, which 0 is for users who did not get close to the faulty line, 1 for those who found the faulty line but could not fix the issue, and 2 for the ones who fix the fault. Record the score in the data sheet.

4 Interview questions

When the participant navigates to each of the post-task interview slides, ask each of the questions in turn. If the participant does not directly answer the question or answers only in brief, ask additional follow up questions to elicit a more detailed response.

4.1 Task 1 Debriefing Questions

4.1.1 Experimental

1) We asked you to debug by working backwards systematically from the failure to the cause. How would you describe the approaches you used in accomplishing the task?
2) In what ways did the approaches you used help you make progress towards your goal, if at all?
3) In what ways did the approaches you used hinder you from making progress towards your goal, if at all?

4.1.2 Control

1) We asked you to debug the game. Describe the strategy or strategies you used to make progress on the task.
2) How did this strategy help in making progress on the task? 3) In what ways, if any, did the strategy get in the way of making progress?

4.2 Task 2 Debriefing Questions

4.2.1 Experimental

1) We asked you to design by first enumerating a set of scenarios, writing a test for each scenario, and then writing code to make the test pass. How would you

describe the approaches you used in accomplishing the task?
2) In what ways did the approaches you used help you make progress towards your goal?
3) In what ways did the approaches you used hinder you from making progress towards your goal?

4.2.2 Control

1) Describe the strategy or strategies you used to make progress on the task.
2) How did this strategy help in making progress on the task?
3) In what ways, if any, did the strategy get in the way of making progress?

4.3 After Study Debriefing Questions

4.3.1 Experimental

1) In what ways did you find that the StrategyTracker tool helped support you in using the strategies?
2) In what ways did you find that the StrategyTracker tool hindered you in using the strategies?
3) Do you have any questions about the goals and purpose of our study?

4.3.2 Control

Do you have any questions about the goals and purpose of our study?

5 After the study session

Stop the audio recorder, save the audio recording, including the participant identifier in the filename, and commit the participant code changes to Git.