Project 6: Response Dial Simulator

Due: Monday, October 11 at 11:59 PM

Description: iClicker remotes are used by students in classes to answer questions during lectures. This allows instructors to quickly gauge the level of student comprehension.

iClicker remotes are just one type of audience response system. Political organizations and marketers use other systems to test the effectiveness of speeches, presentations, and advertisements with focus groups. In this project, we will write a program that simulates a device used by one of these systems. We will call this device a "response dial."

Response dials are used to gauge the level of audience agreement. Unlike an iClicker remote, a response dial is always set to one of five settings: 1, 2, 3, 4, or 5. When a dial is handed to an audience member, it is initially set to 3, which indicates a neutral response. If the audience member has a positive reaction to what they are hearing, they can turn the dial to 4 or 5, which indicate agree and strongly agree respectively. Conversely, if they have a negative reaction, they can turn the dial to 2 or 1, which indicate disagree and strongly disagree.

Each response dial periodically checks its dial setting and compares it to the previous setting. The dial records the total number of times that the user choses each of the five settings. It also records the total number of times that the setting changes in a positive direction (e.g., 2 to 3), a negative direction (e.g., 5 to 3), or no change (e.g., 1 to 1).

Response dial data can be used to evaluate things like the persuasiveness of a campaign speech. For example, if an audience member chooses 4 or 5 many times and rarely changes the dial, they were probably persuaded by the speech. On the other hand, if the dial had many positive and negative changes, the audience member was probably conflicted about the candidate's message.

Objectives: Your program will be graded according to the rubric below. Please review each objective before submitting your work so you don't lose points.

1. Construct a Scanner and read input from the keyboard. (10 points)
2. Use a sentinel-controlled loop to read dial settings from the keyboard until the user tells the program to end. (15 points)
3. Store the sentinel value in a constant variable, and use this variable in the loop condition. (5 points)
4. Use a priming read to get the first dial setting before entering the loop. Read subsequent settings at the end of the loop. (10 points)
5. Use accumulators and conditional statements to update the correct accumulator based on the current dial setting. (15 points)

6. Use accumulators and conditional statements to store the number of times the dial setting changes in the positive direction, negative direction, and no change. (25 points)
7. Use print statements to output information about the dial while the user is entering settings. This includes information about the current setting and the change from the previous setting. The format of the output statements must exactly match the example. (5 points)
8. Print out prompts, echo inputs, and show results in exactly the format given in this handout. (5 points)
9. Use meaningful variable names, consistent indentation, and whitespace (blank lines and spaces) to make your code readable. Add comments where appropriate to explain the overall steps of your program. (10 points)

1324 Students: You must arrive to your laboratory on time, work actively and collaboratively with your assigned partner, and stay until you have completed the project or the laboratory ends in order to get credit for this assignment.
1323 Students: Work independently and submit your program through the Zylab.

Sample Output: Below is an example run of the program. The user input is in bold. The output of your program must match this **exactly** when given the same input.

```
Response Dial Simulator
-----------------------
Initial setting: 3
Enter the next setting (1-5) or -1 to stop.
5
Positive change: 3 to 5
Current setting: 5
Enter the next setting (1-5) or -1 to stop.
4
Negative change: 5 to 4
Current setting: 4
Enter the next setting (1-5) or -1 to stop.
4
No change: 4 to 4
Current setting: 4
Enter the next setting (1-5) or -1 to stop.
2
Negative change: 4 to 2
Current setting: 2
Enter the next setting (1-5) or -1 to stop.
-1

Response Summary
```

```
----------------
1 was chosen 0 time(s).
2 was chosen 1 time(s).
3 was chosen 0 time(s).
4 was chosen 2 time(s).
5 was chosen 1 time(s).

Positive changes: 1
Negative changes: 2
No changes: 1
```

Notice the following things about this example:

1. The user can only enter the integers 1, 2, 3, 4, 5, and -1. The first five inputs are the possible dial settings. The last input tells the program to stop. Assume the user is incapable of entering any other input because in a real system, they would be!.
2. The response summary includes the number of times that each setting (1, 2, 3, 4, and 5) is chosen by the user. It also includes the number of times that the change from the previous setting is positive (an increase), negative (a decrease), or 0 (the same).
3. The dial is initially set to 3 when given to the user. This setting does not count towards the number of times that 3 is chosen by the user, but it does determine whether the first input is a positive change, a negative change, or no change.

Implementation Suggestions:

1. Think strategically about how to implement the program. Don't just start typing and hope for the best. Try starting small and adding features slowly. For instance, write code to read a single value from the user and print it to the console. Then add a loop to read and print values indefinitely until the user enters the sentinel value. Make sure you're your sentinel controlled loop is working perfectly before proceeding. Then add code to compare the most recent value to the previous value. Print whether the new value is a positive change, a negative change, or no change.
2. Your program needs variables to store the current and previous settings of the dial. Think carefully about where in your loop these variables should be updated, particularly the variable that stores the previous value. Understanding where to do this is probably the most difficult part of the project.
3. While testing your code, you may inadvertently cause your program to get stuck in an infinite loop. If you think this has happened, check if the square button above the console is red. This indicates that your program is running. Click the button to stop the program.

Submission Instructions: Submit your source code to the Project 6 Zylab.