**Practical Exercise 10 - Using Functions in Workflows**

**Exercise Description**

**In this practical exercise, our goal is to work with different functions, both general purpose as well as status checks, to get familiar with the feature.**

Here are the instructions for the exercise:

1. Create a file named 09-functions.yaml under the .github/workflows folder at the root of your repository.
2. Name the workflow 09 - Using Functions.
3. Add the following triggers with event filters and activity types to your workflow:
   1. pull\_request
   2. workflow\_dispatch
4. Add a single job named echo1 to the workflow. The job should run on ubuntu-latest and should contain five steps:
   1. The first step, named Failing step, should exit with a non-zero code.
   2. The second step, named I will be skipped, should execute only if the previous steps have successfully completed. It should print the following text: "I will print if previous steps succeed."
   3. The third step, named I will execute, should execute only if any of the previous steps have failed. It should print the following text: "I will print if any previous step fails."
   4. The fourth step, also named I will execute, should execute only if the workflow **has not been cancelled**. It should print the following text: "I will always print, except when the workflow is cancelled."
   5. The fifth step, named I will execute when cancelled, should execute only when the workflow **has been cancelled**.
5. Commit the changes and push the code. Trigger the workflow from the UI and take a few moments to inspect the result of the workflow run. Which steps executed?
6. Add a new step named Sleep for 20 seconds before the first step of the job. The step should run the sleep 20 shell command. This gives you some time to cancel the workflow manually from the UI.
7. Commit the changes and push the code. Trigger the workflow from the UI and quickly cancel the workflow by clicking on the three dots at the right of the row containing the workflow run information. Take a few moments to inspect the result of the workflow run. Which steps executed?
8. Last but not least, let's work with some information from pull requests. For that:
   1. Add three steps at the very beginning of the job (before the sleep step).
   2. The first step, named Print PR title, should simply print the title of the PR to the screen.
   3. The second step, named Print PR labels, should print any labels present on the PR.
      1. The label information is available under the github.event.pull\_request.labels context.
      2. The information should be printed in a nicely formatted JSON string. If you are not sure how to do that, check the **Tips** section below.
   4. The third step, named Bug step, should execute if any previous steps have failed (but not if the workflow has been cancelled) **and** if the PR title contains the string fix. It should print the line "I am a bug fix".
9. Commit the changes and push the code.
10. Open a pull request by introducing some dummy changes to the root README.md file and committing the changes to a new branch. Take a few moments to inspect the result of the workflow run.
11. Test closing and reopening the PR with a different title, now including the string fix. How has this impacted the output of the workflow run?
12. Change the workflow triggers to contain only workflow\_dispatch to prevent this workflow from running with every push and pollute the list of workflow runs.

**Tips**

**Printing multi-line JSON strings**

To print a multi-line JSON string from the toJSON GitHub function, you can use the following pattern:

1. steps:
2. - name: Print PR labels
3. run: |
4. cat << EOF
5. ${{ toJSON(github.event.pull\_request.labels) }}
6. EOF

<DESCRIPTION>