# Bash Scripting Assignment: Autograder Design

## Overview:

In this assignment, you will design an autograder to evaluate student assignments. The autograder will process the submissions, run the code, compare the output, and generate a report and folders.

### Input File Details:

You will be provided with an input file (click here to see a sample) that contains the following lines of information (The bullet number indicates the line number):

1. **Use Archive:** Specifies whether the submissions are in archived format. The possible values are true or false (e.g., true).
2. **Allowed Archived Formats**: Specifies the allowed archive types if archived is used, otherwise ignore this line. The possible values are zip, rar, and tar (e.g., zip  rar).
3. **Allowed Programming Languages**: Specifies the programming languages accepted. The possible values are c, cpp, python, and sh.  (e.g., c  cpp python sh)
4. **Total Marks**: The full score for the assignment (e.g., 100).
5. **Penalty for Unmatched/Non-existent Output**: Marks deducted for each unmatched or missing output (e.g., 5).
6. **Working Directory**: The directory where the submissions are stored (e.g., /home/assignment).
7. **Student ID Range**: First and last student ID (e.g., 2005001 2005121).
8. **Expected Output File Location**: Path to the file containing the expected output.
9. **Penalty for Submission Guidelines Violations**: Marks deducted for not following submission instructions (e.g., 10).
10. **Plagiarism analysis file**: The file containing student ids involved in plagiarism (e.g., plagiarism.txt).
11. **Plagiarism Penalty**: Percentage of the total marks deducted for plagiarism involvement (e.g., 100%).

### Task Breakdown:

1. **Input File Passing:** Pass the input file's path using the command line argument (e.g., 200500x.sh -i input_file.txt). Then check whether the input file is in valid format as mentioned above. If not, give a proper usage message and exit the program.
2. **Processing Submissions**:
    - The working directory contains files named by student IDs (e.g., 2005001.zip, 2005002.rar, etc if archived is used, or just 2005001.c 2005002.py, etc ).

- Unarchive each file if is archived and check that the extracted folder has a file that corresponds to the student ID and is in one of the allowed languages. If the submissions are not archived, then create directories named as students' id and move the submission files to the corresponding directories. Check that the submission files correspond to the student ID and is in one of the allowed languages.
- Run the student's program and save its output in the format student_id_output.txt inside the student's folder.
- Compare the generated output with the expected output file. Deduct marks for any missing lines. For simplicity, only consider whether the lines of the expected output file exist in the generated output file.
- If a submission is not found for a student ID in the given range, add 'missing submission' to remarks.

3. **Marks Report Generation**:
   - Create a marks.csv file with the following columns: id, marks, marks_deducted, total_marks, remarks.
4. **Handling Submission Issues**: There are several cases where students may not follow submission instructions:
   - **Case 1**: The submission is a folder, not an archived file. If the folder name is valid (student ID), skip unarchiving and proceed with the evaluation.
   - **Case 2**: The submitted archive format is not allowed. Skip evaluation.
   - **Case 3**: The submitted file is not in an allowed programming language. Skip evaluation.
   - **Case 4**: The extracted folder name does not match the student ID. Proceed with the evaluation but log the issue.
   - **Case 5:** The submitted file is not in the valid student id range. Skip the evaluation.

     For each issue, deduct marks for not following guidelines and append remarks as 'issue case #1' if the issue is case 1 and so on.

5. **Create two directories**: Create two directories if the two directories don't exist. If exist, clear the contents. The two directories are:
   - **issues**: Move the unzipped folders with problems.
   - **checked**: Move the unzipped folders which are evaluated properly.
6. **Plagiarism Deduction**:
   - For each plagiarized submission, apply the plagiarism penalty as a deduction and append remarks as 'plagiarism detected'. Don't change the evaluated marks.

## Output:

The final output should include (some sample test cases -> click here) :

- A marks.csv file containing each student's id, marks, deductions, total marks, and remarks.
- Organized folders: issues (problematic submissions) and checked (valid submissions).

**Marks Distribution:**

| # | Tasks | Marks |
|---|---|---|
| 1 | Passing input file from Command Line | 5 |
| 2 | Input file validation | 15 |
| 3 | Handling different archiving formats | 5 |
| 4 | Generation of output file of the submitted programs | 10 |
| 5 | Handling different programming languages | 5 |
| 6 | Calculating the marks deducted for mismatch | 10 |
| 7 | Handling plagiarism | 5 |
| 8 | Handling submission issues | 25 |
| 9 | Organizing the folders as checked and issues | 10 |
| 10 | Generating the marks.csv as specified | 10 |
| Total Marks | | 100 |

## Submission Guidelines:

- ➔ Name your shell file as your ***<your_id>.sh*** (***2005xxx.sh***) and upload the file in Moodle.
- ➔ **Deadline: Saturday, 14 September 2024 11:00 PM.**
- ➔ **Do not copy from your classmate. In this case, both of you will get -100%. Also, do not copy code from any website or ChatGPT. You will get -200% in this case.**