

grading students hackerrank solution

Grading students Hackerrank solution is a common challenge among new programmers and students learning to code. This problem not only tests one's understanding of basic programming concepts but also introduces the idea of implementing algorithms to solve real-world issues. In this article, we will delve into the problem statement, discuss the approach to solving it, provide a sample solution, and analyze the solution's efficiency. By the end, readers will have a comprehensive understanding of how to tackle the Grading Students problem on HackerRank.

Understanding the Problem Statement

In the Grading Students challenge, you are given a list of student grades and a set of rules to round those grades according to specific conditions. The task is to determine the final grades for each student based on the following criteria:

1. If the difference between the grade and the next multiple of five is less than 3, round the grade up to that multiple of five.
2. If the grade is less than 38, it is not rounded because the student will still fail.

The objective is to return a list of the final grades for all students after applying the rounding rules.

Input Format

- The first line contains an integer `n`, the number of students.
- The following `n` lines contain integers representing the grades of each student.

Output Format

- Output `n` lines, each containing the final grade after rounding.

Example Input and Output

To better illustrate the problem, let's consider an example:

Input:

```
4
73
67
38
33
```

Output:

```
```\n75\n67\n40\n33\n```
```

Explanation:

- 73 rounds to 75 (next multiple of 5).
- 67 stays the same.
- 38 rounds to 40.
- 33 remains unchanged as it is below 38.

## Approach to Solve the Problem

To solve the Grading Students problem, we can break down the solution into the following steps:

1. Read Input: Gather all grades as input.
2. Loop Through Each Grade: For each grade, determine whether it needs to be rounded.
3. Apply Rounding Rules: Implement the rounding logic based on the provided conditions.
4. Store Results: Store the final grades in a list.
5. Print Output: Output the final grades.

## Detailed Steps

1. Input Reading: Read the number of students and their respective grades from standard input.
2. Rounding Logic:
  - If the grade is less than 38, it's not rounded.
  - Calculate the next multiple of five using the formula:  
``next_multiple_of_five = (grade // 5 + 1) * 5``.
  - Check if the difference between the next multiple of five and the original grade is less than 3.
  - If so, round up; otherwise, keep the original grade.
3. Output: Print the final grades.

## Sample Python Implementation

Here is a Python implementation of the Grading Students solution:

```
```python\ndef grading_students(grades):\n    final_grades = []\n\n    for grade in grades:\n        if grade < 38:
```

```

final_grades.append(grade)
else:
    next_multiple_of_five = (grade // 5 + 1) * 5
    if next_multiple_of_five - grade < 3:
        final_grades.append(next_multiple_of_five)
    else:
        final_grades.append(grade)

return final_grades

if __name__ == "__main__":
    n = int(input())
    grades = []
    for _ in range(n):
        grade = int(input())
        grades.append(grade)

    result = grading_students(grades)
    for grade in result:
        print(grade)
    ``

```

Explanation of the Code

- The `grading_students` function takes a list of grades as input.
- It iterates through each grade and applies the rounding logic described earlier.
- The results are stored in a list called `final_grades`, which is returned at the end of the function.
- The `if __name__ == "__main__":` block handles the input and output operations.

Efficiency Analysis

The time complexity of the solution is $O(n)$, where n is the number of students. This is because we are processing each grade exactly once. The space complexity is also $O(n)$ due to the storage of final grades in a list.

Edge Cases

When dealing with the Grading Students problem, consider the following edge cases:

- **Minimum Input Size:** What happens when there is only one student? Ensure that the program can handle this case correctly.
- **All Failing Grades:** If all grades are below 38, the output should reflect that no grades are rounded.
- **All Passing Grades:** If all grades are above 38, the rounding logic should be applied uniformly.

Conclusion

The Grading Students problem on HackerRank is an excellent exercise for those looking to improve their coding skills and understand basic algorithms. By following a structured approach to solve the problem, programmers can enhance their logical thinking and problem-solving abilities.

In this article, we explored the problem statement, presented a sample solution, and analyzed the efficiency of the approach. With practice, tackling similar algorithmic challenges will become easier, paving the way for more complex programming tasks in the future.

Frequently Asked Questions

What is the grading students problem on HackerRank?

The grading students problem involves reading a list of student grades and applying a specific rounding rule to determine the final grades, ensuring that certain thresholds for rounding up are met.

What are the rules for rounding grades in the grading students challenge?

Grades are rounded up if they are within 2 points of the next multiple of 5. If the original grade is less than 38, it is not rounded.

How do you handle edge cases in the grading students problem?

Edge cases include grades that are already at a multiple of 5, grades just below a rounding threshold, and grades below the minimum passing grade.

What programming languages can you use to solve the grading students problem on HackerRank?

You can use various programming languages such as Python, Java, C++, Ruby, and JavaScript to implement your solution.

Can you provide a sample input and output for the grading students problem?

Sample input: [73, 67, 38, 33]. Sample output: [75, 67, 40, 33].

What is a common approach to solving the grading students problem?

A common approach is to iterate through each grade, apply the rounding rules, and store the modified grades in a new list.

How do you test your solution for the grading students problem?

You can test your solution by using various test cases, including edge cases and large datasets, to ensure it accurately handles all scenarios.

What are some potential performance issues when solving the grading students problem?

Performance issues may arise with very large inputs, but the problem is generally efficient since it requires a linear scan through the grades.

Is there a built-in function in Python to help with rounding in the grading students problem?

While Python has a built-in `round()` function, you will need to implement custom logic to ensure the rounding rules specific to this problem are followed.

What is the importance of understanding the grading students problem for coding interviews?

Understanding this problem helps demonstrate your ability to implement algorithms, handle edge cases, and work with arrays, which are common topics in coding interviews.

[Grading Students Hackerrank Solution](#)

Grading Students Hackerrank Solution

Related Articles

- [granny wheelchair math game](#)
- [great expectations study guide answer key](#)
- [gone girl written by gillian flynn](#)

[Back to Home](#)