Project Part 1

Generative AI

Saarland University – Winter Semester 2024/25

Alexandru Andriţa

7069106

alan00004@stud.uni-saarland.de

1 Remarks

I would like to start with a remark. Since the annotations have a low score, I need to stress that in the annotating process I considered each word in the hint and I marked them quite harshly. Thus, I provided an explanation for almost each prompt to explain my way of thinking.

2 Implementation Assignment: I1

GPT-4o-mini: Looking at the summary from the *results* file, I extract the overall value requested:

- $RPass = 88.0\%$
- $REdit = 22.863636363636363$

The overall values calculated by myself according to the tables below:

- $RPass = 88\%$
- $REdit = 21.869565217391304$

For the $RPass$ of each individual problem, we are going to use the following formula (multiplying by 100 since we are calculating the percentage):

$$RPass = \frac{\text{total of programs with Correct=True}}{5} \cdot 100$$

For the $REdit$ of each individual problem, we are going to use the following formula:

$$REdit = \frac{\text{sum of distance different of default value "-1"}}{\text{total number of programs where distance != -1}}$$

Extracting the boolean *Correct* flag and *Distance* value from the *results* file:

Problem 1	Correct	Distance
Program 1	True	4
Program 2	False	-1
Program 3	True	15
Program 4	True	19
Program 5	True	13

Preprint. Under review.

The result from *Program 2* will not be included in the average calculation since *Distance* is assigned the default value -1.

Problem 1	RPass (%)	REdit
Averages	80	12.75

Problem 2	Correct	Distance
Program 1	True	1
Program 2	True	63
Program 3	True	16
Program 4	True	85
Program 5	True	54

Problem 2	RPass (%)	REdit
Averages	100	43.8

Problem 3	Correct	Distance
Program 1	True	7
Program 2	True	19
Program 3	True	11
Program 4	True	36
Program 5	True	10

Problem 3	RPass (%)	REdit
Averages	100	16.6

Problem 4	Correct	Distance
Program 1	True	53
Program 2	True	5
Program 3	True	33
Program 4	False	-1
Program 5	True	2

The result from *Program 4* will not be included in the average calculation since *Distance* is assigned the default value -1.

Problem 4	RPass (%)	REdit
Averages	80	23.25

Problem 5	Correct	Distance
Program 1	False	-1
Program 2	True	12
Program 3	True	4
Program 4	True	2
Program 5	True	39

The result from *Program 1* will not be included in the average calculation since *Distance* is assigned the default value -1.

Problem 5	RPass (%)	REdit
Averages	80	14.25

3 Implementation Assignment: I2

Analysis completed with the **GPT-4o-mini** model using as guidance document ([1]).

Problem 1	HCorrect	HInformative	HConceal	HComprehensible	HGood
Program 1 hint	1	1	0	1	0
Program 2 hint	1	1	1	1	1
Program 3 hint	1	1	0	1	0
Program 4 hint	1	1	0	1	0
Program 5 hint	1	1	1	1	1

The hint for *Program 4* is *The condition to check if the sequence is empty is incorrect; instead of 'if seq == () or []', it should be 'if seq == () or len(seq) == 0'.* The hint is correct and informative, giving the right information to the learner. On the downside, it is too detailed, also giving straight away the solution. In this case, HConceal = 0.
The hint for *Program 5* is *The code incorrectly returns 0 when x is greater than the last element, rather than appending x to the end of the sequence*. Same as the above, the hint is correct for all 3 categories (HCorrect, HInformatinve and HComprehensible). In case of HConceal, one could argue that the information is too detailed. In my opinion, the information given makes the learner think how this `append` should be done. If he is really appending the number at the end of the sequence and no position returned, then the program is not correct. Thus, in my eyes, the hint given is not presenting the solution straight away.

In this case (**Problem 1**), HGood = 1 for 2/5 cases, meaning HGood = 40%.

Problem 2	HCorrect	HInformative	HConceal	HComprehensible	HGood
Program 1 hint	1	1	0	1	0
Program 2 hint	0	1	0	1	0
Program 3 hint	1	1	0	1	0
Program 4 hint	0	1	0	1	0
Program 5 hint	1	1	0	1	0

For *Program 2*, the hint is `The 'unique_day' and 'unique_month' functions should return 'False' if the count is not exactly one, rather than checking for greater than one`. The hint is readable (HComprehensible = 1), but it is not good (thus HGood

= 0) because it sugests that the comparison `if counter > 1 then return False` should be forever deleted. We could argue that it is informative because it provides the case when `counter = 0`, but since it is too detailed (the implementation is written step-by-step), HConceal = 0.
In case of *Program 4*, `unique_month` and `contains_unique_day` are not even implemented. On the one hand, the hint does not contain any information regarding these 2 functions. We could argue that the hint is good and informative, but the solution, in the way that is implemented, works partially. The case when the searched day is not appearing at all (count = 0) is not taken into consideration. Thus, I will give HInformative = 1. HConceal = 0 as it is giving too much detail and HComprehensible = 1 as the hint is readable and contains essential information. Considering the value of the first 4 attributes, HGood = 0.

For (**Problem 2**), even though there are cases when most of the attributes have the value of 1, HGood percentage here is 0%. This is mainly due to HConceal = 0 because too much information is revealed via the hint.

Problem 3	HCorrect	HInformative	HConceal	HComprehensible	HGood
Program 1 hint	1	1	0	1	0
Program 2 hint	1	1	0	1	0
Program 3 hint	1	1	1	1	1
Program 4 hint	1	1	1	1	1
Program 5 hint	1	1	0	1	0

For the first program, the hint is `You should use a list instead of a tuple for ‘occurrences‘ and also ensure to use ‘new_lst‘ instead of ‘new_list‘ when appending items`. The hint is partially correct. The part regarding using a list instead of a tuple is not true. It works very well with tuple. The second part of the hint specifies the way to solve the true bug. But, it is too specific and reveals the solution directly, hence HConceal = 0. We could argue that HComprehensible = 1 because, even though the first half of the hint could be considered redundant, the second part truly solves the issue.
When it comes to *Program 4* (Hint: `The issue with your code is that modifying the list while iterating over it can lead to skipped elements and incorrect results`), there are 2 bugs in the code. First of all, the list is modified while also iterating through it. This could create issues related to indices when removing elements. Second problem is the typo `ls`. As the task is to generate a hint for one at least one problem in the code, we could consider the given hint good, informative, concealed (as the information given does not present the solution straight away) and comprehensive.
The hint given for the last program i.e. `You should append ‘i‘ to ‘new_lst‘ instead of trying to concatenate it, which is causing a type mismatch error` is good and informative. It clearly presents the error thrown by the program and also a states a concrete approach to solve the issue. The downside is that it has too many details. A better description would have been "Make sure there is no type mismatch in your program". In this case, the learner would have needed to find the line the issue might arrive from.

In this case (**Problem 3**), HGood = 1 in 2 of the cases. This yields HGood = 40%.

Problem 4	HCorrect	HInformative	HConceal	HComprehensible	HGood
Program 1 hint	0	1	0	1	0
Program 2 hint	1	1	0	1	0
Program 3 hint	1	1	0	1	0
Program 4 hint	1	1	0	1	0
Program 5 hint	0	0	0	1	0

When it comes to the fourth program, we could argue that the hint is correct w.r.t. one of the bugs. Indeed, the sort is ascending instead of descending, but there are a couple of more bugs. The `left` and `right` lists are defined, the merge of the lists is not done at the end and the division by "`/`" is not working as expected when the length of the list is an odd number. We could argue that the hint is comprehensive, but it is not concealed as it suggests the code solution straight away. Moreover, we could say that the given hint is informative.

In this case (**Problem 4**), HGood is again 0%. Though there are a couple of programs i.e. *Program 2,3 and 4* for which, in case the hint was not that detailed, HConceal would have been 1 which implies HGood = 1, as all other 3 attributes are 1 already. With HConceal = 1, HGood would have been HGood = 60%.

Problem 5	HCorrect	HInformative	HConceal	HComprehensible	HGood
Program 1 hint	1	1	0	1	0
Program 2 hint	1	1	0	1	0
Program 3 hint	1	1	0	1	0
Program 4 hint	1	1	1	1	1
Program 5 hint	1	1	1	1	1

The hint for the first program is almost perfect. It is a pity the hint suggests directly the change that should be made in the code.
For the second program, we have the same situation. Unfortunately, the hint is giving the solution straight away: `You are trying to access elements of integers as if they were lists by using indexing (e.g., ‘i[1]‘), which will result in an error; instead, compare the integers directly`.
When it comes to the third hint, again, it contains too many details for implementation e.g. `The condition in the while loop should check if the length of ‘new‘ is less than ‘k‘ instead of checking ‘lst‘` or `initialize ‘new‘ as your result list instead of ‘list‘ which shadows the built-in type.`. Besides HConceal = 0, the other 3 attributes are 1 as the hint is good (solves the issues in the program and the description of the problem and solution are comprehensive).
The hint given for *Program 5* is `You need to initialize the variable ‘i‘ before using it in the while loop and fix the logic for selecting the maximum elements from the list`. Although it is mentioned that `i` needs to be first initialized, the solution of the buggy program is more than that. The way the max element is appended to the list is not correct and the hint is suggesting exactly this error by `fix the logic for selecting the maximum elements`. Thus, we could argue that the hint is correct as it solves at least one bug (but in our case both), it is informative, it is not giving too much information as the `i` declaration and the logic of selecting the maximum element needs to be thought through. At the same time, we could argue that it does not contain irrelevant information and it is easily understandable. Thus, all attributes could be assigned value of 1, hence HGood = 1 as well.

In this case (**Problem 5**), the percentage of HGood is 40%. It is fair to mention that for this 5^{th} problem, the hints were the closest to the correct version. If it was not for the too detailed description for the first 3 cases, HGood would have resulted as HGood = 100%.

Overall, HGood = 24%. Looking at this value, it is quite small. What I saw during reading the model outputs and analysing the code is that for many of the hints, they were correct and they were solving most of the bugs (even all bugs in some cases). The issue that appears most often is that the hint is too detailed and already gives the solution for implementation and the learner only needs to copy it directly. No more reasoning required.

4 Implementation Assignment: I3

Phi-3-mini: Looking at the summary from the *results* file, I extract the overall value requested:

- $RPass = 36.0\%$
- $REdit = 18.11111111111111$

The overall values calculated by myself according to the tables below:

- $RPass = 36\%$
- $REdit = 18.11111111111111$

For the $RPass$ of each individual problem, we are going to use the following formula (multiplying by 100 since we are calculating the percentage):

$$RPass = \frac{\text{total of programs with Correct=True}}{5} \cdot 100$$

For the $REdit$ of each individual problem, we are going to use the following formula:

$$REdit = \frac{\text{sum of distance different of default value "-1"}}{\text{total number of programs where distance != -1}}$$

Extracting the boolean *Correct* flag and *Distance* value from the *results* file:

Problem 1	Correct	Distance
Program 1	True	23
Program 2	False	-1
Program 3	True	5
Program 4	False	-1
Program 5	False	-1

The results from *Program 2, Program 4* and *Program 5* will not be included in the average calculation since *Distance* is assigned the default value -1.

Problem 1	RPass (%)	REdit
Averages	40	14

Problem 2	Correct	Distance
Program 1	False	-1
Program 2	False	-1
Program 3	False	-1
Program 4	False	-1
Program 5	False	-1

In this case, since all distances have value "-1", none of them will be included in the average.

Problem 2	RPass (%)	REdit
Averages	0	0

Problem 3	Correct	Distance
Program 1	False	-1
Program 2	False	-1
Program 3	False	-1
Program 4	False	-1
Program 5	True	10

In this case, only the distance of *Program 5* will be included in the average because it is the only one whose distance $\neq$ -1.

Problem 3	RPass (%)	REdit
Averages	20	10

Problem 4	Correct	Distance
Program 1	True	51
Program 2	True	4
Program 3	True	5
Program 4	False	-1
Program 5	True	30

The result from *Program 4* will not be included in the average calculation since *Distance* is assigned the default value -1.

Problem 4	RPass (%)	REdit
Averages	80	22.5

Problem 5	Correct	Distance
Program 1	False	-1
Program 2	False	-1
Program 3	True	10
Program 4	False	-1
Program 5	True	25

The result from *Program 1, Program 2* and *Program 4* will not be included in the average calculation since *Distance* is assigned the default value -1.

Problem 5	RPass (%)	REdit
Averages	40	17.5

5 Implementation Assignment: I4

Analysis completed with **Phi-3-mini** model using as guidance ([1]).

Problem 1	HCorrect	HInformative	HConceal	HComprehensible	HGood
Program 1 hint	1	1	0	1	0
Program 2 hint	1	1	1	1	1
Program 3 hint	1	0	0	1	0
Program 4 hint	0	0	0	1	0
Program 5 hint	0	0	0	0	0

The hint for *Program 1* is the following: `The student's function does not handle the case where the sequence is empty, which should return 0 as per the problem statement`. Since it is directly specifying that the case when the sequence is empty is not handled correctly and it is also providing a straight and clear solution, in my opinion, HConceal = 0.
The hint for *Program 3* is `The function returns immediately after the first comparison, not accounting for the case where x should be inserted before a previous occurrence of the same value`. The first half of the hint is correct because no matter the value of x and current element in the sequence, the program will be terminated. Thus, in my opinion, HCorrect = 1. On the other hand, the second half of the sequence is wrong and is not informative at all. The description refers to the case when the element we want to insert in the sequence already exists, case that is covered with the current implementation. Thus, all others besides HComprehensible and HCorrect are 0. HComprehensible is 1 only because the first part of the hint is actually useful.
In case of the 4^{th} *program*, I considered all 4 attributes 0 because the hint is not solving the issue and it contains only redundant information. The issue the hint is describing is actually not a problem in our code. The case where the element that we want to insert in the sequence is already covered by the code. The hint is `The function does not handle the case where x is already in the sequence and should be placed before any previous occurrence`.
When it comes to the last program for this problem, the explanation is the same as in the previous hint. It refers to a case that is already handled correctly by the current implementation and the actual bug is not described at all.

Since many of hints are not relevant (either they refer to non-existing bugs or provide too many details), HGood = 20%, quite low (percentage of **Problem 1**).

Problem 2	HCorrect	HInformative	HConceal	HComprehensible	HGood
Program 1 hint	0	0	0	0	0
Program 2 hint	1	1	0	1	0
Program 3 hint	1	1	1	1	1
Program 4 hint	1	1	1	1	1
Program 5 hint	0	0	0	0	0

In case of *Program 1*, the hint i.e. `The function 'unique_day' incorrectly compares the day with the entire tuple instead of just the day part; it should be 'if birthday[1] == day:'`, is incorrect. There is no tuple comparison. The comparison is correctly done between two integers. The actual problem is the typo `day`. Instead of `day`, the code was expecting `date`, `day` not existing in the current method. Since the hint does not help the learner identify the problem and since it also suggests a code fix that does not work (actually, the hint suggests the same code as the current 'live' code which is buggy), all attributes are set 0.
This time, I am considering the hint for *Program 2* as correct, but, since it kind of specifies when exactly the function should return True and when False, the learner does not have to reason much about a code fix. Hence, HConceal = 0 (Hint = `The function 'unique_day' incorrectly returns 'True' when the day is unique, it should return 'False' in that case`).
The hint is correct and leads to the solving of one bug in the code. Although the hint for *Program 3* is quite explicit, I considered it as not giving too much information because a Python Interpreter

would show the same error message. Hint is `The variable ‘tf‘ in ‘contains_unique_day‘ is not initialized before its use, which will raise a ‘NameError‘` and in code, exactly in the named function, Python interpreter will raise the same error message as the hint is suggesting.
For *Program 4*, the hint is correct. It solves one bug and suggests the learner what should be changed i.e. a whole function is missing the implementation.
The hint is describing a problem that in the actual code is not a real issue. The true issue is that if a day does not meet the unique requirements, then we are directly returning False, without analyzing the remaining dates from the list. The hint is saying that we return after checking the first day, but the wording is confusing (hint = `The function ‘contains_unique_day‘ incorrectly returns immediately after checking the first day, not iterating through all days in the month`).

Percentage (**Problem 2**) for HGood = 40%.

Problem 3	HCorrect	HInformative	HConceal	HComprehensible	HGood
Program 1 hint	0	0	0	0	0
Program 2 hint	1	1	1	1	1
Program 3 hint	1	1	0	1	0
Program 4 hint	1	1	0	1	0
Program 5 hint	1	1	0	1	0

Regarding the *first program* and corresponding hint, the `occurrences` is actually initialized with an empty tuple, contrary to what the hint is saying (hint = `The student’s code does not initialize the ‘occurrences‘ variable with an empty tuple, which could lead to a ‘TypeError‘ when trying to use the ‘not in‘ operator on an uninitialized tuple`). In our case, the problem is the usage of `new_list` instead of `occurrences`.
Concerning *Program 2*, in my opinion, although the hint is explicitly saying `returns ‘None‘, leading to a potential TypeError when trying to concatenate ‘None‘ with a list`, I would say the hint is complete and could be used to reason about the 2 issues present in the code.
For *Program 3*, hint is correct and informative, but gives too much information. Already from the hint, the student knows the problem is the call of `sort` method. Thus, HConceal = 0. The hint also includes some characters i.e. `The student’\u25e6s code` but I do not consider this an impediment in understanding the meaning of the whole LLM answer.
In the case of *Program 4*, same as above: hint correct but too much information revealed.
As for *Program 5*, the hint is describing the issue in the code using a deep explanation. Since before I considered a hint correct and informative even though only half of it was referring to the correct approach, I will do it again. HConceal = 0 because the hint is suggesting the implementation directly i.e. `the correct operation is ‘new_lst.append(i)‘`.

In case of **Problem 3**, HGood = 20%.

Problem 4	HCorrect	HInformative	HConceal	HComprehensible	HGood
Program 1 hint	0	0	0	0	0
Program 2 hint	1	1	0	1	0
Program 3 hint	1	1	0	1	0
Program 4 hint	0	0	0	0	0
Program 5 hint	0	0	0	0	0

For *Program 1*, the hint is readable but it contains redundant and incorrect information that cannot be used to solve the bug.

For *Program 2*, hint is correct but it gives straight away the solution to fix the buggy code.
In the case of *Program 3*, some part of the hint is useful and some is not. Thus, we consider the hint correct and informative. HConceal = 0 because again, the solution is displayed right away.
The hint for *Program 4* i.e. `The student's merge function does not handle the case where one list is exhausted before the other, potentially leading to an IndexError when popping from an empty list` is not correct. The code actually handles this case correctly. There are other issues in the code that could not be detected by Phi-3-mini, thus the hint is irrelevant and redundant.
Hint not related to actual bugs in *Program 5*. Hint refers to problems that are indeed working as expected. Thus, these are not real problems.

HGood = 0% for **Problem 4**. Quite bad. I have seen that in these 5 last cases, the bugs identified either were not existing at all or the hint was showing the solution directly. Due to the nature of the algorithm required and the task that needed to be completed, the LLM might have been confused by the implementations and wrong code analysis came into play.

Problem 5	HCorrect	HInformative	HConceal	HComprehensible	HGood
Program 1 hint	0	0	0	0	0
Program 2 hint	1	1	0	1	0
Program 3 hint	1	1	1	1	1
Program 4 hint	1	1	1	1	1
Program 5 hint	1	1	0	1	0

Hint for *Program 1* refers to the case where the same maximum value could be duplicated in the output. This is no problem, there is also a test case where the entries are duplicated in the output. The hint should have described how the stop condition of the `while` loop is incorrect. Thus, all attributes are 0.
Hint is correct and informative. It explains the problems, but also give the solution straight away i.e. `incorrectly compares elements as if they were tuples, but the list contains integers; it should compare the values directly`. Thus, HConceal = 0.
In the case of *Program 3*, since the hint is specifying again directly the error that arises when code is ran, I will consider this as a correct hint as Python would show a similar message, telling the learner directly through the error message what is the issue.
In the case of *Program 4*, the hint is correct. It suggests that the list is modified while in the loop, this actually being the behavior occurs. Since it does not specify how to handle the problem or where is the issue, but just gives a general message, the learner has to reason about it.
For *Problem 5*, hint is correct for one the bugs. But, it gives again too much information by saying `it should iterate k times, not k-1`, thus I consider HConceal = 0.

The percentage of **Problem 5** is 40%.

The **overall percentage** is 24%. A general observation that I can make is the following: I see that the results generated by Phi-3-mini model are worse than the GPT-4o-mini. This is happening not only here in exercise I4, but also later when strengthening the prompt and the usage of information when generating a hint. Although there is difference between the HGood percentages (GPT-4o-mini loses many points because most of the results are too detailed). The former model is understanding and better providing a hint for the issue in code. When we are talking about Phi-3-mini, it seems that many of the suggestions are not even related to the actual problems in the buggy code.

6 Implementation Assignment: I5

According to the given tips, the `project_part1_repair.py` has been adjusted so that when GPT-4o-mini is used, the temperature is 0.7, while for Phi-3-mini, we are using the same temperature and in addition, we enabled sampling.

In order to consider 3 candidates and then choose the best candidate, the method `generate_repair` has been adjusted as it follows:

1. created a loop that goes from 0 to 2 in order to include all 3 candidates
2. for each iteration, we are calling the LLM and the fixed code is extracted
3. using the fixed code and the testcases, we find out how many tests passed with the proposed code solution
4. if all tests pass, then we compare the edit distance and select the repair which has the lowest distance
5. in case not all tests passed, I tried 2 different approaches
   - ranking first fixed proposal according to the number of tests passed: first of all, I am checking if the current repair has more tests passed. In this case, I am comparing the edit distance and if it is smaller than the current value, then I am keeping a copy of the repair (commented as `Attempt 1` in code)
   - checking only if at least one test passed: in this case, I am comparing directly with the distance and keeping the repair with the smaller edit distance (commented as `Attempt 2` in code)
6. at the end, the best repair is returned. If the best repair turns out to be None or an empty string (since no code fixed was found), then an empty string is returned signifying that the LLM was not able to find a solution for the buggy problem

GPT-4o-mini (method 1 [5])

Problem 1	Correct	Distance
Program 1	True	4
Program 2	False	−1
Program 3	True	5
Program 4	True	14
Program 5	True	13

Problem 3	Correct	Distance
Program 1	True	7
Program 2	True	19
Program 3	True	11
Program 4	True	36
Program 5	True	10

Problem 2	Correct	Distance
Program 1	True	1
Program 2	True	63
Program 3	True	16
Program 4	False	−1
Program 5	True	54

Problem 4	Correct	Distance
Program 1	True	53
Program 2	True	5
Program 3	True	5
Program 4	True	27
Program 5	True	2

Problem 5	Correct	Distance
Program 1	True	1
Program 2	True	12
Program 3	True	4
Program 4	True	2
Program 5	True	35

Averages	RPass (%)	REdit
Problem 1	80	9
Problem 2	80	33.5
Problem 3	100	16.6
Problem 4	100	18.4
Problem 5	100	10.8
Overall	92	17.347826086956523

7 Implementation Assignment: I6

The introductory description from section I5 (6) applies here as well.

Phi-3-mini (method 1 [5])

Problem 1	Correct	Distance
Program 1	False	−1
Program 2	False	−1
Program 3	True	5
Program 4	False	−1
Program 5	False	−1

Problem 3	Correct	Distance
Program 1	False	−1
Program 2	False	−1
Program 3	False	−1
Program 4	False	−1
Program 5	True	10

Problem 2	Correct	Distance
Program 1	False	−1
Program 2	True	54
Program 3	False	−1
Program 4	False	−1
Program 5	False	−1

Problem 4	Correct	Distance
Program 1	True	3
Program 2	True	4
Program 3	True	5
Program 4	True	91
Program 5	False	−1

Problem 5	Correct	Distance
Program 1	True	30
Program 2	False	−1
Program 3	True	4
Program 4	False	−1
Program 5	False	−1

Averages	RPass (%)	REdit
Problem 1	20	5
Problem 2	20	54
Problem 3	20	10
Problem 4	80	25.75
Problem 5	40	17
Overall	36	22.88888888888889

Phi-3-mini (method 2 [5])

The following sets of results is a consequence of `Attempt 2` from code.

Problem 1	Correct	Distance
Program 1	True	16
Program 2	False	−1
Program 3	True	5
Program 4	False	−1
Program 5	False	−1

Problem 3	Correct	Distance
Program 1	False	−1
Program 2	False	−1
Program 3	False	−1
Program 4	False	−1
Program 5	True	10

Problem 2	Correct	Distance
Program 1	False	−1
Program 2	False	−1
Program 3	True	24
Program 4	False	−1
Program 5	False	−1

Problem 4	Correct	Distance
Program 1	True	51
Program 2	True	4
Program 3	True	5
Program 4	False	−1
Program 5	False	−1

Problem 5	Correct	Distance
Program 1	False	-1
Program 2	False	-1
Program 3	False	-1
Program 4	False	-1
Program 5	False	21

Averages	RPass (%)	REdit
Problem 1	40	10.5
Problem 2	20	24
Problem 3	20	10
Problem 4	60	20
Problem 5	20	21
Overall	32	17

Looking at both attempts with Phi-3-mini, it seems the second attempt made the distance smaller, indicating that the repairs are better. On the other hand, RPass is smaller indicating the number of repairs successfully identified has decreased.

8 Implementation Assignment: I7

Analysis completed with **GPT-4o-mini** model using as guidance ([1]).

Problem 1	HCorrect	HInformative	HConceal	HComprehensible	HGood
Program 1 hint	1	1	0	1	0
Program 2 hint	0	0	0	0	0
Program 3 hint	1	1	1	1	1
Program 4 hint	1	1	1	1	1
Program 5 hint	1	1	1	1	1

The hint for *Program 1* is correct and informative. Unfortunately, it gives too much information i.e. it returns the solution straight away.
The hint from *Program 2* describes a problem that actually never exists in the code. It says the code does not work correctly in case `x` is less than the first element, but this is covered in the right way already with the existing code.
The hint for *Program 3* is correct when evaluating all four attributes because it tells the learner the code exists prematurely the loop but it does not tell the reason, hence making the student think about any issues.

HGood = 60% in this case (**Problem 1**). It seems that for most of the cases, the hint was on point and it did not reveal the solution while commenting about the root cause(s).

Problem 2	HCorrect	HInformative	HConceal	HComprehensible	HGood
Program 1 hint	1	1	0	1	0
Program 2 hint	0	0	0	0	0
Program 3 hint	1	1	1	1	1
Program 4 hint	1	1	1	1	1
Program 5 hint	1	1	1	1	1

In case of *Program 2*, the hint is incorrect because it says the program should return True `for cases where the count of occurrences is greater than one instead of equal to one.` This is incorrect as the two statements from the phrase contradict themselves. In the right way, they should be written in the `if-statement` with an `or` in between.
Hint for *Program 4* correctly describes and gives clues to help the learner solve the issue. It does not solve all bugs but one of them at least is resolved like this.

HGood = 60% in case of **Problem 2**.

Problem 3	HCorrect	HInformative	HConceal	HComprehensible	HGood
Program 1 hint	0	0	0	0	0
Program 2 hint	1	1	1	1	1
Program 3 hint	1	1	0	1	0
Program 4 hint	1	1	0	1	0
Program 5 hint	1	1	0	1	0

The hint for *Program 3* is correct, but it is too explicit as it states directly `You should not sort the list before removing duplicates, as this alters the original order of elements`. From this, the learner would directly know that calling the `sort` method is incorrect. Thus, HConceal = 0.
In case of *Program 4*, the hint is too explicit as it tells the learner that modifying the list while iterating might skip elements, which essentially, is the problem in the code (besides the typo with the name of the list).
The hint is correct but it gives too much information as it specifies which function could be used for correct concatenation, hence HConceal = 0. Hint: `You need to use the ‘append‘ method to add elements to ‘new_lst‘, instead of trying to concatenate them directly`.

For **Problem 3**, HGood = 20%. Low score, but this is mostly because the prompts give a possible solution straight away.

Problem 4	HCorrect	HInformative	HConceal	HComprehensible	HGood
Program 1 hint	0	0	0	0	0
Program 2 hint	1	1	1	1	1
Program 3 hint	1	1	0	1	0
Program 4 hint	1	1	0	1	0
Program 5 hint	1	1	1	1	1

The hint for *Program 1* says the `while` loop is not efficient, but this does not have anything to do with the real code problems.
The hint for *Program 2* indicates that the program is returning `None` and says the result should not be returned directly. This is almost the solution, but I consider HConceal = 1 because the learner still has to reason about a possible solution and understand why the direct `return` is not good in here.
In case of *Program 3*, the hint says `You are trying to compare the age of the largest tuple using the entire tuple instead of just the age value`. The second part of the hint is the solution directly.
For *Program 4*, the hint directly says integer division should be use i.e. `You need to use integer division when calculating the midpoint of the list in the ‘sort_age‘ function`. Thus, HConceal = 0.
The hint for *Program 5* solves at least one bug. This indicates all four attributes are 1, thus HGood = 1.

In case of **Problem 4**, HGood = 40%.

Problem 5	HCorrect	HInformative	HConceal	HComprehensible	HGood
Program 1 hint	1	1	0	1	0
Program 2 hint	1	1	0	1	0
Program 3 hint	1	1	1	1	1
Program 4 hint	1	1	0	1	0
Program 5 hint	1	1	1	1	1

The hint for *Program 1* is giving the solution straight away i.e. `Change the condition in your while loop from ‘k >= 0‘ to ‘k > 0‘ to prevent the loop from running one extra time`. Thus, HConceal = 0.
Again, like in the first program, the hint for *Program 2* is giving the solution straight away: `You need to compare the integers directly in your for loop instead of using indexing with ‘i[1]‘`.
Since the hint for *Program 4* is saying `You should move the ‘lst.remove(largest)‘ statement outside of the inner loop to ensure you’re only removing the largest element after finding it in the entire list`, HConceal = 0. In this case, the hint solves one bug. Unfortunately, gives the solution straight away.

HGood for **Problem 5** is HGood = 40%.

Overall for I7, HGood = 44%.

9 Implementation Assignment: I8

Analysis completed with **Phi-3-mini** model using as guidance ([1]).

Problem 1	HCorrect	HInformative	HConceal	HComprehensible	HGood
Program 1 hint	1	1	0	1	0
Program 2 hint	0	0	0	0	0
Program 3 hint	0	0	0	0	0
Program 4 hint	0	0	0	0	0
Program 5 hint	0	1	0	1	0

In case of *Program 1*, the hint is correct. HConceal = 0 because the explanation is included in the hint section and the explanation tells the learner directly what should be changed in order for the bug to be solved.
For *Program 2*, as in case of GPT-4o-mini, the hint is not correct because it refers to a problem that does not actually exist in the code.
The hint for *Program 3* is `The function returns the index immediately after the first element that is greater than x, which does not align with the problem’s requirement to insert x before any existing occurrences.` This is not true because the function returns the index in all cases where the element of the array is equal or greater than `x` and not the other way around.
The hint is *Check the logic in the loop where you compare x with each element in seq*. Again, it refers to a case that is actually working correct. The real problems lies in the way the check for empty sequence is performed and not in any comparison.
When it comes to *Program 5*, I will consider HInformative = 1 and HComprehensible = 1 because although the hint is messy and contains a lot of redundant information, the last part could help the learner understand the root cause of the bug. Last part of the hint is `The logic needs to account for the fact that if ‘x‘ ... if it’s greater than all elements, it should be placed at the end.`

HGood = 0% (**Problem 1**). Super low percentage because the hints mainly described problems that are not related to the buggy code.

Problem 2	HCorrect	HInformative	HConceal	HComprehensible	HGood
Program 1 hint	0	0	0	0	0
Program 2 hint	1	1	1	1	1
Program 3 hint	0	0	0	0	0
Program 4 hint	0	0	0	0	0
Program 5 hint	1	1	1	1	1

The hint for *Program 1* says that the problem in the code is sending the whole `possible_birthdays` array to the `unique_day` function which is not true. The problem is the typo in the `unique_day` method.
Hint for *Program 3* says the initialization of variable `tf` is incorrect which is wrong. Even the place in the code is wrong (the hint is referring to the wrong method). The problem lies in the other 2 methods.
In case of *Program 4*, the problem is not coming from using a tuple like the hint is saying. It is coming from the `if` logic and from missing implementation for two other methods.

HGood = 40% in this case (**Problem 2**).

Problem 3	HCorrect	HInformative	HConceal	HComprehensible	HGood
Program 1 hint	1	1	1	1	1
Program 2 hint	1	1	1	1	1
Program 3 hint	0	0	0	0	0
Program 4 hint	1	1	0	1	0
Program 5 hint	1	1	0	0	0

The hint for *Program 2* partially solves the issues in the code. Thus, I will consider the hint as correct.
There is not hint or explanation generated for *Program 3*.
For *Program 4*, as in case of the GPT-4o-mini model, the hint is directly saying that modifying the list while iterating might skip elements. Thus, HConceal = 0.
The hint is correct and informative for *Program 5*. However, it gives straight away a solution to the learner i.e. `The correct method to add an element to a list is by using the ‘.append()‘ method`. Though, I consider this information redundant because the hint says this append should be done on the else branch which is incorrect: `which the student should have used in the else clause`.

For **Problem 3**, HGood = 40%.

Problem 4	HCorrect	HInformative	HConceal	HComprehensible	HGood
Program 1 hint	0	0	0	0	0
Program 2 hint	1	1	0	1	0
Program 3 hint	1	1	1	1	1
Program 4 hint	0	0	0	0	0
Program 5 hint	0	0	0	0	0

The hint for *Program 1* says the code incorrectly appends the smallest element at the end of the list whereas the code incorrectly selects the largest element. Hint information is redundant and not

related to the actual problem.
Hint for *Program 2* is correct and informative. However, it says `The sort method in Python does not return a value; it sorts the list in place. Use the sorted function instead` which basically gives the solution straight away. Thus, HConceal = 0.
The hint for *Program 3* says the largest element is not removed correctly. I consider this information correct, informative and not redundant because the learner has to rethink about the whole procedure of finding and removing the largest element. This will help in solving at least one bug.
In case of *Program 4*, the hint says `The student's merge function does not handle the case where one list is empty, which could lead to an error when trying to pop from an empty list`. This is incorrect because if one list is empty, then the `while` loop is skipped and no error is shown.
The generated hint for *Program 5* points to a problem that actually does not exist in the code. The hint says the `old` variable is incorrectly updated, which is not true.

HGood = 20% for *Problem 5*.

Problem 5	HCorrect	HInformative	HConceal	HComprehensible	HGood
Program 1 hint	0	0	0	0	0
Program 2 hint	1	1	1	1	1
Program 3 hint	0	0	0	0	0
Program 4 hint	0	0	0	0	0
Program 5 hint	0	0	0	0	0

Information is redundant in case of *Program 1* because it says code is not working correctly for cases when the input list has duplicate entries. The actual problem in the code is that the outputted list contains more elements than expected.
In case of *Program 2*, the given information is correct. It is saying the code will throw a TypeError, but this is the same answer any Python interpreter would show. Thus, according to all four attributes, HGood = 1.
The hint for *Program 3* is saying the code is incorrectly checking if the input list has any elements. This information is incorrect and redundant because it needs to be checked whether the list that is created at this moment has enough elements (according to k value) or not.
Neither hint nor explanation were generated for *Program 4*.
The information from the hint for *Program 5* is not related to the actual (real) bug. Thus, information is not relevant for us.

HGood for **Problem 5** is HGood = 20%.

Overall, HGood = 24%.

10 Implementation Assignment: I9

Since I modified the code more than it was written in the PDF instructions, I will first describe all the changes that have been made and then the results I have got.

10.1 Code changes

The following changes have been applied:

- in `project_part1_evaluate.py`, when the selected mode is `hint`, I added in the dictionary `program_results` a new field for explanation i.e. `program_results["explanation"] = explanation`
- the explanation along with the hint is unpacked right before the results are added in the dictionary. Now, the method `get_hint` is returning the explanation as well.

- explanation is extracted from the method `generate_hint` which is in `project_part1_hint.py`. In this file, I created a new function similar to `extract_hint`, its name being `extract_explanation` and having as parameter the response generated by the LLM
- the function `extract_explanation` works in the following way: I defined the 2 tags used for start and end of explanation i.e. `[EXPLANATION]` and `[/EXPLANATION]`. Then, I am searching for the position of start tag. If no start tag is found, then I return directly empty string as this means there is no explanation provided in the whole response from LLM. If start tag is found, then I keep its position in a variable and continue searching for the end tag. If no end tag is found, then the position for the end tag is attributed the length of the response i.e. end of the answer provided by the LLM. Afterwards, I am cropping from the response the part that symbolizes only the explanation according to my 2 positions: position of start tag and position of end tag.

10.2 Conclusions. Limitations.

I tried different approaches. Firstly, before starting with the analysis, it is important to clarify a couple of aspects. The settings of the models for generating hints stayed the same as in exercises I5 and I6 i.e. for GPT, I used temperature = 0.7. For Phi, same temperature and sampling enabled. These are the specific settings introduced by the previous exercises.
Furthermore, since the exercise is requesting to find the limitations and try different approaches, I can split my tries in two categories: first category is represented by the prompts using only the HINT tag while the second category is represented by the prompts using both HINT and EXPLANATION tags. First of all, I will address the cases where only the HINT tag was used. For this specific case, I tried the following prompts:

1. Based on the provided repair, provide a concise single-sentence hint to the student about one bug in the student's buggy code. Along the hint, please provide an explanation written in a step-by-step format. Output your hint and explanation between [HINT] and [/HINT].
2. Based on the provided repair, provide a concise single-sentence hint to the student about one bug in the student's buggy code. Along the hint, please provide an explanation. The learner should read the explanation and understand what and why is wrong in his code. When providing the explanation, please do not give the solution directly. The explanation must act as a Chain-of-Thought. Output your hint and explanation between [HINT] and [/HINT].
3. Based on the provided repair, provide a concise single-sentence hint to the student about one bug in the student's buggy code. Along the hint, please provide an explanation. The learner should read the explanation and understand what and why is wrong in his code. When providing the explanation, please do not describe the solution of the code. The role of the explanation is to understand the issue(s) in the buggy code. Output your hint and explanation between [HINT] and [/HINT].

Now, I will go through each prompt and provide an analysis concerning the results:

1. In case of the first prompt, most of the hints are describing in detail the problem(s) in the code. Moreover, a proper solution to solve issue(s) is provided. This might come from my formulation that includes the collocation "step-by-step". Equally, in some cases, the hint is explaining what kind of code is not accepted at all in the Python e.g. `they are trying to concatenate an integer to a list, which is not allowed in Python.`
2. Although the second prompt is not very clear in terms of meaning, I tried it anyways since I was curios about the quality of the hints. It seems that this time, the generated hints either include the solution directly or present (by introducing inline code in the hint) where the issue lies exactly. Overall, the quality of the hints in terms of information is better than before, but what is not good again, is the fact that the solution is served right away for the user.
3. The third prompt has the most complex answers. Not only does it sometimes describe the solution, but also gives many details about the bugs. Essentially, if I am putting side by side the hints from the second prompt vs. the hints from the third prompt, they are kind of the same, but in case of the third prompt, the wording is more complex and much clear.

In continuation, I will analyse the case where the tag EXPLANATION has been added. The prompts are more or less the same with the ones from above, but for clarity in explanation, I will include them below:

1. Based on the provided repair, provide a concise single-sentence hint to the student about one bug in the student's buggy code. Along the hint, please provide an explanation. The learner should read the explanation and understand what and why is wrong in his code. When providing the explanation, please do not give the solution directly. The explanation must act as a Chain-of-Thought. Output your hint between [HINT] and [/HINT] and explanation between [EXPLANATION] and [/EXPLANATION].
2. Based on the provided repair, provide a concise single-sentence hint to the student about one bug in the student's buggy code. Along the hint, please provide an explanation. The learner should read the explanation and understand what and why is wrong in his code. When providing the explanation, please do not describe the solution of the code. The role of the explanation is to understand the issue(s) in the buggy code. Output your hint between [HINT] and [/HINT]. Output your explanation between [EXPLANATION] and [/EXPLANATION].

Straight away, I saw an interesting behavior. I tried printing the whole generated response before splitting it by the 2 tags. In some very rare cases, either only the hint or only the explanation was generated. Also, when running using the Phi-3-mini, for 2 programs out of 25, there was no hint or explanation generated. Equally, sometimes, the hint was the same as the explanation. Thus, in the output file, the results might have looked like in the following case:

- "hint": "[EXPLANATION]\n\nThe student's code incorrectly returns the index of the first element that is less than the target value, which does not align with the problem's requirement to insert the value in the correct sorted position."
- "explanation": "The student's code incorrectly returns the index of the first element that is less than the target value, which does not align with the problem's requirement to insert the value in the correct sorted position."

The explanations provided by GPT-4o-mini model have more meaning and provide a deeper understanding of the problem than the Phi-3-mini model. One more important fact to be mentioned is the following: while using the first prompt, the part "Output your hint between [HINT] and [/HINT] and explanation between [EXPLANATION] and [/EXPLANATION]" sometimes confused the models. Either the hint or the explanation were empty in the final output file and all the provided information was not separated between the two tags, but written in one tag only.

In terms of limitations, there is definitely a limitation when requesting the hint and the explanation both between one tag. In this case, the LLM generates an answer, but this answer might be very short and might be useful as a hint only. When I first tried it with one tag, most of the answers were kept short. Printing the whole LLM response proved that besides the hint displayed in the output file, there were no other words added by the model (neither before the start tag nor after the end tag of the hint). The flow proved to be more favorable when a tag used specifically for explanation was added. Although it was quite difficult for the model to not give the solution straight away when completing the explanation part, the results were looking much better as the output was more informative than before.

Acknowledgements

References

[1] Alkis Gotovos Nachiket Kotalwar and Adish Singla. *Hints-In-Browser: Benchmarking Language Models for Programming Feedback Generation*. In NeurIPS (*Datasets and BenchmarksTrack*), 2024. Paper Link: https://openreview.net/pdf?id=JRMSC08gSF.