UDC 004.42:37.091.3:004.8
DOI https://doi.org/10.32782/2521-6643-2025-2-70.14

Dobrovolskyi H. A., Candidate of Technical Sciences,
Associate Professor at the Department of Computer Science,
Zaporizhzhia National University
ORCID: 0000-0001-5742-104X

Sgadov S. A., Senior Lecturer at the Department of Computer Systems and Networks
National University Zaporizhzhia Polytechnic
ORCID: 0000-0002-7994-6530

FORMULATION OF THE INTENTIONALLY GAPPED PROBLEMS FOR AI-LITERATE LEARNING IN HIGHER EDUCATION

This case study proposes and evaluates a pedagogy for integrating generative AI into software engineering students' work by turning known model limitations–hallucination, confident completion, and gap-repair–into structured learning opportunities. Building on evidence that large language models tend to produce fluent but unverifiable answers, the method reframes assessment as a sequence of activities that require higher-order thinking, process transparency, and oral defense. An instructor begins with a «seed» programming task and either diagnoses its ambiguities or deliberately introduces controlled gaps to create a hands-on problem that is underspecified by design. Students then pursue one of two routes: consult an LLM to obtain an initial solution or independently analyze the prompt to surface missing constraints. In both cases, they must request clarifications, revise the specification, and substantiate design choices through tests and justifications. Assessment weights the quality of clarification requests, identification and explanation of ambiguities, correctness and coherence of the final solution, and conceptual mastery demonstrated in a brief viva that probes reasoning rather than code mechanics.

An example–the classic factorial table in C#–illustrates how underspecification (range, data type limits, error policy, output format, and purity) predictably elicits plausible yet defective LLM outputs (e.g., sentinel values that conflate error modes, mixed I/O and computation, unreliable overflow checks). Contrasting this seed with a fully specified "corrected" version shows that explicit constraints improve uniform grading but reduce opportunities to critique AI assumptions. By contrast, an intentionally gapped version most effectively compels students to interrogate model completions, formulate testable requirements, and defend choices orally. The approach thereby promotes responsible AI use (verification over deference), strengthens academic integrity (process artifacts and viva-based authorship evidence), and targets upper levels of Bloom's taxonomy (analysis, evaluation, and creation). The paper concludes that gapped, critique-centered tasks, coupled with process-oriented submissions and oral assessment, offer a scalable, human-centered pathway for AI-literate computing education, and motivates empirical studies comparing learning and integrity outcomes against conventional, fully specified assignments.

Key words: *generative AI, intentionally gapped problems, software engineering education, academic integrity, higher-order thinking, AI literacy.*

Добровольский Г. А., Сгадов С. О. Формулювання завдань із навмисними пастками для штучного інтелекту з метою використанням у вищій освіті

У цьому дослідженні пропонується та оцінюється педагогічні аспекти інтеграції генеративного штучного інтелекту в курс роботу з програмної інженерії шляхом перетворення відомих обмежень моделі – галюцинацій, впевненого завершення та виправлення прогалин – на структуровані навчальні можливості. Спираючись на докази того, що великі мовні моделі, як правило, дають плавні, але неперевірені відповіді, метод переосмислює оцінювання як послідовність дій, що вимагають мислення вищого порядку, прозорості процесу та усного захисту. Викладач починає з «початкового» програмного завдання та або діагностує його неоднозначності, або навмисно вводить контрольовані прогалини, щоб створити практичну проблему, яка недостатньо специфікована за дизайном. Потім студенти обирають один із двох шляхів: звернутися до магістра права (LLM), щоб отримати початкове рішення, або самостійно проаналізувати запит, щоб виявити відсутні обмеження. В обох випадках вони повинні запросити роз'яснення, переглянути специфікацію та обґрунтувати вибір дизайну за допомогою тестів та обґрунтувань. Оцінювання враховує якість запитів на роз'яснення, виявлення та пояснення неоднозначностей, правильність та узгодженість кінцевого рішення, та концептуальне володіння, продемонстроване в короткому опитуванні, яке досліджує міркування, а не механіку коду. Приклад – класична таблиця факторіалів в C# – ілюструє, як недостатня специфікація (діапазон, обмеження типів даних, політика помилок, формат виводу та чистота) передбачувано викликає правдоподібні, але дефектні результати LLM (наприклад, контрольні значення, що змішують режими помилок, змішаний ввід/вивід та обчислення, ненадійні перевірки переповнення). Порівняння цього початкового значення з повністю визначеною «виправленою» версією

показує, що явні обмеження покращують рівномірне оцінювання, але зменшують можливості для критики припущень ШІ. Навпаки, навмисно пропущена версія найефективніше змушує студентів досліджувати завершення моделі, формулювати вимоги, що перевіряються, та захищати вибір усно. Таким чином, цей підхід сприяє відповідальному використанню ШІ (перевірка, а не повага), зміцнює академічну доброчесність (артефакти процесу та докази авторства на основі viva) та орієнтується на верхні рівні таксономії Блума (аналіз, оцінка та створення). У статті робиться висновок, що завдання з пробілами, орієнтовані на критику, у поєднанні з орієнтованими на процес поданнями робіт та усним оцінюванням пропонують масштабований, орієнтований на людину шлях для навчання комп'ютерній грамотності на базі ШІ та мотивують емпіричні дослідження, що порівнюють результати навчання та доброчесності зі звичайними, повністю визначеними завданнями.

Ключові слова: *генеративний штучний інтелект; навмисно створені пастки, освіта в галузі програмної інженерії, академічна доброчесність, мислення вищого порядку, грамотність у сфері штучного інтелекту, усне оцінювання.*

Formulation of the problem. The proliferation of tools like ChatGPT has instigated a paradigm shift in academic practices, challenging traditional notions of authorship, learning, and assessment [21].

Student adoption of GenAI has accelerated dramatically. A 2025 study found that almost all students (92%) now use AI in some form, with 88% having used it for assessments [10]. The most common uses are for explaining concepts, summarizing articles, and brainstorming ideas [10]. However, a notable percentage of students (18%) admit to including AI-generated text directly in their work [1]. This widespread use often stems from a desire to save time and improve work quality.

Despite this high rate of adoption, a significant gap exists in institutional support. While 80% of students agree their institution has a clear AI policy, only 36% have received support to develop AI skills [10]. Moreover, both students and faculty express concerns about the risk of academic misconduct and the potential for AI "hallucinations" or biased results ([1], [11]).

The research objective of this paper is to design modification of tasks and/or solution acceptance rules used while hands-on by computer science and software engineering students, encouraging the correct AI usage. That assumes a redesign of both problems and requirements to their solutions. Instead of strict prohibition of AI we propose to shift accents to fundamental skills like system design, research, complementation, and verification supported with AI.

Analysis of recent research and publications.

The Spectrum of Institutional Responses to AI usage. The response from higher education institutions (HEIs) has been varied, ranging from outright prohibition to full-scale integration. This diversity in response is indicative of the complexity surrounding GenAI, which simultaneously offers powerful new tools for research and learning while posing significant risks to academic integrity.

HEIs have adopted a range of policy stances, often categorized along a continuum from restrictive to permissive [1]. The most common approaches include:

- **Prohibition:** This stance bans AI use for all assessed work, often relying on detection tools to enforce policies. While this approach appears to safeguard academic integrity, it can lead to a «cat-and-mouse» dynamic and fail to prepare students for a professional world where AI proficiency is increasingly valued [2]:
- **Permissive with Disclosure:** This model allows or even encourages GenAI use for certain tasks, provided students transparently disclose their usage. This approach fosters AI literacy and ethical awareness but relies on the student's honesty, which can be difficult to verify.
- **Contextual or Two-Lane Approach:** This is an increasingly popular model. It distinguishes between «secure» (proctored) assessments where AI is prohibited and "open" (unsupervised) assessments where it is permitted with proper acknowledgment. This pragmatic approach tailors policy to the specific learning objectives of a task [21].

Acceptable Usage of Generative AI in Graduate-Level Computer Science Education. General considerations. Generative artificial intelligence (GenAI) tools such as ChatGPT, GitHub Copilot, and DALL·E have become increasingly embedded in higher education, offering novel ways to support learning, research, and creativity. At the core is the principle that AI should help students think, not think for them [3-5]. Students must retain responsibility for the final outputs and ensure their submissions reflect their understanding. AI tools should be used to supplement learning. For instance, students might use ChatGPT to brainstorm coding strategies or clarify algorithmic concepts. However, they must critically assess AI-generated suggestions and refine them based on personal judgment [6]. AI-generated content must always be verified. GenAI tools are prone to hallucinations and inaccuracies. For computer science students, this includes debugging AI-suggested code and validating theoretical explanations against scholarly or technical sources [6]. Ethical use requires academic honesty. Unauthorized use of GenAI to generate entire solutions or codebases may constitute academic misconduct [13]. Even when permitted, copying content verbatim without attribution is inappropriate and may be considered plagiarism. Transparency is another core principle. Disclosing AI assistance–either in an appendix, footnote, or via prompt logs–enhances academic integrity [19]. Recent guidelines emphasize that concealing AI involvement may breach research integrity standards [3]. Privacy must also be respected. Students and researchers should avoid uploading confidential or

personal data to public GenAI platforms due to lack of control over stored inputs [6]. Lastly, usage should comply with tool-specific terms of service. Employing GenAI tools to impersonate others, generate fraudulent content, or violate licensing conditions is unethical and often illegal [16].

Promoting Responsible GenAI Using. Leading institutions now view GenAI not as a threat, but as an opportunity to enhance academic engagement–provided appropriate guardrails are in place [14]. Several strategies support the responsible use of GenAI in computer science education. Many universities have issued AI policies that clarify acceptable and unacceptable use. A study of U.S. institutions found that 63% of top research universities encourage GenAI use with conditions (McDonald et al., 2024). Policies often provide examples (e.g., using ChatGPT for idea generation vs. using it to write an entire thesis) and guide faculty in crafting AI-related syllabus statements. AI literacy encompasses technical competence, critical evaluation, and ethical understanding [10]. Workshops and training sessions help students and faculty learn about prompt engineering, output evaluation, and bias mitigation. Some computer science departments now offer courses or modules explicitly focused on effective GenAI use. Instructors who model appropriate GenAI use help demystify these tools. Demonstrating how to critically engage with AI-generated code or text sets expectations and encourages reflective practice [14]. Faculty can also structure assignments that require students to document their AI interactions. Universities are centralizing GenAI resources through guideline portals, helpdesks, and curated tutorials. Erasmus University, for example, provides students with documentation on responsible use, citation methods, and data privacy [6]. Academic integrity offices are revising honor codes to include GenAI-related clauses. Instructors are also integrating AI ethics into the curriculum, covering fairness, transparency, and accountability. These discussions help students appreciate the societal responsibilities of AI developers and users [19]. Educators are exploring ways to design assignments that harness GenAI constructively while maintaining academic rigor. Table 1 summarizes the common problem types that should stimulate human-led solutions instead of direct AI completion (Table 1.).

Table 1

Types of problems that are less likely to be solved with AI alone

Type	Problem name	Problem description	References
Type 1	Higher-Order Thinking Tasks	Assignments that prioritize creativity, synthesis, and critical evaluation are less amenable to direct AI completion.	[5], [12],[17]
Type 2	Process-Oriented Submissions.	By requiring students to submit drafts, AI prompt logs, or reflective statements, instructors can evaluate not just the product, but the learning process.	[15], [7]
Type 3	Assignments Requiring the Use and Critique of AI Outputs	Some educators embed AI use into the assignment structure. Students may be instructed to use ChatGPT to produce an initial code draft and then improve it, documenting their changes and rationale.	[4], [15]
Type 4	Personalized and Contextualized Prompts.	Assignments that incorporate unique datasets, personal experiences, or course-specific material are harder for AI to solve generically.	[4], [5], [17]
Type 5	Oral and Collaborative Assessments.	Presentations, code walkthroughs, and group projects add layers of accountability. Oral components can verify individual understanding, especially in programming-heavy courses.	[4], [12],[23]

In the next sections we will focus on the higher-order thinking tasks and assignments requiring the use and critique of AI outputs, process-oriented submissions, and oral assessments trying to compose appropriate problems.

Integrating Generative AI into Hands-On Curriculum: A Case Study in Software Engineering and Computer Science. Contemporary GenAI tools are prone to hallucinations and inaccuracies. For computer science students, this includes debugging AI-suggested code and validating theoretical explanations against scholarly or technical sources [6]. In their paper, Varshney et al. [22] highlighted the fact that current LLMs' training forces them to produce a fluent answer in all cases. That means, ANN-driven LLMs inherently tend to always respond (using randomness to choose a plausible completion) rather than abstaining, thus frequently hallucinating in uncommon or unknown contexts due to their training on general patterns. Therefore, the main idea of the proposed method is to compose an assessment problem that requires Higher-Order Thinking, Process-Oriented Submissions, the Use and Critique of AI Outputs which are assessed Orally. The problem composition is based on both the LLMs' hallucinations and their tendency to implicitly extend the input problem using most probable patches.

Structured Workflow for AI-Supported Problem Analysis and Solution Development. The proposed instructional framework commences with the formulation of a *seed problem*, which serves as the foundational element for subsequent learning activities. The primary participants in this framework are the *instructor* and the *student*, whose interaction is structured to cultivate analytical reasoning, critical evaluation of artificial intelligence

(AI) outputs, and metacognitive engagement throughout the problem-solving process. Initially, the instructor examines the seed problem to identify any ambiguities, inconsistencies, or gaps that may render the problem underspecified or unsolvable. When the seed problem is overly precise and lacks opportunities for exploration, the instructor deliberately introduces controlled ambiguities to create a *hands-on problem*. This intentional modification transforms the task into an instrument for guided discovery and critical reasoning, encouraging students to engage actively with the problem definition rather than accepting it uncritically.

Upon receiving the hands-on problem, the student may pursue one of two distinct approaches. The first involves submitting the problem to a large language model (LLM) to obtain a proposed solution. The second involves conducting an independent analysis of the problem statement, aimed at identifying conceptual or logical issues. In cases where ambiguities or contradictions are detected, the student is encouraged to seek clarification from the instructor. Given that the instructor introduced the ambiguities intentionally, the clarification process is straightforward and pedagogically productive, enabling the student to develop the capacity to recognize and articulate problem deficiencies.

Following clarification, the student revises the problem formulation accordingly and proceeds either to complete the solution or to identify further issues requiring additional clarification. Alternatively, when the student engages with an LLM-generated solution, the instructor evaluates both the accuracy and the interpretive quality of the response. Because the instructor is aware of the artificially introduced flaws within the problem, this evaluation focuses on the student's understanding of the underlying concepts, the ability to detect inconsistencies, and the critical appraisal of AI-generated outputs.

The assessment process encompasses several dimensions: (a) the presence and quality of clarification requests, (b) the student's ability to identify and explain problem ambiguities, (c) the overall quality and correctness of the final solution, and (d) the student's demonstrated conceptual understanding during oral evaluation. Moreover, comparing LLM-generated solutions for both the initial and the revised hands-on problems provides empirical insight into the operational characteristics of generative models. Notably, discrepancies between these solutions reflect the inherent tendency of LLMs to implicitly extend or "repair" incomplete input by generating statistically probable continuations, even when the initial problem is intentionally unsolvable. This phenomenon underscores the necessity of human oversight and critical reasoning when interpreting AI-generated responses.

The proposed workflow intentionally fosters *higher-order cognitive processes* by compelling students to engage critically with the problem formulation itself. Students are required to ensure that each problem is articulated with precision, completeness, and logical coherence prior to solution development. Every identified gap, ambiguity, or vague formulation serves as a prompt for scholarly dialogue between student and instructor, thereby transforming the learning environment into a collaborative and reflective inquiry process. This interaction not only enhances the instructor's awareness of the student's reasoning trajectory but also reinforces the student's capacity for rigorous analytical thought.

While the *use and critique of AI outputs* are encouraged within this framework, such engagement remains optional rather than prescriptive. The primary educational objective lies in cultivating a deep understanding of both the problem context and the reasoning underlying the solution–whether human-derived or AI-assisted. To ensure cognitive and conceptual mastery, *oral assessment* constitutes an essential component of the workflow. Through structured discussion, the instructor verifies that the student comprehends, justifies, and can articulate each step of the reasoning process associated with the hands-on problem. This final evaluative stage consolidates understanding, promotes academic integrity, and reinforces reflective engagement in AI-assisted problem solving.

Example Hands-on Problem

Seed problem analysis. To illustrate the proposed analytical workflow, let us consider a representative example of problem evaluation conducted with the assistance of a large language model (LLM), in this case, ChatGPT-5. The examined prompt requested the model to *"read a given problem statement and point to possible logical issues."* The problem statement under review was as follows [24]:

Using C#, write a function Factorial and a program that uses this function to output a table of factorials.

At first inspection, the task appears straightforward and achievable: students are expected to define a function named Factorial and to implement a program that repeatedly calls this function to display a table of factorials, typically for a range of integer values (for instance, from 1 to 10). However, upon closer examination, several specification issues become evident, indicating that the problem definition lacks sufficient precision for unambiguous implementation.

The **first issue** concerns the *absence of a defined range* for the factorial table. The task does not specify for which numbers the factorials should be computed–whether the intended range is 1–10, 0–12, or determined by user input. This omission leads to inconsistent interpretations and prevents uniform assessment of student solutions.

The **second issue** relates to *data type and return type ambiguity*. Given that factorial values grow exponentially with increasing n, the choice of numeric data type significantly affects the correctness and reliability of the program. The int data type in C# overflows for $n>12$, while the long type overflows for $n\approx20$. For larger values, the use of BigInteger becomes necessary. Since the problem statement provides no guidance on the appropriate data type, students may select incompatible or insufficient representations, resulting in undefined behavior or arithmetic overflow.

The **third issue** involves the *absence of error-handling requirements*. Specifically, the problem statement does not indicate how the program should behave when n<0. Because the factorial function is undefined for negative integers, the lack of explicit handling instructions leaves students uncertain about whether to raise an exception, return a sentinel value, or simply terminate execution.

The **fourth issue** pertains to the *lack of specification regarding the output format*. The task requires the generation of a "table of factorials" but does not clarify its structure or presentation. It remains unclear whether the table should include headers, alignment, or formatted spacing, nor does the description indicate whether the results should be displayed in a console window, written to a file, or presented in a graphical interface.

The **fifth issue** concerns *ambiguity in implementation requirements*. The problem does not specify whether the factorial function should be implemented iteratively or recursively, leaving the approach to student discretion. While this flexibility can encourage creativity, it also introduces inconsistency in assessment criteria. Moreover, the task omits the expected input data type–whether int, long, or double–and fails to define the preferred output mechanism or type.

In summary, although the problem statement initially appears pedagogically sound, a detailed examination reveals multiple layers of underspecification. That means, even in the current formulation, the seed task can be used as a starting point of the AI-assisted problem solving when supplemented with the list of requirements for both the solution process and result. Sending the seed problem as prompt to ChatGPT we get the following code:

```
using System;
class Program {
  static long Factorial(int n) {
     if (n < 0) {
        Console.WriteLine($"Error: Factorial is not defined for ({n}).");
        return -1;
     }
     if (n == 0 || n == 1) return 1;
     long result = 1;
     for (int i = 2; i <= n; i++) {
        result *= i;
        if (result < 0) {
           Console.WriteLine($"Error: Factorial value overflowed for {n}.");
           return -1;
        }
     }
     return result;
  }
  static void Main() {
     Console.WriteLine("Factorial Table Generator\n");
     int limit;
     Console.Write("Enter a non-negative integer limit (e.g. 15): ");
     string input = Console.ReadLine();
     if (!int.TryParse(input, out limit) || limit < 0) {
        Console.WriteLine(
"Invalid input. Please enter a non-negative integer.");
        return;
     }
     Console.WriteLine("\n------------------------------");
     Console.WriteLine("|   n   |     n! (Factorial)  |");
     Console.WriteLine("------------------------------");
     for (int i = 0; i <= limit; i++) {
        long f = Factorial(i);
        if (f == -1) break; // stop if overflow or invalid value
        Console.WriteLine($"| {i,5} | {f,18} |");
     }
     Console.WriteLine("------------------------------");
  }
}
```

It contains a set of implementation decisions that were not specified in the seed task:

1. The program asks the user to input the upper limit of the range of the factorial table but does not ask for a lower limit.

2. The factorial function uses an iterative method to calculate factorial.

3. The factorial function operates with long data types inside.

4. The 2-column table is shown with Console.WriteLine statements.

5. From the mathematical point of view, usage of the return -1 statement to indicate errors is not acceptable because the Γ(x) (gamma function, the generalized factorial) equals to "-1" for x≈-0.505 and x≈-3.545

Therefore, the provided code does not solve the seed problem because the real requirements can contradict the list above. Asking a student "Why do you decide to …" about any list item can reveal the level of critical thinking.

The program code itself has multiple easy-to-see issues too. For example:

– Overflow detection is unreliable because result < 0 can indicate either negative or positive argument values;

– The Factorial function has side effects writing to the console, returning a sentinel (-1), mixing computation and UI.

– The program should prevent calls for input>20 when using long.

These issues are due to multiple dirty implementations of the Factorial function found in the data used to train LLM.

The seed problem omits range, data type constraints, error handling, output specification, and implementation requirements. These omissions predictably elicit a spectrum of reasonable but divergent interpretations by both students and LLMs. Paradoxically, the underspecification both invites and tests AI usage. When submitted to an LLM, the model will "repair" gaps by assuming defaults (e.g., using long, prompting the user for a limit, iterative implementation). This creates a natural occasion to coach students on trustworthy AI practices: identify implicit assumptions in the model's output, verify claims (e.g., overflow thresholds for long), and separate concerns (pure function vs I/O). However, without explicit instructional scaffolds, many students will accept the first plausible AI output as authoritative. Thus, as a stand-alone assignment, the seed problem does not reliably encourage correct AI usage; as a trigger for a critique-oriented workflow, it is productive.

The problem can support **higher-order tasks & critique of AI outputs** if the course requires students to surface ambiguities and critique LLM responses. Otherwise, it tends to devolve into routine coding. **Process-oriented submissions** are not inherently required; these must be imposed externally. The openness of the task furnishes good interview material (e.g., why treat n < 0 this way; what is the safe upper bound for long?), but this value again depends on added **oral assessment** design.

In summary, the seed task underperforms on all three aims unless embedded in a structure that compels clarification, verification, and oral defense. Its chief strength is as a diagnostic substrate for revealing LLM "coherence-by-completion" and for prompting student-initiated specification repair.

Corrected Seed Problem Statement. To demonstrate the correct problem statement, let's fill-in all gaps mentioned above with the prompt:

Please, read a given problem statement and point to possible logical issues. Rewrite this problem statement to make it logically complete. The problem statement is below:

Using C#, write a function Factorial and a program that uses this function to output a table of factorials.

The rewritten problem became a bit longer but completely clear and undoubtful:

Write a C# program that generates a table of factorials using a separate function named Factorial.

1. Function requirements:

– *Define a function Factorial(int n) that returns the factorial of a non-negative integer n.*

– *If n is negative, the function should return –1 to indicate invalid input.*

– *Use an appropriate data type (long) to avoid overflow for values up to 20!.*

2. Program requirements:

– *In the Main() method, display a table of factorials for all integers from 0 to 20.*

– *Each line of the table should show the number and its factorial in a clear format, for example:*

n n!
0 1
1 1
2 2
3 6
…

– *The program should call the Factorial function for each number and print the results.*

3. Implementation note: You may implement Factorial using either recursion or iteration.

Sending the seed problem as a prompt to ChatGPT we get the following C# implementation that fulfills the revised problem statement:

```
using System;
class Program {
    // Function to calculate the factorial of a non-negative integer
    static long Factorial(int n) {
```

```
        if (n < 0) return -1; // Invalid input
        long result = 1;
        for (int i = 1; i <= n; i++) {
            result *= i;
        }
        return result;
    }
    // Main method to display the table of factorials
    static void Main() {
        Console.WriteLine("n\t n!");
        Console.WriteLine(new string('-', 20));
        for (int i = 0; i <= 20; i++) {
            long fact = Factorial(i);
            Console.WriteLine($"{i}\t {fact}");
        }
    }
}
```

The code has many fewer issues than the solution of the original problem. For example, the program does not ask the user to input a range of independent variable n because the range is set explicitly in the problem statement; the iterative method to calculate factorial is one of the allowed methods; long data types in the Factorial function are acceptable; and table layout and output method are set.

With constraints explicit, an LLM's answer is more likely to be compliant and less likely to hallucinate unspecified features. Students can still use AI, but the opportunity to critique the model's assumptions is diminished; most model errors will be minor (e.g., style or micro-inefficiencies). Accordingly, the corrected problem encourages disciplined implementation more than critical AI literacy. To keep AI critique central, instructors should append a requirement to "submit one LLM-generated version and a human-revised version, highlighting divergences from the specification and justifying changes."

Clear criteria enable rubric-based grading and automated tests, but they also make answers more templated and thus easier to source from AI or peers. Integrity is therefore neutral to slightly at risk unless the assignment explicitly requires (a) a prompt/verification log, (b) small design justifications (e.g., error signaling alternatives), and (c) targeted oral questioning.

The problem now emphasizes application and precision rather than analysis and evaluation. It can still reach **higher-order thinking** if students must argue for or against the chosen error signal (−1 vs. exceptions), justify data-type trade-offs, or propose property-based tests for boundary cases.

The statement itself does not require process artifacts; adding a short requirements-to-tests trace and a reflection on overflow guarantees would align it with the objective.

Because the "trickiness" is reduced, **oral assessment** should target design rationales (e.g., separation of concerns, contracts, complexity), turning a straightforward build into a concept-defense conversation.

The corrected statement optimizes fairness and reliability but, without add-ons, underutilizes AI's pedagogical value. It is ideal when the emphasis is demonstrably correct implementation, less so when the aim is cultivating AI critique.

Hands-on Problem Statement. To demonstrate the correct problem statement, let's fill-in all gaps mentioned above using the prompt:

Add to a given problem statement carefully hidden gaps that make a correct implementation impossible without assumptions. The resulting problem statement should be suited for an exam or test. The problem statement is below:
Using C# write a function Factorial and a program that uses this function to output a table of factorials.

Being fed with this prompt, ChatGPT composed the following hands-on problem statement:

Using C#, write a function Factorial that calculates the factorial of a number.
Then write a program that uses this function to output a table of factorials.
The program should:
– *Call the Factorial function for all required values.*
– *Display results in a clear tabular form.*
– *Ensure correct handling of invalid input values.*
The Factorial function should return an appropriate result for all possible inputs, and the program should print the table up to a suitable limit.

The statement above contains hidden gaps / issues not visible to students but intentional:

1. No input range or data type specified–unclear if factorial should work with int, long, or double, or for negative or non-integer numbers.

2. “All required values” and “suitable limit” are undefined–no explicit range for the table (e.g., 1–10, 0–20, etc.).

3. “Appropriate result for all possible inputs” is impossible, since factorial grows too fast and causes overflow–but no instruction on how to handle overflow or large values.

4. “Invalid input values” are mentioned but not defined–should that include negatives, fractions, strings, or numbers exceeding numeric limits?

5. No output format (columns, alignment, etc.) is defined, yet it must be “tabular” and “clear.”

Careful analysis of the generated hands-on problem reveals the following issues:

1. “Up to a suitable limit” is vague–students or developers don’t know how far to compute.

2. Input domain not clearly defined. The formulation “calculates the factorial of a number” doesn’t specify integer only or non-negative values.

3. No mention of what data type to use for large results.

4. The “ensure correct handling,” doesn’t define what “correct” means.

5. “Display results in a clear tabular form” is subjective.Different implementations may vary; may not match expected output.

6. The problem statement doesn’t specify whether recursion, iteration, or library functions are allowed.

7. For very large numbers (e.g., 30+), long will overflow.

While the original problem contains 5 issues, the list above contains 7 issues, ensuring that the hands-on problem cannot be correctly solved “as-is” without detailing. The issues of logic of the hand-on problem are used as subjects of the oral assessment along with the common question about solution details.

When used as a prompt to ChatGPT the hands-on problem statement generates the specious C# implementation:

```
using System;
class Program{
  // Function to calculate factorial
  static long Factorial(int n) {
    if (n < 0) {
      Console.WriteLine(
        $"Error: Factorial is not defined for negative numbers ({n}).");
      return -1;  // Error code for invalid input
    }
    long result = 1;
    for (int i = 2; i <= n; i++) {
      try {
        checked { result *= i; }
      }
      catch (OverflowException) {
        Console.WriteLine($"Error: Factorial value overflowed for {n}!");
        return -1;  // Error code for overflow
      }
    }
    return result;
  }
  static void Main() {
    Console.WriteLine("Factorial Table Generator");
    Console.WriteLine("=========\n");
    Console.Write("Enter a non-negative integer limit (e.g. 20): ");
    string input = Console.ReadLine();

    if (!int.TryParse(input, out int limit) || limit < 0) {
      Console.WriteLine("Invalid input. Enter a non-negative integer.");
      return;
    }
    Console.WriteLine("\n--------------------------------");
    Console.WriteLine("|   n   |     n! (Factorial)    |");
    Console.WriteLine("----------");
    for (int i = 0; i <= limit; i++) {
      long fact = Factorial(i);
```

```
      if (fact == -1) {
        Console.WriteLine($"|  {i,-4} |     Error             |");
      }
      else {
        Console.WriteLine($"|  {i,-4} |  {fact,-18} |");
      }
    }
    Console.WriteLine("--------------------------------");  }
}
```

The generated code uses checked to detect overflow; the iterative loop handles 0! and 1! naturally, clear messages and simple table make output easy-to-read. However, the solution does not explicitly restrict values of the independent variable: limits of the input variable are not suggested, and the data type is long but could be float or BigInteger. Handling of errors is not correct because the return -1; statement occurs for both negative n and overflow.

For very large numbers (20+), long will overflow, but that may be invalid behavior. Displaying results in a tabular form is still subjective. The solution uses an iterative method, which is completely voluntary and may not match the environment where the code will be used.

Also the code itself has multiple issues. First, Factorial has mixed concerns because it prints to console and returns a magic value (-1). The long data type overflows at 21! so the code should show this limitation while asking for user input. Per-iteration try/catch is slow; it will be better to check limits before the loop.

This formulation is optimal for **teaching responsible AI usage**. Submitting it to an LLM will produce plausible but specious code: mixed concerns, ambiguous error signaling, performance-unaware checks, and overflow pitfalls. Students are thus required to interrogate the model's completions, elicit clarifications, and reconcile contradictions. A structured requirement to (a) document the AI's implicit assumptions, (b) verify them against constraints the student proposes, and (c) iterate the prompt after clarification turns AI from a shortcut into an object of study.

Integrity is strengthened by design. Because each cohort (or even each student) can receive slightly varied ambiguities, turnkey copying is less effective. More importantly, grading can weight (i) the quality of clarification requests, (ii) specification repair, and (iii) the critique of AI outputs–all of which are difficult to counterfeit without understanding. The accompanying oral assessment can then target the very gaps the student identified, producing high-fidelity evidence of authorship and comprehension.

The problem centers **higher-order tasks and critique of AI outputs**: students must surface domain constraints (type limits, domain of definition), craft testable requirements, and critique AI behaviors (overflow detection, error signaling).

The task naturally calls for process artifacts: a clarification log, an assumptions ledger, a mini-requirements document, and a test plan. These align directly with the learning objective and provide auditability.

The embedded ambiguities supply rich, individualized prompts (e.g., "Defend your chosen range and data type; what evidence falsified the alternative?"). **Oral examinations** become shorter, more focused, and more equitable because they work from the student's own documented decisions.

Therefore, the intentionally gapped problem is the strongest vehicle for all three aims. It converts LLM fallibility into a teachable phenomenon and systematically links process evidence with oral verification.

Discussion. The proposed method of problem formulation aims to focus on encouraging the correct AI usage, academic integrity, and the following learning objectives: higher-order thinking tasks requiring the use and critique of AI outputs, process-oriented submissions, and oral assessments trying to compose appropriate problems.

From this point of view, the seed task underperforms on all three aims unless embedded in a structure that compels clarification, verification, and oral defense. Its chief strength is as a diagnostic substrate for revealing LLM "coherence-by-completion" and for prompting student-initiated specification repair. The corrected statement optimizes fairness and reliability but, without add-ons, underutilizes AI's pedagogical value. It is ideal when the emphasis is demonstrably correct implementation, less so when the aim is cultivating AI critique. The intentionally gapped hands-on problem is the strongest vehicle for all three aims. It converts LLM fallibility into a teachable phenomenon and systematically links process evidence with oral verification.

Also, the provided example highlights a few technical observations:

1. Across all versions, insisting that Factorial be pure (no I/O, no sentinel conflation) and that error policy be explicit (exceptions, Result<T,E>, or domain restriction) yields natural, AI-resistant discussion prompts. Embedding this as a required design choice maintains higher-order focus even in the corrected statement.

2. Requiring students to compare LLM outputs on the original versus revised prompts operationalizes AI critique. The seed and gapped versions support this directly; the corrected version can re-introduce it by asking students to "break" the LLM with minimal prompt perturbations and analyze failure modes.

3. All versions benefit from mandated artifacts: a prompt/response log with timestamps, a requirements-to-tests traceability table, and a short "AI reliability note" (claims accepted, claims rejected and evidence). These artifacts transform detection into prevention.

4. Micro-vivas (5–7 minutes) keyed to a student's own clarification ledger are efficient and high-signal. Instructors can rotate through themes: numeric limits and types, error semantics, testing strategy, and refactoring for purity. The gapped version makes this especially authentic; the seed and corrected versions can still use oral checks by focusing on design rationales.

For courses prioritizing AI literacy and integrity, adopt the intentionally gapped hands-on problem as the default, with grading weighted toward clarification quality, AI critique, and test evidence, culminating in a focused oral defense. For skills check-offs or early labs where uniformity matters, use the corrected seed problem, but append process requirements (prompt log, test rationale) and a brief comparative critique of at least one LLM output to retain some higher-order focus. Use the underspecified seed problem chiefly as a diagnostic or as the first stage of a two-stage assignment that culminates in a student-produced corrected specification and validation suite, thereby converting initial ambiguity into a structured learning arc.

Conclusions. The integration of generative AI into higher education is an ongoing, complex process. The initial instinct to ban the technology is giving way to a more nuanced understanding of its potential as a pedagogical tool. The evidence suggests that the most effective institutional responses are those that are proactive, transparent, and focused on empowering the academic community with AI literacy (arXiv, 2025). As GenAI continues to evolve, so must our policies and our curriculum, ensuring that academic integrity is maintained not through prohibition, but through a thoughtful, human-centered approach to education.

The proposed method of problem formulation aims to focus on encouraging the correct AI usage, academic integrity, and the following learning objectives: higher-order thinking tasks requiring the use and critique of AI outputs, process-oriented submissions, and oral assessments trying to compose appropriate problems.

In sum, the gapped formulation best embodies the trio of goals: it requires students to use AI, compels them to critique its outputs, documents their reasoning as process evidence, and verifies understanding orally. The corrected formulation secures reliability and fairness but needs added scaffolds to teach AI critique; the raw seed formulation can be powerful only when embedded in a process-first, viva-verified workflow.

The natural future work can include pedagogical experiments to clarify if students assigned the gapped problem demonstrate higher performance on AI critique, process quality, and oral explanation than students given a corrected, fully specified version.

Bibliography:

1. Anara. *AI in Higher Education Statistics: The Complete 2025 Report.* URL: https://anara.com/blog/ai-in-education-statistics.

2. Association of Pacific Rim Universities (APRU). *Generative AI in Higher Education: Current Practices and Ways Forward.* URL: https://www.apru.org/wp-content/uploads/2025/01/APRU-Generative-AI-in-Higher-Education-Whitepaper_Jan-2025.pdf.

3. arXiv. *Adapting University Policies for Generative AI: Opportunities, Challenges, and Policy Solutions in Higher Education.* URL: https://arxiv.org/html/2506.22231v1.

4. Attewell S. *Exploring AI and Assessment – Avoid, Outrun or Embrace.* URL: https://www.jisc.ac.uk.

5. Carden G., Freeman J. (Eds.). *AI and the Future of Universities (HEPI Report 193)* Oxford: Higher Education Policy Institute, 2025. URL: https://www.hepi.ac.uk.

6. Erasmus University. *Generative AI Usage Guidelines* URL: https://www.eur.nl.

7. Forsyth R. *ChatGPT: What Should Educators Do Next?* Assessment in Higher Education, 2023. URL: https://assessmentinhe.wordpress.com.

8. Transformations in Academic Work and Faculty Perceptions of Artificial Intelligence in Higher Education *Frontiers in Education*. 2025. Vol. 10, Article 1603763. DOI: 10.3389/feduc.2025.1603763. URL: https://www.frontiersin.org/journals/education/articles/10.3389/feduc.2025.1603763/full.

9. Higher Education Policy Institute (HEPI). *Skilling Up for AI: How Higher Education Can Support the Responsible Adoption of Generative AI–* London: HEPI, 2023.

10. Higher Education Policy Institute (HEPI). *Student Generative AI Survey 2025.* URL: https://www.hepi.ac.uk/wp-content/uploads/2025/02/HEPI-Kortext-Student-Generative-AI-Survey-2025.pdf.

11. Hostetter R. та ін. *Student and Faculty Perceptions of Generative Artificial Intelligence in Student Writing.* URL: https://www.researchgate.net/publication/383927117.

12. Jisc. *Does ChatGPT Mean the End of the Essay as an Assessment Tool?* URL: https://www.jisc.ac.uk.

13. KU Leuven. *Responsible Use of Generative AI by Students.* URL: https://www.kuleuven.be.

14. McDonald R., Xu J., Thomas P. *Generative AI in Higher Education: Analysis of Institutional Policies.* URL: https://arxiv.org/abs/2401.00671.

15. Mollick E. R., Mollick L. *New Modes of Learning Enabled by AI Chatbots: Three Methods and Assignments.* URL: https://papers.ssrn.com.

16. OpenAI. *Usage Policies.* 2023. URL: https://openai.com/policies/usage-policies.

17. Phipps L. *Means, Motive, Opportunity: A Composite Narrative About Academic Misconduct* – Jisc National Centre for AI, 2023. URL: https://nationalcentreforai.jiscinvolve.org.

18. Tillmanns T., Salomão Filho A., Rudra S., Weber P., Dawitz J., Wiersma E., Dudenaite D., Reynolds S. Mapping Tomorrow's Teaching and Learning Spaces: A Systematic Review on GenAI in Higher Education. *Trends in Higher Education*. – 2025. Vol. 4, No. 1. DOI: 10.3390/higheredu4010002. URL: https://doi.org/10.3390/higheredu4010002.

19. UNESCO. *Guidance for Generative AI in Education and Research.* URL: https://unesdoc.unesco.org.

20. University of North Carolina. *Graduate School Guidance on AI Tools.* URL: https://gradschool.unc.edu.

21. University of Sydney. *Artificial Intelligence Policy.* 2025. URL: https://www.sydney.edu.au/students/academic-integrity/artificial-intelligence.html.

22. Varshney N., Yao W., Zhang H., Chen J., Yu D. *A Stitch in Time Saves Nine: Detecting and Mitigating Hallucinations of LLMs by Validating Low-Confidence Generation.* URL: https://arxiv.org/abs/2307.03987.

23. Webb M. *[Cited in Jisc, 2023].* URL: https://www.jisc.ac.uk.

24. Сгадов С. О. *Методичні вказівки до виконання лабораторних робіт з дисципліни «Основи технології. NET» для студентів спеціальності 123 Комп'ютерна інженерія усіх форм навчання.* Запоріжжя: Національний університет «Запорізька політехніка», 2025. 40 с.

References:

1. Anara. (2025). *AI in higher education statistics: The complete 2025 report.* Retrieved from: https://anara.com/blog/ai-in-education-statistics

2. Association of Pacific Rim Universities (APRU). (2025). *Generative AI in higher education: Current practices and ways forward.* Retrieved from: https://www.apru.org/wp-content/uploads/2025/01/APRU-Generative-AI-in-Higher-Education-Whitepaper_Jan-2025.pdf

3. arXiv. (2025). *Adapting university policies for generative AI: Opportunities, challenges, and policy solutions in higher education.* Retrieved from: https://arxiv.org/html/2506.22231v1

4. Attewell, S. (2024, April 22). *Exploring AI and assessment – avoid, outrun or embrace.* Retrieved from: https://www.jisc.ac.uk

5. Carden, G., & Freeman, J. (Eds.). (2025). *AI and the future of universities (HEPI Report 193).* Higher Education Policy Institute. Retrieved from: https://www.hepi.ac.uk

6. Erasmus University. (2023). *Generative AI usage guidelines.* Retrieved from: https://www.eur.nl

7. Forsyth, R. (2023, January 5). *ChatGPT: What should educators do next?* Assessment in Higher Education. Retrieved from: https://assessmentinhe.wordpress.com

8. Frontiers. (2025). *Transformations in academic work and faculty perceptions of artificial intelligence in higher education. Frontiers in Education, 10*, Article 1603763. https://doi.org/10.3389/feduc.2025.1603763

9. Higher Education Policy Institute (HEPI). (2023). *Skilling up for AI: How higher education can support the responsible adoption of generative AI.* Higher Education Policy Institute.

10. Higher Education Policy Institute (HEPI). (2025). *Student generative AI survey 2025.* Retrieved from: https://www.hepi.ac.uk/wp-content/uploads/2025/02/HEPI-Kortext-Student-Generative-AI-Survey-2025.pdf

11. Hostetter, R., et al. (2024). *Student and faculty perceptions of generative artificial intelligence in student writing. ResearchGate.* Retrieved from: https://www.researchgate.net/publication/383927117

12. Jisc. (2023, January 10). *Does ChatGPT mean the end of the essay as an assessment tool?* Retrieved from: https://www.jisc.ac.uk

13. KU Leuven. (2023). *Responsible use of generative AI by students.* Retrieved from: https://www.kuleuven.be

14. McDonald, R., Xu, J., & Thomas, P. (2024). *Generative AI in higher education: Analysis of institutional policies. arXiv preprint*, arXiv:2401.00671. https://doi.org/10.48550/arXiv.2401.00671

15. Mollick, E. R., & Mollick, L. (2022). *New modes of learning enabled by AI chatbots: Three methods and assignments.* Retrieved from: https://papers.ssrn.com

16. Open A. I. (2023). *Usage policies.* Retrieved from: https://openai.com/policies/usage-policies

17. Phipps, L. (2023, March 27). *Means, motive, opportunity: A composite narrative about academic misconduct. Jisc National Centre for AI.* Retrieved from: https://nationalcentreforai.jiscinvolve.org

18. Tillmanns, T., Salomão Filho, A., Rudra, S., Weber, P., Dawitz, J., Wiersma, E., Dudenaite, D., & Reynolds, S. (2025). *Mapping tomorrow's teaching and learning spaces: A systematic review on GenAI in higher education. Trends in Higher Education, 4*(1), 2. https://doi.org/10.3390/higheredu4010002

19. UNESCO. (2023). *Guidance for generative AI in education and research.* Retrieved from: https://unesdoc.unesco.org

20. University of North Carolina. (2023). *Graduate school guidance on AI tools.* Retrieved from: https://gradschool.unc.edu

21. University of Sydney. (2025). *Artificial intelligence policy.* Retrieved from: https://www.sydney.edu.au/students/academic-integrity/artificial-intelligence.html

22. Varshney, N., Yao, W., Zhang, H., Chen, J., & Yu, D. (2023). *A stitch in time saves nine: Detecting and mitigating hallucinations of LLMs by validating low-confidence generation. arXiv preprint*, arXiv:2307.03987. https://doi.org/10.48550/arXiv.2307.03987

23. Webb, M. (2023). *[Cited in Jisc, 2023].* Retrieved from: https://www.jisc.ac.uk

24. Сгадов, С. О. (2025). *Metodychni vkazivky do vykonannia laboratornykh robit z dystsypliny «Osnovy tekhnolohii.NET» dlia studentiv spetsialnosti 123 Komp'iuterna inzheneriia usikh form navchannia.* Zaporizhzhia: Natsionalnyi universytet «Zaporizka politekhnika».

Дата надходження статті: 27.10.2025
Дата прийняття статті: 17.11.2025
Опубліковано: 30.12.2025