Exercise - Activity

COMP-321, Fall 2025, Section 001

Original Draft: November **23th**, 2025 (23:59)
Presentation (in class time): November **26th**, December **1st** or December **3th**
Final Submission: December **10th**, 2025 (23:59)

Writing one Kattis problem

Background: In the world of competitive programming, problem design is as intellectually demanding and creatively rewarding as problem solving itself. This assessment invites you to step into the role of a problem setter, a crucial figure in programming contests, by crafting your own original Kattis-style problem. Through this exercise, you will deepen your grasp of algorithmic thinking and computational constraints, while honing your ability to communicate complex technical ideas with clarity and precision.

Important Preparation: Before proceeding with the development of your problem, it is essential that you carefully review the resources provided.

- First, download and thoroughly examine the file named `problemskeleton.zip` available on MyCourses. This archive contains a template that you are expected to use as the foundation for your submission. Make sure to understand its structure and fill in all relevant sections appropriately.
- Second, you must read the official documentation on the Kattis problem package format, which outlines the required components and conventions for problem submissions. This information is crucial for ensuring your problem is compatible with the Kattis system. The documentation is available at the following link: https://www.kattis.com/problem-package-format/.
- Third, please read the following document, which provides additional guidelines on writing programming contest problems. It was authored by Greg Hamerly and is available at: `https://cs.baylor.edu/~hamerly/courses/4144_22s/writing_problems.txt`

Failure to consult these resources may result in incomplete or invalid submissions. Please take the time to go over both the template and the documentation carefully before you begin.

Submission Requirements: For this assessment, you are required to submit the following materials to MyCourses by the deadline:

Deliverable 1: A description of your original problem ()

For this first deliverable, you are required to submit a LaTeX document that presents a clear, complete, and well-organized description of your original problem. Your submission must include the following components:

- A detailed and precise problem statement.
- Clear input and output specifications.
- At least one sample input/output pair.
- At least one original visual element: either an illustration (a non-essential image that enhances the visual appeal of the problem) **or** a figure (an essential image that helps clarify or explain part of

the problem).

Important: The LaTeX file you need to complete is located in the `problem_statement` folder inside the `problemskeleton` directory. Before you begin, please read the `README.txt` file included in the `problem_statement` folder, as well as the comments within the provided LaTeX template. These contain essential instructions and guidance.

Deliverable 2: One Correct and Efficient Solution ()

For this second deliverable, you are required to implement at least one correct, efficient, well-structured, and well-documented solution in each of the following programming languages: C++, Java, and Python 3. Each solution must be contained in a single file and placed in the `submissions/accepted` folder, located within the `problemskeleton` directory.

Deliverable 3: One Incorrect or Inefficient Solution ()

For this third deliverable, you must implement at least one incorrect or inefficient solution that demonstrates one or more common Kattis errors, such as `Wrong Answer`, `Time Limit Exceeded`, or `Memory Limit Exceeded`. Each solution must be contained in a single file and placed in the corresponding folder inside the `submissions/` folder, located within the `problemskeleton` directory.

Deliverable 4: Test-Case Generator ()

For this deliverable, you must implement a test case generator that produces valid input and output files for your problem (see the next two deliverables for more information). The generator may be written in C++, Java, or Python, and should include both hand-crafted edge cases and randomly generated test cases to ensure comprehensive coverage. Place your generator inside the folder `test_case_generator` folder, located within the `problemskeleton` directory.

Deliverable 5: Input Files ()

For the fifth deliverable, you are required to submit input test cases in two categories:

- **Sample Inputs:** At least **3** files
- **Secret Inputs:** At least **20** files

Each input file must use the `.in` suffix, and the combined size of all input files must be less than **500 KB**. Please organize your files into the appropriate subfolders within the data directory:

- `sample/`
- `secret/`

Additionally, you must include a file named `Description.txt` that provides a brief explanation of what each input test case is designed to evaluate. Examples of test case types include:

- Edge cases
- Large input / TLE (Time Limit Exceeded) cases
- Complicated random cases
- Normal random cases

Please ensure that your descriptions are clear and concise to help us understand the purpose of each test case.

Deliverable 6: Output Files ()

For this deliverable, you must submit both sample and secret output files with the `.ans` extension. Each output file must correspond to an input file with the same base name (i.e., matching `.in` and `.ans` file pairs). These file pairs will be used to test submitted solutions. Therefore, it is essential that the number and names of the input and output files match exactly. Additionally, the total combined size of all input files must not exceed 500 KB.

Deliverable 7: Input Validator ()

For this deliverable, you must create an input format validator that returns exit code `42` when the input

is correctly formatted, and a different exit code when formatting issues are detected. Your validator must be submitted in the `input_format_validators` folder. Please notice that this validator should test input format as described in the problem text, and should match this description exactly.

Please ensure you read the `README.txt` file located in that folder, as it contains important additional instructions and guidelines.

Deliverable 8: Presentation ()

You are required to prepare a slide deck to support a 20-minute presentation of your problem. The presentation should be clear, concise, and visually engaging, please avoid slides that are overly text-heavy. Your slides should effectively communicate the key aspects of your problem, including its motivation, structure, input/output format, and solution strategies. At a minimum, include one slide summarizing each deliverable. Support your presentation with experiments and visualizations based on your problem and solutions.

Below are example questions you may be asked during your presentation. Your slides should help you address these topics/questions:

1. **Deliverable 1: Problem Statement**
   - What inspired you to design this problem?
   - How did you decide between using an illustration or a figure?
   - How was the visual element (illustration or figure) created?
   - What is the significance of the problem title?
   - How did you determine the input constraints?
   - Why did you choose the specific sample input/output examples?
   - What is the estimated difficulty level of your problem (e.g., on the Kattis scale)?
2. **Deliverable 2: Correct and Efficient Solution**
   - What is the time and space complexity of your solution?
   - Could a more efficient solution be implemented?
   - Could a less efficient solution still pass the test cases?
   - Which of the three programming languages performed best?
   - Which language was most suitable for implementing your solution, and why?
   - What are the best-case and worst-case scenarios for your solution?
3. **Deliverable 3: Incorrect or Inefficient Solution**
   - What is the time and space complexity of this solution?
   - Why is this solution incorrect or inefficient?
   - How many test cases does it fail, and why?
   - How does its performance compare to the correct solution?
   - Does the solution fail in all three programming languages?
4. **Deliverables 5 and 6: Input and Output Files**
   - What general and edge cases are covered by your test cases?
   - How did you ensure a balanced distribution of best, average, and worst-case scenarios?
   - Are there any cases you intentionally chose not to cover? Why?

- How robust are your test cases in detecting incorrect solutions?

5. **Deliverables 4 and 7: Input Validator and Test-Case Generator**
   - What challenges did you encounter while creating the validators/generators?
   - How did you ensure the correctness and completeness of your validators/generators?
   - How do your validators/generators handle edge cases and malformed inputs?
6. **Deliverable 8: Presentation**
   - What did you learn from this project?
   - How did you approach the design and delivery of your presentation?
   - To what extent did you use LLM technologies (e.g., Copilot, ChatGPT) during this project?
   - How was the collaboration and task distribution within your team?
   - What would you improve if you had more time?

Topics for the Problems:

For this assessment, your team will be randomly assigned one of the predefined problems. Each problem is associated with a designated Teaching Assistant (TA), who will support your team throughout the development process, provide feedback during your presentation, and evaluate your final submission.

Data Structures (Akash)

1. Stack
2. Queues
3. Priority Queue I
4. Priority Queue II
5. Balanced BST
6. Heaps
7. Hash Set
8. Hash Map
9. Union Find I
10. Union Find II
11. Prefix Sums
12. Trees I
13. Trees II

Algorithm Paradigms I (Parham)

14. Sorting – Implement or simulate a well-known sorting algorithm
15. Sorting – with constraints
16. Sorting – custom comparators
17. Sorting – no comparison-based algorithm
18. Bit-wise operators
19. Search in Strings
20. Complete Search
21. Recursive Backtracking
22. Divide & Conquer
23. Merge Sort (D&C)
24. Binary Search I
25. Binary Search II
26. Ternary Search

Algorithm Paradigms II (Ziqi)

27. Dynamic Programming 1D I
28. Dynamic Programming 1D II
29. Dynamic Programming 2D I
30. Dynamic Programming 2D II
31. Dynamic Programming Combinatorics
32. Dynamic Programming Optimization
33. Dynamic Programming in Sequences/Strings
34. Dynamic Programming – Game Theory
35. Greedy – Activity Selection
36. Greedy – Job Scheduling
37. Greedy – Involving Smart/Constrained Sorting
38. Greedy – Involving PQ
39. Greedy + DP

Ad-Hoc + Graph Theory (Ari)

40. Ad-hoc Simulation Problem
41. Ad-hoc Implementation Challenge – Test Case
42. Ad-hoc Implementation Challenge – Time Waster
43. Ad-hoc Mathematical Puzzle
44. Ad-hoc String Manipulation
45. Ad-hoc Date and Time Manipulation
46. Ad-hoc Pattern Recognition
47. Graph Traversal – BFS
48. Graph Traversal – DFS
49. Connected Components
50. Topological Sort
51. Minimum Spanning Tree
52. Shortest Path

What to Submit

You will find a file named `problemskeleton.zip` on MyCourses. This archive contains a template that you are expected to use as the foundation for your submission. You must submit the same file on MyCourses, but with all your completed deliverables added to their corresponding folders.

Please rename the folder to reflect the topic number of the problem your team was assigned. For example, if your team was randomly assigned Problem 40 (which corresponds to "Ad-hoc Simulation Problem"), your submission should be named: `problemskeleton_Problem40.zip`.

Where to Submit

Submit your assignment through the **Assignments** section on MyCourses. Note that only **one member of the team** should submit the final file on behalf of the group.

When to Submit

This assignment includes three key deadlines:

1. **Original Draft:** November **23rd**, 2025 (23:59)
   A complete initial draft of your problem and deliverables is expected by this date.
2. **Presentation (during class):** November **26th**, December **1st**, or December **3rd**
   Your team will present the problem during one of these class sessions. The Teaching Assistant assigned to your topic will provide feedback on both your draft and your presentation.
3. **Final Submission:** December **10th**, 2025 (23:59)
   The final version of your submission, incorporating feedback from the presentation, is due on this date.