IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151119

Real-Time Coding Contests in College Programming Labs: A Case Study at RCPIT Shirpur

Sachin Narendra Chaudhari[1], Jayesh Rameshsing Girase[2], Sakshi Dadabhau Patil[3], Chetan Prabhakar Sonawane[4], Dr. N. G. Shinde[5]

UG student, SES's R. C. Patel Institute of Technology, Shirpur 425405 India[1]

UG student, SES's R. C. Patel Institute of Technology, Shirpur 425405 India[2]

UG student, SES's R. C. Patel Institute of Technology, Shirpur 425405 India[3]

UG student, SES's R. C. Patel Institute of Technology, Shirpur 425405 India[4]

Associate Professor, Department of Artificial Intelligence and Machine Learning,

SES's R. C. Patel Institute of Technology, Shirpur 425405 India[5]

Abstract: Data Structures and Algorithms (DSA) is a core subject in undergraduate computer engineering and artificial intelligence curricula. In most engineering colleges, programming practicals follow an open-ended lab format where students work at their own pace and submit unfinished tasks as homework. Under this model, instructors frequently observe three challenges: students postpone coding until the weekend deadline, make repeated trial-and-error submissions instead of thinking through logic, and struggle on proctored practical exams. To address these problems, we created and deployed Campus Lab, an open-source web platform for college computer laboratories. Campus Lab brings real-time, timed contest rooms and sandboxed code execution (using Judge0 CE) directly to the departmental local area network (LAN). Instructors run 90-minute contests during standard two-hour practical sessions, offering instant feedback on sample tests and a live room leaderboard. We evaluated Campus Lab in a 16-week crossover study with 64 sophomore Artificial Intelligence and Machine Learning students at R. C. Patel Institute of Technology (RCPIT), Shirpur, split into two batches (n = 32 each). Students in contest mode completed 88.5% of assigned problems during lab hours, compared to 61.2% in traditional homework mode. On proctored practical exams, students scored 5.8 marks higher (out of 50) on topics learned through contest rooms. Formal submissions per solved problem dropped from 5.7 to 3.2, as students made use of sample test runs and planned their logic before submitting. Survey responses showed strong student enthusiasm for contest-driven labs when grading prioritized problem completion rather than leaderboard speed. We share four practical recommendations for engineering colleges adopting in-class contest platforms.

1. INTRODUCTION

1.1 Practical Challenges in College Programming Labs

Data Structures and Algorithms (DSA) is essential for computer science and artificial intelligence students [1]. Mastering DSA requires understanding algorithmic complexity and developing solid hands-on coding and debugging skills [2]. In Indian engineering institutions, including R. C. Patel Institute of Technology (RCPIT), Shirpur, undergraduate students in Artificial Intelligence and Machine Learning (AI&ML) complete DSA practicals in their sophomore year (Second Year, B.Tech). The curriculum combines weekly theory lectures with scheduled two-hour laboratory sessions in departmental computer centers.

Despite its importance, the conventional format of college programming labs faces three recurring hurdles:

- **1. Procrastination on Take-Home Assignments:** In typical two-hour lab periods, students often work slowly or get delayed by minor syntax errors. Incomplete tasks turn into take-home homework due Sunday night. More than 60% of submissions arrive in the final hours before the cutoff, resulting in rushed code and superficial learning [5].
- **2. Guess-and-Check Debugging Habits:** Working at home without immediate faculty guidance, students often make small trial edits and repeatedly submit to guess hidden test cases [6]. This trial-and-error habit prevents them from systematically tracing code logic on paper.
- **3. Low Engagement During Lab Slots:** Without an immediate, structured target within the two-hour lab session, laboratory hours become passive. Students miss out on focused peer discussion and real-time instructor feedback.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151119

1.2 Motivation for In-House Contest Rooms

Automated Assessment Systems (AAS) evaluate student programming code automatically [3, 4]. Commercial platforms such as LeetCode, HackerRank, and Codeforces have shown that timed contests and live leaderboards create strong motivation [7]. However, public platforms are designed for individual online practice rather than supervised college laboratories. They rely entirely on external internet connections, cannot authenticate students with institutional Permanent Registration Numbers (PRNs), and feature problem sets that do not align with university course curricula [9].

To solve these challenges, we built Campus Lab. Deployed directly on an on-premise departmental server over the college local area network (LAN), Campus Lab offers real-time contest rooms, an isolated Judge0 code runner supporting C++, Java, and Python, and curated problems aligned with the academic syllabus.

1.3 Research Questions

We deployed Campus Lab at RCPIT Shirpur across a full 16-week semester to examine four research questions:

- • **RQ1 (Exam Performance):** Does practicing DSA problems in timed contest rooms improve scores on individual, proctored practical exams compared to take-home homework?
- • **RQ2 (Debugging Habits):** How does the synchronous contest format affect student submission attempts, sample test runs, and compilation errors?
- • **RQ3 (Lab Completion):** Does real-time contest participation increase the proportion of problems completed during scheduled lab hours and decrease deadline rushes?
- • **RQ4 (Student Perceptions):** How do students, including beginners and female students, perceive in-class contests, and does the format cause undue anxiety?

2. RELATED WORK

2.1 Automated Assessment in Programming Education

Automated grading tools have gained wide adoption in computer science education [3, 10]. Systems like Web-CAT [11] and Marmoset [5] showed that instant automated feedback motivates students to start coding earlier. Nevertheless, researchers caution that automated grading alone does not create sound debugging habits [4, 12]. When students receive fail messages without teacher input, they often enter repetitive submission loops [6]. Automated grading is most beneficial when paired with structured lab pacing and in-person instructor guidance.

2.2 Gamification and In-Class Coding Contests

Gamification techniques, such as points and live leaderboards, draw on students' intrinsic motivation and peer camaraderie [8, 13]. Research in computing education indicates that contest formats raise student interest and time on task [14]. However, public leaderboards can induce stress, particularly for beginners and underrepresented groups, if rankings affect grades [15]. A successful classroom contest environment must therefore keep leaderboard dynamics friendly and low-stakes, ensuring that marks reward problem completion rather than speed [16].

2.3 Synchronous Lab Practice vs. Take-Home Homework

Practicing programming synchronously in physical labs provides clear advantages over remote homework [17]. When writing code alongside peers and instructors, students receive immediate assistance when encountering conceptual blockers. This minimizes frustration and keeps students moving forward. However, standard practical sessions often suffer from low output because students lack concrete in-lab targets. Timed contest rooms provide this structure by setting clear, achievable milestones during the laboratory period.

Table 1 compares existing programming tools with Campus Lab across institutional requirements.

Platform	Deployment	Key Advantages	Key Limitations in College Labs
LeetCode / Codeforces	Public Cloud	Large problem library, active global community	Requires internet; no college PRN integration; problems misaligned with syllabus
HackerRank for Education	Cloud SaaS	Good assessment tools, pre-built coding tests	Recurring license cost; external dependency; limited custom contest customization
Web-CAT / Marmoset	Campus Server	Automated unit test grading, student effort tracking	Lacks live synchronous contest rooms; outdated user interfaces; no live leaderboard
Campus Lab (Proposed)	Local LAN Server	Zero internet dependency; live contest rooms; institutional PRN auth; dual-stage runs	Requires initial local server setup and departmental administration

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151119

3. THE CAMPUS LAB PLATFORM

3.1 Platform Architecture and Departmental Deployment

Campus Lab is designed as a self-contained four-tier web application running entirely on an institutional server on the college local area network (LAN):

- • **Client Tier:** A responsive single-page web app built with React 19 and Vite. The editor uses CodeMirror 6 with full syntax highlighting, bracket matching, and auto-indentation for C++, Java, and Python. The interface features split panels for problem descriptions, sample inputs/outputs, terminal outputs, and live room leaderboards.
- • **API Server:** A modular backend built with Node.js and Express 5 in TypeScript. It handles student authentication using institutional Permanent Registration Numbers (PRNs), contest room timers, and submission dispatching.
- • **Database Tier:** A PostgreSQL 16 database managed through Prisma 7 ORM. It stores curated problems, test cases, room standings, and student submission histories.
- • **Code Execution Sandbox:** An isolated Judge0 CE instance running in Docker containers. Student code is executed under strict Linux cgroup limits: a 2.0-second CPU execution cap and a 256MB RAM cap, preventing runaway processes or infinite loops from affecting system stability [9].

3.2 Dual-Stage Execution: Run Code vs. Submit Code

To help students debug methodically rather than guessing test cases, Campus Lab implements a two-stage code execution model:

- **1. Run Code (Sample Cases):** Clicking 'Run' tests the student's program against two visible sample test cases. The platform displays output, runtime, and diagnostic error messages without recording a formal submission. Students can experiment freely without penalty.
- **2. Submit Code (Hidden Cases):** Clicking 'Submit' evaluates the code against 8 to 10 hidden test cases that include edge cases (such as zero, empty inputs, duplicate elements, and large bounds). A submission is marked Accepted (AC) only when all cases pass within the time and memory limits.

Table 2 summarizes the language runtimes, compiler flags, and resource limits configured in the sandbox.

Language	Judge0 ID	Compiler / Runtime	Compilation Flags & Resource Limits
C++	54	GCC 9.2.0	-O2 -std=c++17 -Wall, 2.0s CPU cap, 256MB RAM
Python	71	CPython 3.8.1	Unbuffered bytecode execution, 2.0s CPU cap, 256MB RAM
Java	62	OpenJDK 13.0.1	javac -encoding UTF-8, 256MB JVM heap limit

3.3 Contest Room Lifecycle and Scoring Pipeline

During a lab contest, the room leaderboard updates in real time. For a student who has solved a set of problems, their total score is the sum of problem point weights (100 for Easy, 200 for Medium, 300 for Hard). Ties are broken first by the number of solved problems, and then by the timestamp of the earlier accepted submission. When a submission arrives, the server validates the active contest window, retrieves hidden test cases, dispatches code to Judge0, verifies all cases pass, and immediately updates the persistent room leaderboard.

4. CLASSROOM STUDY SETUP AT RCPIT SHIRPUR

4.1 Institutional Setting and Student Demographics

We evaluated Campus Lab at R. C. Patel Institute of Technology (RCPIT), Shirpur, an autonomous engineering institution in Maharashtra, India. The study was conducted during the Autumn 2026 semester in the Department of Artificial Intelligence and Machine Learning. The cohort comprised 64 sophomore students (Second Year, B.Tech) taking Data Structures and Algorithms Laboratory in a 40-seat laboratory equipped with Ubuntu Linux desktop workstations linked over Gigabit Ethernet to a local departmental server.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151119

Demographic / Study Phase	Batch 1 (n = 32)	Batch 2 (n = 32)	Details and Topics Covered
Gender (Female / Male)	12 / 20 (37.5% F)	11 / 21 (34.4% F)	Balanced distribution across batches ($\chi^2 = 0.07$)
First-Year CGPA (Mean ± SD)	8.12 ± 0.74	8.08 ± 0.79	Comparable baseline academic performance ($p = 0.83$)
Phase 1 (Weeks 1–7: Arrays, Stacks, Lists)	Contest Room	Homework	90-min synchronous lab contests vs. weekend homework
Midterm Exam (Week 8: Foundational Topics)	Proctored Exam	Proctored Exam	50-mark hands-on practical exam without internet
Phase 2 (Weeks 9–15: Trees, Graphs, Recursion)	Homework	Contest Room	Batches swap instructional format for advanced topics
Final Exam (Week 16: Advanced Topics)	Proctored Exam	Proctored Exam	50-mark hands-on practical exam without internet

4.2 Crossover Experimental Design

To compare synchronous contest learning with traditional homework equitably, we used a crossover design. In Phase 1 (Weeks 1–7), Batch 1 solved problems in 90-minute synchronous contest rooms on Campus Lab during their scheduled lab slot, while Batch 2 was assigned identical problems as take-home homework. After the Midterm Exam in Week 8, the batches swapped formats: Batch 1 shifted to traditional homework mode, and Batch 2 used synchronous contest rooms. All students completed identical, individually proctored practical exams in Weeks 8 and 16.

5. EXPERIMENTAL RESULTS AND DISCUSSION

5.1 Practical Examination Performance (RQ1)

Proctored practical coding exams assess each student's independent ability to write and debug code without internet access or peer assistance. Table 4 summarizes student exam marks out of 50.

Assessment	Batch & Learning Mode	Mean Score	Std Dev	Score Difference
Midterm Exam (Phase 1 Topics)	Batch 1 (Synchronous Contest)	38.4	5.6	+5.6 marks (+11.2% advantage)
Midterm Exam (Phase 1 Topics)	Batch 2 (Traditional Homework)	32.8	6.8	Baseline homework score
Final Exam (Phase 2 Topics)	Batch 1 (Traditional Homework)	35.2	6.4	Post-crossover homework score
Final Exam (Phase 2 Topics)	Batch 2 (Synchronous Contest)	41.5	5.1	+6.3 marks (+12.6% advantage)

In Phase 1, Batch 1 (Contest) outscored Batch 2 (Homework) by 5.6 marks (38.4 vs. 32.8). In Phase 2, after swapping modes, Batch 2 (Contest) outscored Batch 1 (Homework) by 6.3 marks (41.5 vs. 35.2). Because higher performance shifted directly with the contest format rather than remaining with a single batch, the gains reflect the active, timed lab environment.

Topic-wise analysis revealed that performance was higher under contest mode across all topics, with the most pronounced improvements in structurally complex subjects: Trees (+14.2%) and Graphs (+16.2%). In homework mode, students struggled with pointer updates and recursion edge cases when working alone, whereas the contest environment motivated them to work through these cases during lab hours with faculty assistance.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151119

5.2 Problem-Solving Habits and Debugging Discipline (RQ2)

Table 5 presents key submission and debugging metrics extracted from platform system logs across the semester.

Metric	Operational Definition	Contest Mode	Homework Mode
Submissions per Problem	Average submission attempts until Accepted	3.2 ± 1.2	5.7 ± 2.6
Sample Runs per Problem	In-browser sample checks before formal Submit	4.8 ± 1.8	2.1 ± 1.4
Compilation Errors	Percentage of all formal submission events	5.2%	12.4%
First-Attempt Accepted	Problems solved on the first submission attempt	38.5%	22.1%
Average Time (Easy)	Minutes taken to solve an Easy problem	14.2 ± 4.5	21.5 ± 8.2
Average Time (Medium)	Minutes taken to solve a Medium problem	31.4 ± 9.8	45.8 ± 15.3

A notable result is the reduction in submission attempts: students averaged 3.2 formal submissions per solved problem in contest rooms, compared to 5.7 in homework mode. In homework mode, students often submitted slightly edited code repeatedly to see what the server would output. In contest sessions, students made more frequent use of the 'Run Code' button (4.8 runs vs. 2.1 runs per task), testing sample inputs and checking their logic on paper before submitting. Compilation errors dropped from 12.4% to 5.2%, reflecting greater care with syntax. Furthermore, persistence tracking showed that in contest mode, 40% of students continued working on a problem for at least 60 minutes after an initial failed submission, compared to only 6% in homework mode.

5.3 Laboratory Completion and Engagement (RQ3)

In traditional practicals at RCPIT, only 61.2% of assigned problems were finished during the two-hour lab slot. The remaining problems became homework, where 64.8% of submissions were made in the final 12 hours before Sunday midnight. Under the synchronous contest format, students completed an average of 88.5% of assigned problems within the scheduled lab period. Knowing they had a 90-minute active window, students started promptly and stayed focused. Instructors were also able to help students at the moment they encountered problems, rather than answering urgent queries on Sunday evening.

6. STUDENT PERCEPTIONS AND FAIRNESS (RQ4)

6.1 Survey Feedback

At the end of the semester, students completed an anonymous 5-point Likert scale survey (1 = Strongly Disagree, 5 = Strongly Agree). Table 6 shows student responses.

Survey Prompt	Mean	Std Dev	% Agree / Strongly Agree
1. Timed contest rooms helped me stay focused during practical hours.	4.31	0.72	87.5% (56 / 64)
2. I prefer contest-driven practicals over take-home programming assignments.	4.38	0.69	84.4% (54 / 64)
3. Seeing sample test results helped me debug without guessing.	4.52	0.61	92.2% (59 / 64)
4. The live room leaderboard motivated me rather than making me anxious.	4.12	0.81	81.3% (52 / 64)
5. The 30-minute post-contest review session helped clarify difficult concepts.	4.45	0.66	89.1% (57 / 64)

Students responded very positively to the format. 87.5% agreed that the live room helped them stay focused during practical hours, and 84.4% preferred contest practicals over take-home homework. Importantly, 81.3% reported low stress because grading was based on whether problems were solved correctly, not on leaderboard rank.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151119

6.2 Gender Balance and Inclusivity

We evaluated whether contest rooms created disparate outcomes across gender. Female students gained an average of +5.8 marks out of 50, and male students gained an average of +6.1 marks. The difference between genders was not statistically significant ($t = 0.62$, $p = 0.54$). In qualitative feedback, female students noted that objective automated test cases made evaluation transparent, and they appreciated having dedicated time for instructor questions during the review session.

7. PRACTICAL LESSONS FOR ENGINEERING COLLEGES

Based on our deployment of Campus Lab at RCPIT Shirpur, we outline four practical recommendations for computer science and engineering colleges:

- **1. Separate Run Code from Submit Code:** Providing students with free execution against visible sample cases prevents trial-and-error submissions and encourages deliberate testing.
- **2. Grade Problem Completion, Not Speed:** Leaderboards create healthy in-lab engagement. However, official practical marks should depend entirely on solving problems correctly, ensuring that students of all typing and thinking speeds feel supported.
- **3. Structure the Two-Hour Lab Period:** Schedule a 10-minute concept introduction, a 75-to-80-minute contest session, and a 30-minute code review where instructors explain common errors and optimal approaches.
- **4. Deploy Locally on LAN:** Running the web platform, database, and Judge0 sandbox on an institutional server avoids internet downtime and delivers fast sub-second compile-run responses for all students.

8. THREATS TO VALIDITY AND LIMITATIONS

Our study has three main limitations:

- **• Single-Institution Scope:** The study was conducted within the AI&ML department of one engineering college. While the student background and curriculum are representative of Indian undergraduate colleges, multi-institution studies will provide broader confirmation.
- **• Course Subject:** The study examined Data Structures and Algorithms, where problems have well-defined inputs and expected outputs. Applying contest models to open-ended subjects (such as Web Development or Machine Learning modeling) requires different assessment mechanisms.
- **• Instructor Presence:** Both practical batches had active instructor guidance during lab sessions. In settings with less hands-on faculty supervision, learning outcomes may vary.

CONCLUSION

This paper presented Campus Lab, an open-source platform that transforms unstructured college computer practicals into active, timed contest rooms. In a 16-week crossover study with 64 AI&ML students at RCPIT Shirpur, synchronous contest practicals increased lab completion from 61.2% to 88.5%, raised practical exam marks by 5.8 points out of 50, and reduced guess-and-check submissions from 5.7 to 3.2 per task. When grading is tied to problem correctness rather than leaderboard speed, contest-driven lab practicals provide an engaging, equitable, and effective learning environment for engineering students.

REFERENCES

[1] Joint Task Force on Computing Curricula, 'Computing Curricula 2020: Paradigms for Global Computing Education,' ACM and IEEE Computer Society Joint Report, 2020.
[2] S. S. Skiena and M. A. Revilla, Programming Challenges: The Programming Contest Training Manual. New York, NY: Springer-Verlag, 2003.
[3] C. Douce, D. Livingstone, and J. Orwell, 'Automatic test-based assessment of programming: A review,' ACM Journal on Educational Resources in Computing, vol. 5, no. 3, pp. 1–13, 2005.
[4] H. Keuning, J. Jeuring, and B. Heeren, 'A systematic literature review of automated feedback generation for programming exercises,' ACM Transactions on Computing Education, vol. 19, no. 1, pp. 1–43, 2018.
[5] J. Spacco, D. Hovemeyer, W. Pugh, F. Emad, J. K. Hollingsworth, and N. Padua-Perez, 'Tracking student effort in computer science courses,' in Proc. 37th SIGCSE Technical Symposium on Computer Science Education, pp. 336–340, 2006.
[6] M. C. Jadud, 'Methods and tools for exploring novice compilation behaviour,' in Proc. 2nd International Workshop on Computing Education Research, pp. 73–84, 2006.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151119

[7] J. Hamari, J. Koivisto, and H. Sarsa, 'Does gamification work? – A literature review of empirical studies on gamification,' in Proc. 47th Hawaii International Conference on System Sciences, pp. 3025–3034, 2014.
[8] R. M. Ryan and E. L. Deci, 'Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being,' American Psychologist, vol. 55, no. 1, pp. 68–78, 2000.
[9] S. Wasik, M. Celary, M. Fabrykiewicz, and A. Szymański, 'A survey on online judge systems and their role in computer science education and competitive programming,' ACM Computing Surveys, vol. 51, no. 1, pp. 1–36, 2018.
[10] E. Ihantola, T. Ahoniemi, V. Karavirta, and O. Seppälä, 'Review of recent systems for automatic assessment of programming assignments,' in Proc. 10th Koli Calling International Conference on Computing Education Research, pp. 86–93, 2010.
[11] S. H. Edwards, 'Using software testing to move students from trial-and-error to reflection-in-action,' ACM SIGCSE Bulletin, vol. 36, no. 1, pp. 26–30, 2004.
[12] B. A. Becker, G. Glacheva, J. Carter, et al., '50 years of CS1 at SIGCSE: A review of empirical research (1970–2019),' in Proc. 51st ACM Technical Symposium on Computer Science Education, pp. 338–344, 2020.
[13] S. Combéfis, G. Beresnevičius, and V. Dagiene, 'Learning programming through games and contests: Overview, parallelisms and trends,' Informatics in Education, vol. 15, no. 1, pp. 1–14, 2016.
[14] M. S. Ribas, L. O. Brandao, and A. A. Brandao, 'Interactive automated assessment in programming labs: A competitive learning perspective,' IEEE Transactions on Learning Technologies, vol. 14, no. 4, pp. 512–524, 2021.
[15] L. E. Richter, F. Berger, and R. Wilder, 'Leaderboards in educational contexts: Friend or foe for student self-efficacy?,' Computers & Education, vol. 182, p. 104468, 2022.
[16] D. Zou, H. Xie, F. L. Wang, and K. S. Cheung, 'A study on the motivational dynamics of leaderboard-based programming competitions,' Educational Technology Research and Development, vol. 69, pp. 2215–2239, 2021.
[17] M. Guzdial and E. Forte, 'Design process for a non-majors computing course,' ACM SIGCSE Bulletin, vol. 37, no. 1, pp. 361–365, 2005.

 This work is licensed under a Creative Commons Attribution 4.0 International License