ACTE: Individual/Group Computer Programming Rubric

DIRECTIONS
1. Ask the student/group for the **printed portfolio**. Digital or electronic portfolios are not accepted. Allow no more than 5 minutes for project setup and score the printed portfolio during setup.
2. Confirm that the program, algorithms, source code, interface, data, documentation, tests, and production assets are student-created. **AI-generated code, programming content, or production assets are prohibited.** If suspected or disclosed, pause scoring and refer the entry to ACTE staff.
3. Allow no more than 10 minutes for the project presentation. The student may choose which portion of a larger body of work to show.
4. Judges may ask questions for no more than 5 minutes and are not limited to the topics listed in the rubric.
5. Enter a score for every criterion. Add the /30 portfolio score and /70 project/presentation score for the /100 final score, then initial page 2.

PRINTED PORTFOLIO CHECKLIST			Points
Online Orientation Certificate	**0** Certificate is missing.	**10** Certificate is printed and placed as page 1 of the portfolio.	
Printed Portfolio	**0** No printed portfolio is submitted, or only a digital/electronic portfolio is provided.	**10** A complete printed portfolio is submitted at judging. Digital portfolios are not accepted.	
Required Portfolio Sections	**0** Required printed portfolio sections are not provided.	**10 - Printed portfolio includes these 9 sections:** 1. Online Orientation Certificate (first page) 2. Project identification page: student name(s), school, category, level, and region 3. Table of contents 4. Project narrative: purpose, learning, and benefit 5. Software and hardware used 6. Problem statement, target user, requirements, inputs/outputs, constraints, research, algorithm, flowchart or pseudocode, data design, interface plan, and development schedule 7. Representative student-written source code, version history, development screenshots, test cases/results, error logs, debugging notes, performance checks, revisions, and final executable/output evidence 8. Works cited/bibliography and credits for libraries, frameworks, APIs, tutorials, datasets, media, and other sources 9. Signed statement that the program, source code, algorithms, documentation, and production assets are student-created and contain no AI-generated code, content, or assets	

Judges Printed Portfolio Checklist Score	**/30**

This section is valued at 30 points total.

COMPUTER PROGRAMMING PROJECT & ORAL PRESENTATION RUBRIC

Select the score range that best matches the demonstrated level. Enter one numeric score from 0-10 for each criterion.

PROJECT					Points
Problem, Requirements & Algorithmic Plan					
0 No identifiable problem, user, requirements, inputs/outputs, algorithm, or plan.	**1-3** Problem and requirements are unclear; little evidence of planning, decomposition, pseudocode, or flowcharting is provided.	**4-6** A basic problem and plan are evident, but requirements, inputs/outputs, constraints, algorithm, or data design is incomplete.	**7-9** Clear problem, user needs, requirements, inputs/outputs, constraints, decomposition, and logical algorithm are documented.	**10** Exceptional plan thoroughly defines the problem, analyzes requirements, decomposes complexity, and designs an efficient, purposeful algorithm.	
PROJECT					**Points**
Code Structure, Readability & Programming Concepts					
0 No functioning student-written source code is provided.	**1-3** Code is incomplete, copied without understanding, disorganized, poorly named, or difficult to follow.	**4-6** Code uses basic sequence, selection, iteration, variables, and/or functions, but organization, naming, reuse, or comments are uneven.	**7-9** Well-organized code uses appropriate data types, control structures, functions/classes, naming, comments, and modular design.	**10** Exceptional code demonstrates advanced abstraction, modularity, maintainability, readability, reuse, and purposeful programming concepts.	
PROJECT					**Points**
Functionality, Accuracy & User Experience					
0 Program does not run or cannot perform its intended function.	**1-3** Major features are missing or broken; incorrect results, crashes, confusing input/output, or poor validation prevent reliable use.	**4-6** Core features work, but accuracy, validation, error handling, interface clarity, data handling, or consistency is uneven.	**7-9** Program reliably meets requirements with accurate results, clear input/output, useful validation, and effective user experience.	**10** Production-ready program exceeds requirements with robust features, precise results, strong validation, graceful errors, and polished usability.	
PROJECT					**Points**
Testing, Debugging, Efficiency & Security					
0 No testing or debugging evidence is provided, or the program cannot be evaluated.	**1-3** Testing is informal; major errors remain and little evidence of debugging, edge-case checks, efficiency, or safe data handling is shown.	**4-6** Basic tests and debugging are evident, but coverage, edge cases, performance, security, or error recovery is inconsistent.	**7-9** Documented test cases cover normal and edge conditions; effective debugging, efficient logic, and appropriate security practices support reliability.	**10** Rigorous automated or systematic testing, optimization, secure design, error recovery, and performance analysis produce exceptional reliability.	
PROJECT					**Points**
Originality, Process Evidence & Student Authorship					
0 Originality, development process, or student authorship cannot be established.	**1-3** Work shows limited original decisions; copied code, sources, versions, or student contribution are unclear.	**4-6** Student contribution is evident, but originality, process evidence, source credits, or technical decisions are basic.	**7-9** Original, purposeful program shows clear student ownership, appropriate source credits, and strong evidence of planning, coding, testing, and revision.	**10** Distinctive solution proves independent authorship and complete understanding of major algorithms and code decisions.	
PRESENTATION					**Points**
Explanation of Development Process & Professional Delivery					
0 Does not explain or demonstrate the program or development process.	**1-3** Explanation is difficult to follow; limited preparation or understanding of how the program was planned and coded.	**4-6** Explains major development steps and code, but detail, organization, testing, debugging, or delivery is inconsistent.	**7-9** Clearly explains the problem, algorithm, language, code structure, data, testing, debugging, revisions, and final performance professionally.	**10** Confident, concise demonstration shows command of the complete software-development workflow and can explain key algorithms and code.	
QUESTIONS					**Points**
Programming Knowledge, Problem-Solving & Responses					
0 Does not answer judges' questions.	**1-3** Answers are incomplete or show little understanding of the language, algorithm, code, data, tests, or errors.	**4-6** Answers most questions and describes basic code, data, testing, or debugging challenges.	**7-9** Provides specific, accurate responses about algorithms, syntax, structures, functions/classes, data, testing, security, sources, and revisions.	**10** Insightful responses demonstrate deep programming knowledge, effective debugging, computational thinking, reflection, and equal group contribution.	

AI-GENERATED PROGRAMMING CONTENT PROHIBITED: Entries may not contain generative-AI-created source code, algorithms, pseudocode, flowcharts, interfaces, tests, documentation, text, images, datasets, or other production assets. Judges should not independently penalize suspected use; pause scoring and refer the entry to ACTE staff for review.

Judges Printed Portfolio Checklist Score	**/30**
Computer Programming Project & Oral Presentation Score	**/70**
Total Project Score	**/100**
Ranking (ACTE Staff Only)	

Judges Initials: _____