

## AP Computer Science Principles | Unit 1

### 1.1 Purpose, Requirements, and Users

#### Solutions

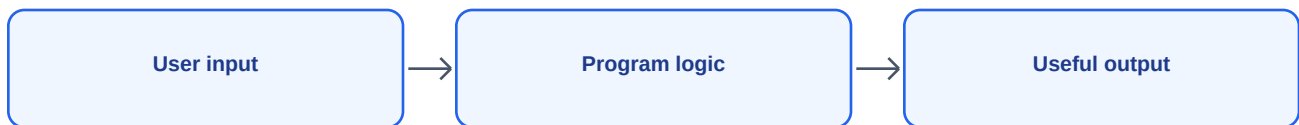
##### Reference scenario

A homework planner contains buttons and a list. State its user-facing purpose and measurable behavior.

Users: students

Input: task and due date

Output: upcoming tasks in date order



#### 1.1.1

For a cafeteria app that lists meals and allergens, state one purpose, one input, and one output.

Purpose: help users choose a suitable meal. Input: a selected day or dietary filter. Output: matching meals and their allergen information.

#### 1.1.2

Turn "make the app easy to use" into a testable requirement for adding a homework task.

For example: a student can enter a task and due date, save it, and see it in the task list without re-entering information. Define how the usability test will observe successful completion.

## AP Computer Science Principles | Unit 1

### 1.2 Iterative Design

#### Solutions

##### Reference scenario

User testing finds that an unlabeled icon prevents students from saving a task.

Prototype 1: 2 of 6 users save successfully

Revision: add visible Save label

Prototype 2: 5 of 6 users save successfully

#### 1.2.1

Why is a small prototype useful before implementing every feature?

It lets users and developers test important assumptions early, so major interface or requirement problems can be corrected before they become expensive to change.

#### 1.2.2

Write a second revision cycle for users who still cannot locate saved tasks.

Observe where users look, prototype a clearer task-list entry or confirmation, test the same retrieval task with representative users, and revise based on the results.

## AP Computer Science Principles | Unit 1

### 1.3 Collaboration

#### Solutions

#### Reference scenario

Two teammates propose different search interfaces. Resolve the disagreement with evidence.

Requirement: locate a task by keyword

Option A: text search

Option B: category-only menu

Users: tasks may span categories

#### 1.3.1

Name two collaboration practices that reduce accidental loss of another teammate changes.

Use a shared version history or controlled file workflow, and communicate which changes are being made.

Review and reconcile edits before combining them.

#### 1.3.2

Why is attribution important when a team uses external code or media?

It identifies what was created by the team and what came from others, supports license compliance, and helps reviewers understand the contribution. Attribution alone does not grant permission to use material.

## AP Computer Science Principles | Unit 1

### 1.4 Testing and Debugging

#### Solutions

##### Reference scenario

A program admits ages 13 through 18 inclusive, but its condition excludes age 18.

Current condition: `age >= 13 AND age < 18`

Required upper boundary: 18 included

Test input: 18

Actual: rejected

#### 1.4.1

Provide expected results for ages 12, 13, 18, and 19 under the stated requirement.

12: reject. 13: admit. 18: admit. 19: reject. Both boundaries must be included and their adjacent outside values excluded.

#### 1.4.2

Why does one successful test at age 15 not establish that both boundaries work?

An interior value can pass under several incorrect comparisons. Boundary and adjacent outside cases distinguish inclusive from exclusive conditions.