Chapter 1 Exam Format, Grading, and Tips

1.1 Exam Format and Materials

Figure 1-1 shows the format of the AP Computer Science exam. The exam takes 3 hours of test time, plus breaks and time for instructions. It is divided into two sections. Section I consists of 40 multiple-choice questions with a total allotted time of 1 hour and 30 minutes (2.25 minutes per question on average). Section II consists of four free-response questions with a total allotted time of 1 hour and 30 minutes (22.5 minutes per question on average). The free-response questions usually consist of two parts each (but occasionally might have three parts).

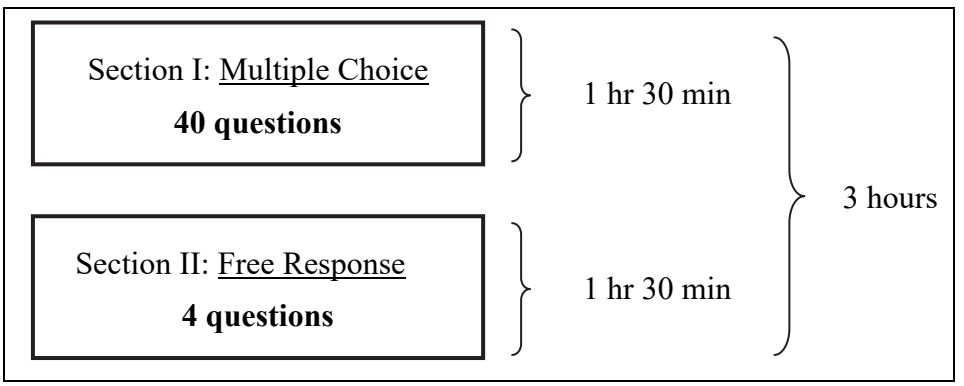

Figure 1-1. AP Computer Science exam format

No computers, calculators, other devices, books, or materials are allowed, only paper, pencil and an eraser.

Pens are allowed, but we highly recommend that you use a pencil.

At the exam you will receive the *Quick Reference* page, both in your multiple-choice questions booklet and again in your free-response questions booklet (in other words, the same *Quick Reference* page will be given to you twice). The *Quick Reference* provides a list of the Java library classes and their methods included in the AP Java subset. Students and teachers can obtain a copy at the College Board's AP site (see www.skylit.com/cblinks.html). The *Quick Reference* is provided for reference — it is expected that you will already be very familiar and comfortable with the required Java library classes and their methods before the exam.

The multiple-choice section is a mixture of questions related to general computer science terms, program design decisions, specific elements of Java syntax, logical analysis of fragments of Java code, properties of classes, and some OOP concepts. Starting at the 2020 exam, the College Board has made the topics of the free-response questions more specific:

Question 1: Methods and Control Structures
Question 2: Class
Question 3: Array / ArrayList
Question 4: 2D Array

Each free-response question is graded out of 9 points, with all points weighted equally. The second part of the question often refers to the class or method implemented in the first part, but each part is graded separately, and your implementation of Part (a) does not have to be correct in order for you to receive full credit for Part (b).

In the free-response Question 2 you will be asked to write a complete small Java class. Your implementation will be graded based on the correctness of the implementation of your class, including the encapsulation principle (declaring all instance variables private), and other criteria.

In the past, the AP CSA exams included a specific case study, developed by the Exam Development Committee. In 2008-2014, the case study was GridWorld.

Starting with the 2015 exam, neither GridWorld nor any other case study was part of the exams, and they won't be in the future.

The College Board requires a typical AP CSA course to include at least 20 hours of computer lab work. The Development Committee has made available sample labs for practice.

These labs are only examples designed to illustrate the scope and difficulty of lab work in a typical AP CSA course. The content and code in these labs <u>will not be tested</u> on the AP CSA exams.

1.2 The Java Subset

The old course description included a table that defined the AP Java Subset. There is no such table in the new CED, but you can glean the elements of the subset from the *Essential Knowledge* components in the CED. The Java subset remains essentially the same, with the exception of abstract classes and interfaces.

So what is in the subset? Actually, quite a bit:

- Comments: `/*` ... `*/`, `//`, and `/**` ... `*/`, preconditions and postconditions.
- `boolean`, `int`, and `double` primitive data types. `(int)` and `(double)` casts. **Other primitive data types, including `char`, are not in the subset and should be avoided on the exam.**
- The assignment operator =. The arithmetic operators +, -, *, /, %. The increment/decrement operators ++ and -- (**use only the postfix form `k++` or `k--`, and do not use them in expressions)**. The compound assignment operators +=, -=, *=, /=, %=. The relational operators <, >, <=, >=, ==, !=. The logical operators &&, ||, !.
- `if-else` statements. The `true` and `false` keywords.
- Literal strings in double quotes. `\n`, `\\`, and `\"` escape characters in literal strings. The + and += operators for concatenating strings. `String`'s `compareTo`, `equals`, `length`, `substring`, and `indexOf(String s)` methods.
- `System.out.print` and `System.out.println`.
- One-dimensional and two-dimensional **(rectangular only)** arrays. `array.length`. Arrays of objects. Initialized arrays such as
```
int[] arr = {1, 2, 3};
int[][] m = {{1, 2, 3}, {4, 5, 6}};
```
Default values of elements for arrays defined using the `new` operator.
- `for` and `while` loops. The "enhanced" `for` loop ("for each" loop): `for(`*`type`* `x : values)...`
- Classes. Constructors. The `new` operator. `public` and `private` methods. `static` methods. Overloaded methods. The `return` statement. `void` methods.
- `static` instance variables and `static final` variables (constants). `null`. All instance variables are `private`.
- Inheritance, `extends`. Calling a superclass's constructor from a subclass, as in `super(...)`. Overriding a superclass's methods. Calling a superclass's method from a subclass, as in `super.someMethod(...)`. Passing `this` object to a method, as in `otherObject.someMethod(this)`.
- Understanding `ArithmeticException`, `NullPointerException`, `IndexOutOfBoundsException`, `ArrayIndexOutOfBoundsException`, `ConcurrentModificationException`.
- The `ArrayList<`*`E`*`>` class (see Section 2.6).

- Other library classes, methods, and constants:

  String: `length()`, `substring(...)`, `indexOf(String s)`
  Integer: `Integer(int x)`, `intValue()`;
  `Integer.MIN_VALUE` and `Integer.MAX_VALUE`
  Double: `Double(double x)`, `doubleValue()`
  Math: `abs(int x)`, `abs(double x)`,
  `pow(double base, double exp)`, `sqrt(double x)`,
  `random()`
  Object: `equals(Object other)`, `toString()`

  Also understand the `toString` and `equals` methods for all objects and the `compareTo` method for `String`, `Integer`, and `Double` objects.

> **If you feel you must stray from this subset in your free-response solution, you might have misunderstood the problem and be making it harder than it is.**

At the same time, it is OK to use such out-of-the-subset features as `Math.min(x, y)` and `Math.max(x, y)` methods, `ArrayList`'s `contains` and `remove(object)` methods, `break` in loops, and other simple tools that all exam readers are familiar with.

Things that are <u>not</u> in the AP Java subset and should be avoided include the following:

- Java syntax abominations, such as the `?_:_` operator
- `++` and `--` in expressions (as in `a[i++]`)
- Primitive data types other than `boolean`, `int`, and `double` (`char`, `long`, `float` are <u>not</u> in the subset)
- Bitwise logical operators and shift operators: `~`, `&`, `|`, `^`, `<<`, `>>`.

Also <u>not</u> in the subset and will not be tested:

- The `switch` statement, the `do-while` loop, `break` and `continue` in loops
- The prefix form of `++` and `--` operators (`++k`, `--k`)
- Library classes and methods other than those mentioned above
- Abstract classes and interfaces
- Checked exceptions and `try-catch-finally` statements
- `System.in` and `Scanner`; any input and output other than `System.out.print` and `System.out.println`
- `enum` data types

1.3 Tested Terms, Concepts, and Algorithms

In addition to Java, you need to be familiar with the following terms, concepts, and algorithms mentioned in the *Essential Knowledge* components of the CED:

- Rounding: `(int)(x + 0.5)` for a positive `double x`; `(int)(x - 0.5)` for negative `x`
- *Autoboxing* / *unboxing*
- Constructor and method *signature* (name and data types of parameters)
- *Invoking* constructors, calling methods using the "dot" operator
- *Local* variables; *instance* variables
- *Formal parameters*
- Passing parameters *by value* (an object is passed as a copy of reference)
- *Accessor* and *mutator* (*modifier*) methods
- *Class, subclass*, the *IS-A* relationship between objects
- *Polymorphism*
- Array / `ArrayList` algorithms: traverse, find the minimum or maximum, find the sum or average of elements, examine all pairs of consecutive elements, find duplicates, examine all elements for a given property
- Sequential Search and iterative and recursive versions of Binary Search
- Selection Sort, Insertion Sort, and Mergesort
- Understand **(but not write)** recursive code

The Java subset features and the above terms, concepts, and algorithms are reviewed in the subsequent chapters.

1.4 Grading

The exam is graded on a scale from 1 to 5. Table 1-1 shows the College Board's college credit recommendations and grade equivalents for AP grades.

AP Grade	Credit Recommendation	College Grade Equivalent
5	Extremely well qualified	A
4	Well qualified	A-, B+, B
3	Qualified	B-, C+, C
2	Possibly qualified	n/a
1	No recommendation	n/a

Table 1-1. AP grades, credit recommendations, and college grade equivalents

Grades of 5 and 4 are called "extremely well qualified" and "well qualified," respectively, and usually will be honored by colleges that give credit or placement for AP CSA. A grade of 3, "qualified," may be denied credit or placement at some colleges. Grades of 2, "possibly qualified," and 1, "no recommendation," will not get you college credit or placement.

Table 1-2 presents published statistics and grade distributions for the 2017 and 2018 exams. 65,133 students took the exam in 2018; 45.9 percent of them scored 4 or 5.

AP Computer Science A	2018		2017	
	Number	%	Number	%
Students	65,133	100.0	60,519	100.0
Grade:				
5	16,105	24.7	14,623	24.2
4	13,802	21.2	12,650	20.9
3	14,222	21.8	13,271	21.9
2	7,738	11.9	6,970	11.5
1	13,266	20.4	13,005	21.5
4 or 5	29,907	45.9	27,273	45.1

Table 1-2. 2018 and 2017 grade distributions

The multiple-choice and free-response sections weigh equally in the final grade.

The College Board uses a weighted combination of the multiple-choice (MC) and free-response (FR) scores to determine the final total score:

```
totalScore =
     MC_coeff * countCorrect + FR_coeff * FR_score;
```

One point is given for each correct answer to a multiple-choice question.

There is no penalty for giving a wrong answer to a multiple-choice question, so it is a good strategy not to leave any answers blank.

Solutions to free-response questions are graded by a group of high school teachers and college professors. Scores are based on a *rubric* established by the Chief Reader, Exam Leader, and Question Leaders. Each free-response question is graded out of 9 points, with partial credit given according to a rubric.

The final score is obtained by adding the MC and FR weighted scores. The MC and FR coefficients are chosen in such a way that they give equal weights to the multiple-choice and free-response sections of the exam. For example, if the exam has 40 multiple-choice questions and 4 free-response questions, weights of 1.0 for multiple-choice and 1.1111 for free-response will give each section a maximum total of 40, for a maximum possible total score of 80.

The cut-off points are determined by the Chief Reader in consultation with the College Board and may vary slightly from year to year based on the score distributions and close examination of a sample of individual exams. Table 1-3 shows the cut-off points for the 2015 exam.

Composite score range	Percent range	Grade
62 - 80	77.5 - 100	5
44 - 61	55.0 - 77.4	4
31 - 43	38.8 - 54.9	3
25 - 30	31.3 - 38.7	2
0 - 24	0 - 31.2	1

Table 1-3. 2015 score cut-off points for grades

Over 98% of students who answer correctly at least 27 out of 40 questions on the multiple-choice section typically receive a 4 or a 5 for the whole exam.

The College Board releases the free-response questions 48 hours after the exam and posts them on their website. We post our annotated solutions at www.skylit.com/beprepared/ a few hours after the questions are posted. The College Board posts the scoring rubrics for the free-response questions sometime in the summer. Every few years the College Board releases a complete exam, including a diagnostic guide for the multiple-choice questions, grading rubrics for the free-response questions, and the cut-off points for grades. See www.skylit.com/beprepared/notes.txt for the latest updates.

1.5 College Credit

Most colleges will take your AP courses taken and exam grades into account in admission decisions. But acceptance of AP exam results for credit and/or placement varies widely among colleges. In general, the AP CSA course corresponds to a CS-1 course (Introductory Computer Science or Computer Programming I), a one-semester course for computer science majors. Some colleges may base their decision on your grade, and some may not give any credit at all. Consult the websites of the colleges you are interested in and the College Board's AP credit policies page https://apstudent.collegeboard.org/creditandplacement/search-credit-policies.

1.6 Exam Taking Tips

Some things are obvious:

- If you took the time to read a multiple-choice question and all the answer choices, take an extra ten seconds and guess. Most likely you have eliminated one or two wrong answers even without noticing. **Do not leave any multiple-choice answers blank: there is no penalty for wrong answers**.
- If a common paragraph refers to a group of questions and you took the time to read it, try each question in the group.
- Do read the question before jumping to the code included in the question. Notes to multiple-choice questions in our practice exams might show you some shortcuts.
- In questions with I, II, III options, work from the answers; for example, you might be able to eliminate two or three answer choices if you are sure that Option I doesn't work.

There are a few important things to know about answering free-response questions.

Remember that all free-response questions have equal weight. The first question is likely to be the easiest.

In a nutshell: be neat, straightforward, and professional; keep your exam reader in mind; don't show off.

More specifically:

1. Stay within the Java subset, except for a few obvious shortcuts, such as `Math.max(...)` and `Math.min(...)`.

2. Remember that the elegance of your code does not count. More often than not, a brute-force approach is the best. You may waste a lot of time writing tricky, non-standard code and trick yourself in the process or mislead your exam reader who, after all, is only human. Your exam reader will read your solution, but will not test it on a computer.

3. Superior efficiency of your code does not count, unless the desired performance of the solution is specifically stated in the question.

4. Remember that Parts (b) and (c) of a question are graded independently from the previous parts, and may actually be easier: Part (a) may ask you to write a method, while Part (b) or Part (c) may simply ask you to use it. It is not uncommon for method(s) specified in Part (a) to be called in subsequent parts. Do so, even if your Part (a) is incorrect or left blank. Do not re-implement code from earlier parts in later parts — you will waste valuable time and may lose points for doing so.

5. If a question presents a partial definition of a class with certain methods described but not implemented ("implementation not shown"), call these methods whenever appropriate in your code — do not write equivalent code yourself.

6. Bits of "good thinking" count. You may not know the whole solution, but if you have read and understood the question, go ahead and write fragments of code that may earn you partial credit points. On the free-response question #2, write the class, constructor, and method headers even if you are not sure how to implement them. But don't spend too much time improvising incorrect code.

7. Don't waste your time erasing large portions of work. Instead, cross out your work neatly, but only after you have something better to replace it with. Do not cross out a solution if you have no time to redo it, even if you think it is wrong. You won't be penalized for incorrect code and may get partial credit for it. Exam readers are instructed not to read any code that you have crossed out. But if you wrote two solutions, be sure to cross one out: otherwise only the first one on the page will be graded.

8. Read the comment above the method header quickly — it usually restates the task in a more formal way and sometimes gives valuable hints. Assume that all preconditions are satisfied — do not add unnecessary checks to your code!

9. One common mistake is to forget a `return` statement in a non-`void` method. Make sure the returned value matches the specified method return type.

10. Do not ignore hints in the question description. If an algorithm is suggested for a method (as in "you may use the following algorithm"), don't fight it, just use it!

11. Remember that the exam readers grade a vast number of exams in quick succession during a marathon grading session every June. Write as neatly as possible. Space out your code (don't save paper).

12. Always indent your code properly. This helps you and your exam reader. If you miss a brace but your code is properly indented, the reader (as opposed to a Java compiler) may accept it as correct. Similarly, if you put each statement on a separate line, a forgotten semicolon might not be held against you.

13. Follow the Java naming convention: the names of all methods, variables, and parameters start with a lowercase letter. Use meaningful, but not too verbose, names for variables. `count` may be better than `a`; `sum` may be better than `temp`; `row`, `col` or `r`, `c` may be better than `i`, `j`. But `k` is better than `loopControlVariable`. If the question contains examples of code with names, use the same names when appropriate.

14. Don't bother with comments; they do not count and you will lose valuable time. Occasionally you can put a very brief comment that indicates your intentions for the fragment of code that follows. For example:

```
// Find the first empty seat:
...
...
```

15. Don't worry about `imports` — assume that all the necessary Java library classes are imported.

16. Code strictly according to the specifications and preconditions and postconditions. Avoid extraneous "bells and whistles" — you will lose points. Never add `System.out.print/println` in solutions unless specifically asked to do so. Never read any data in your method — it is passed in as a parameter or is available as instance or static variables of the class.

17. Do not use in your code specific numbers, strings, or dimensions of arrays given as examples in explanations of questions. If the question says, "For example, a two-dimensional array `pixelValues` may contain the following image" and shows an array of 4 rows by 5 columns, do not use 4 and 5 in your code — make your code work with an array of any size.

18. Don't try to catch the exam authors on ambiguities: there will be no one to hear your case, and you'll waste your time. Instead, try to grasp quickly what was meant and write your answer.

19. Don't quit until the time is up. Use all the time you have and keep trying. The test will be over before you know it.