Sources for Sample Questions

Here are several good sources of questions like those that are found on AP Computer Science Exams:

AP Central

- Free-response questions from prior years' AP Exams. Use the detailed scoring guidelines to score your students' solutions, which will help them understand the benefit of demonstrating their understanding of the algorithms, even if their solutions aren't completely correct.
  [AP Central > The Exams > Exam Questions]
- *AP Computer Science Course Description*
  Download the latest version from the Course Home Pages on AP Central.

AP Computer Science Review Books

Horwitz, Susan. *Addison-Wesley's Review for the AP Computer Science Exam in Java*. Boston: Addison-Wesley, 2004.

Levine, David, and Kathy Larson. *5 Steps to a 5: AP Computer Science.* New York: McGraw-Hill, 2005.

Litvin, Maria. *Be Prepared for the AP Computer Science Exam in Java*. 2nd ed. Andover, Mass.: Skylight Publishing, 2006.

Litvin, Maria, and Gary Litvin. *175 Multiple-Choice Questions in Java.* Andover, Mass.: Skylight Publishing, 2005.

Teukolsky, Roselyn. *Barron's How to Prepare for the AP Computer Science Advanced Placement Examination (Java Version).* 2nd ed. Hauppauge, N.Y.: Barron's Educational Series, 2003.

Trees, Fran, and Cay Horstmann. *Advanced Placement Study Guide to Accompany Cay Horstmann's Java Concepts Study Guide.* 4th ed. Hoboken, N.J.: Wiley, 2005.

Exam Tips for Students

Discuss test-taking strategies with your students. You may print and distribute the tips I've shared with my class (see below). This document can also be found at:

[AP Central > Computer Science A (or AB) > Exam Tips]

Tips for Students Taking AP Computer Science Exams

Before the Exam

1. Allow more than one night for review. If you have questions, you'll have time to ask someone else for help.
2. Review the information about course content in the *AP Computer Science Course Description*.
   a) The Topic Outline lists the major topics covered on the AP CS Exams.

b) The Commentary on the Topic Outline explains the outline more fully and gives some Java-language examples.
c) The AP Computer Science Java Subset (appendix A) lists Java language features that will be tested.
d) The Standard Java Library Methods Required for the A (appendix B) or AB (appendix C) Exam are important: you will be given reference guides to the classes when you take the exam, but being aware of the existence of some of the less common class methods might save you time.
e) Implementation classes for linked list and tree nodes are covered in CS AB (appendix D).

If you find anything in these sections that is not familiar to you, ask your teacher for clarification.

3. Review the case study, both the code and the narrative.
4. Review all of your tests from your AP CS course.
5. Work sample problems from the Course Description and AP CS Exam review books.
6. Don't spend time memorizing specific code, e.g., searches and sorts. Recognizing a binary search will help you analyze it more quickly, but no questions will require regurgitation of specific code.
7. Get a good night's sleep before the exam.
8. Bring pencils and a watch to the exam.

General Tips

1. Problems are usually placed on the exam in order of perceived difficulty. However, some topics that are generally considered difficult might be easier for you than others, so make sure you allow yourself time to try each problem.
2. If you get stuck on a problem, don't waste time. Circle the number in the exam booklet and come back to it later.
3. Glance at your watch periodically, especially if you find yourself having difficulties or getting distracted. That should help to get you back on track.
4. Consult the *Quick Reference* documents during the exam. Be sure to call all methods with the correct parameters.
5. Use the case-study code as a syntax and logic guide for all problems, not just case-study problems. You can find examples of almost every kind of syntax you'll need to write or interpret.

Multiple-Choice Questions

1. Know when to guess. If you can eliminate one or two of the choices, the odds are in your favor.
2. Work backwards, if possible, to eliminate some answers.
3. Read all the answers before making your final decision.

Free-Response Questions

1. Make sure you have answered the problem.
   a) As you first read the problem, underline, circle, or star important details.
   b) Watch out for "You will receive no credit if ..." or "You will not receive full credit if ..."
   c) Reread the problem definition after you think you've finished writing the code, paying close attention to the details you marked.

   d) Write your code to satisfy the formal definition of the problem, but check your code with all the examples provided. (They are usually chosen by the exam developers to illustrate some of the special cases, and you're wasting a valuable resource if you ignore them.)
   e) Sometimes the preconditions and postconditions outline the algorithm to solve the problem. Don't ignore them.
2. Never let the beginning of a problem prevent you from getting points at the end.
   a) Most problems have multiple sections, and sometimes the later sections are easier than the earlier ones.
   b) You will frequently be told that you may make calls to one or more methods that you were asked to write in previous sections of the problem. The instructions might say, for example, "Assume that [method] works as specified, regardless of what you wrote in part (a). Solutions that reimplement functionality provided by this method, rather than invoking it, will not receive full credit." The best solution will probably include a call to the method written in part (a). You may earn credit for calling the method correctly, even if the code for the method you wrote for part (a) was incorrect or blank. However, if you copy the code from that method into the new section, even if it's completely correct, you will not earn full credit for your code.
3. Aim for clarity in your programming.
   a) Organize, indent, assign meaningful variable names, and write neatly. (If you accidentally omit a curly brace, but your indentation clearly conveys your intent, you'll be given the benefit of the doubt.)
   b) Remember that humans will be grading your work. Be legible.
   c) Commenting will not help your score, except perhaps if your code is confusing and the comment enlightens the Exam Reader about what you were trying to accomplish. (However, comments are not a substitute for correct code!)
4. Never give a "recipe" for the answer. If you know how to do even part of the problem, write code for the part you know. Partial credit is often awarded for having the correct loop bounds, initialization in the presence of an attempt to sum values in an array, and so on. Credit is not given for writing a description of what you would do if you had time.
5. Occasionally, you'll be asked to "justify your answer." You generally won't earn any credit without a justification—a sentence or phrase is often sufficient.
6. If you write two solutions to a problem, cross out the one you don't want to have scored. Don't waste time erasing it; a simple X is sufficient. (If a Reader finds two solutions to a problem, and neither one is crossed out, the first one will be scored.)
7. Don't waste time copying the method header or pre- and postconditions; start writing the code just below them.
8. If you run out of room:
   a) Find a blank page and continue your code.
   b) Indicate clearly what you've done, both on the page where you started the response and on the page where you continued it.
9. Avoid using classes that aren't specifically given to you as part of the exam and aren't part of the Java language or the case study. While you may have developed or used other classes in your AP course, don't use them in your responses on the AP CS Exam.

Good luck, and do your best!