2021 AP CS A Exam
Exam Overview

APCSA AP Exam

Exam Date and Time:

Tuesday, June 1 at 4PM Eastern Time at home on the Digital Testing App.

The Digital Testing App can be installed on desktops and laptops. Personal Chromebooks are not allowed. Only school-managed Chromebooks which already have the App installed can be used.

Please sign in 30 minutes before 4PM to do preliminary registration.

Exam Directions(from actual Exam)

This directions are the same as on your actual exam. Read it now and don't waste time reading it on the exam!

The AP Computer Science A exam is 3 hours long. Section I lasts 90 minutes and has 40 multiple-choice questions. Section II lasts 90 minutes and has 4 free-response questions.

The **Java Quick Reference** is available by clicking the **Reference** button in the toolbar at the upper right of the digital application screen.

IMPORTANT!!

Please note that you are responsible for pacing yourself. The clock will turn red when 5 minutes remain, but the proctor or app will not give you any other time updates.

Do not spend too much time on any one question, but remember that you cannot go back to a question once you've moved on to the next question.

You may use scratch paper for notes and planning, but credit will only be given for responses entered in this application.

Section Directions–Section I

Section I: 90 minutes, 40 multiple-choice questions.

Determine the answer to each of the following questions or incomplete statements, then select the best answer to each question.

Please note:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Assume that declarations of variables and methods appear within the context of an enclosing class.
- Assume that method calls that are not prefixed with an object or class name and are not shown within a complete class definition appear within the context of an enclosing class.
- Unless otherwise noted in the question, assume that parameters in method calls are not null and that methods are called only when their preconditions are satisfied.

Section Directions–Section II

Section II: 90 minutes, 4 free-response questions.

In this section, you will answer questions 1, 3, and 4 in multiple parts, which will be shown on separate screens. **You may move back and forth among the parts of each of these questions** while you are answering that particular question. **Once you submit your response to the last part of the question and move on to the next question, you cannot go back to any part of the question you just finished.**

You will answer Question 2 in a single screen, and **once you submit your response to Question 2, you will not be able to go back to it.**

You are advised to spend approximately 22 minutes answering each question.

Show all your work. Remember that program segments are to be written in Java.

Please note:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not null and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

For prompts that reference a specific question part, you can click on the » symbol shown in the prompt to automatically scroll to the referenced location.

Section I: Multiple Choice
40 Questions | 1 Hour 30 Minutes | 50% of Exam Score

- The multiple-choice section includes mostly individual questions, occasionally with 1–2 sets of questions (2 questions per set).
- Computational Thinking Practices 1, 2, 4, and 5 are all assessed in the multiple-choice section.

Section II: Free Response

4 Questions | 1 Hour 30 Minutes | 50% of Exam Score

- All free-response questions assess Computational Thinking Practice 3: Code Implementation, with the following focus
  - **Question 1:** Methods and Control Structures—Students will be asked to write program code to create objects of a class and call methods, and satisfy method specifications using expressions, conditional statements, and iterative statements.
  - **Question 2:** Classes—Students will be asked to write program code to define a new type by creating a class and satisfy method specifications using expressions, conditional statements, and iterative statements.

Note that Question 2 will be answered completely on one screen.
Questions 1, 3 and 4 have multiple parts and for each question, you can move among the parts but once you submit your answer, you cannot go back.

- **Question 3:** Array/ArrayList—Students will be asked to write program code to satisfy method specifications using expressions, conditional statements, and iterative statements and create, traverse, and manipulate elements in 1D array or ArrayList objects.

- **Question 4:** 2D Array—Students will be asked to write program code to satisfy method specifications using expressions, conditional statements, and iterative statements and create, traverse, and manipulate elements in 2D array objects.

Even though not explicitly listed, Strings are very popular on the Free Response part of the AP exams. Know the string methods listed on the reference sheet really well!

Java Reference Sheet

or here:

https://apcentral.collegeboard.org/pdf/ap-computer-science-a-java-quick-reference.pdf

Strings

Strings and String methods have shown up every year. Likely it will show up on this year's AP Exam.

You may have to work with 1D array or ArrayList of Strings.

You should know how to use **substring(int beg, int end), substring(int beg), indexOf, equals, length and compareTo** methods. See the College Board reference sheet.

At Home Test

We're confident that the vast majority of AP students will follow the rules for taking the exams.

For the small number of students who may try to gain an unfair advantage, we have a comprehensive and strict set of protocols in place to prevent and detect cheating.

While some of these practices are confidential to maximize their effectiveness, students and education professionals can learn more about our security measures.

At Home Test

At a minimum, test takers should understand that those attempting to gain an unfair advantage will either be blocked from testing or their AP scores will be canceled, and their high school will be notified as will colleges or other organizations to which the student has already sent any College Board scores (including SAT® scores).

And they may be prohibited from taking a future Advanced Placement® Exam as well as the SAT, SAT Subject Tests™, or CLEP® assessments.

Collaboration

Collaborating with others is *not* considered acceptable open notes: AP Exams give you an opportunity to show *your* mastery of a subject—not someone else's. It is strictly forbidden to give or receive aid during the exam. Any students found using the work of others, exchanging or sharing information on exam topics, collaborating via any online platform, or soliciting tips for problem-solving approaches will be investigated for violating exam security.

Free Response

General Strategies for FRQS:

1) Read the questions before you read the code! This will help you focus.

2) Unless a question specifically addresses efficiency, focus on clear, simple code rather than complicated, efficient code.

3) Comments are not necessary. Use only to organize your thoughts.

4) Pay attention to the code they give you:

- Do you have more than one class? Use it!
- Did they give you methods? You will need them!
- Are there instance variables that you should use? If so, use them. Do not re-declare.
- Only use accessor and mutator methods to access private instance variables.
- Do not re-write or change method headers.
- Use parameters (names and types) as given

Free Response

General Strategies for FRQS:

5) If part (b) references a method written in part (a), you will likely need to use this method. If it is not clear to you how, stop and think before you write.

6) Pay attention to the return type. If it is not void, be sure to include a return statement!

7) If a method comment says /* Implementation not shown */. You do not to implement the method. But you'll likely need to use the method in your code at least once.

Free Response

General Strategies for FRQS:

8) Write down some code for every question! Leave Nothing Blank! Get partial credit if not full credit!

- does it need a loop?
- an if statement?
- return statement or assignment statement?

Example 1:

```
public double funMethod(){
     double result;
     …
     return result;
}
```

More Examples

Leave Nothing Blank! Get partial credit if not full credit!

```
public int[] funMethod(int length){
     int[] list = new int[length];
     …
     return list;
}
public ArrayList<Bug> funMethod(){
     ArrayList<Bug> buggies = new ArrayList<Bug>();
     …
     return buggies;
}
```

More Examples

Leave Nothing Blank! Get partial credit if not full credit!

```
public boolean funMethod(int something){
      …
      if(something > somethingElse)
            return true;
      // must have a default return; every return can't be
      // locked in an if
      return false;
}
```

Question 3: Arraylist/1D Array

ArrayList/1D Arrays

We will first prepare for the FRQs by reviewing ArrayList and 1D arrays. This will be Question 3 on the Free Response.

From previous years' exams, it is likely that Question 3 is an ArrayList question instead of/in addition to an 1D array question.(ArrayList or both, unlikely to be just a 1D array question)

It's very common to have a class(e.g Student) and another class which contains either a 1D array of Student objects or an ArrayList of Student objects.

```
public class Student {
   …
   public String getName(){..}
   public double getGpa(){…}
}
```

ArrayList/1D Arrays

```
public class Course {
   private ArrayList<Student> students;

     public Course() {
       students = new ArrayList<Student>();
          ...
     }
}

OR
public class Course {
   private Student[] students;

     }
}
```

Traversing 1D Array vs. ArrayList

Arrays:

```
double sum = 0;
for(int i = 0; i < students.length; i++){
     sum += students[i].getGpa();
}
```

ArrayLists:

```
double sum = 0;
for(int i = 0; i < students.size(); i++){
     sum += students.get(i).getGpa();
}
```

Traversing 1D Array vs. ArrayList

Use for each loops if you are only traversing the arraylist and not removing items from it.

Both Arrays and Arraylists:

```
double sum =0;
for(Student s: students){
     sum += s.getGpa();
     // notice no [i] and no get(i)!!
}
```

ArrayList

Remember for Arraylists that if you remove an item, items to the right of it will shift left!

And if you add an item to the list, items will shift right to accommodate!

Make sure to take this into account when you are adding or removing items.

ArrayList/1D Array Tips

When using the ArrayList method remove(), remember you can only REMOVE BY INDEX (not by object), so you must use a traditional for loop (and NOT a for each loop).

```
for(int i=0; i<list.size(); i++){
        if(list.get(i).getScore() <= 50)
                list.remove(i);
                i--;//AND don't forget to decrement the index!
}
//OR traverse the list backwards, i.e.
for(int i=list.size()-1, i>=0, i--){
        if(list.get(i).getScore() <= 50)
                list.remove(i--);
}
```

ArrayList/1D Array Tips

Avoid array and ArrayList out-of-bounds exceptions! Especially when comparing consecutive elements.

Example: You need to compare consecutive elements to determine if the values are increasing.

```
public boolean increasing(int[] numbers){
     for(int i=0; i < numbers.length-1; i++)
          if(numbers[i] >= numbers[i+1])
               return false;
     return true;
}
```

Finding the smallest or largest

When you need to return an element in a list (i.e. largest, smallest,…), assign the initial element to the **first one** in the list.

Example : You need to find the Dog with the most fleas.

```
//returns the Dog object in pack with the most fleas
public Dog mostFleas(ArrayList<Dog> pack){
     Dog most = pack.get(0);
     for(Dog dog : pack)
          if(dog.getNumFleas() > most.getNumFleas())
               most = dog;
     return most;
}
```

Manipulating an Array using Arraylist

An ArrayList can be helpful when you are trying to manipulate an array (i.e. remove items, rearrange in random order, etc.).

Example:

If you need to remove several objects from an array based one some condition, a strategy would be to save all the objects to an ArrayList and remove the objects before storing the remaining objects back to the array.

See code on the next slide.

Manipulating an Array using Arraylist

```
//returns a new array with bad things removed
public Thing[] removeBadThing(Thing[] widgets){
     ArrayList<Thing> good = new ArrayList<Thing>();
     for(Thing wid : widgets){
          if(!wid.isBad()) // i.e good
               good.add(wid);
     }
     Thing[] goodOnes = new Thing[good.size()];
     for(int i=0; i < goodOnes.length; i++)
          goodOnes[i] = good.get(i);
     return goodOnes;
}
```