2021

AP CollegeBoard

AP® Computer Science A

Sample Student Responses and Scoring Commentary

Inside:

Free Response Question 3

- ☑ Scoring Guideline
- ☑ Student Samples
- ☑ Scoring Commentary

Applying the Scoring Criteria

Apply the question scoring criteria first, which always takes precedence. Penalty points can only be deducted in a part of the question that has earned credit via the question rubric. No part of a question (a, b, c) may have a negative point total. A given penalty can be assessed only once for a question, even if it occurs multiple times or in multiple parts of that question. A maximum of 3 penalty points may be assessed per question.

1-Point Penalty

v) Array/collection access confusion (`[] get`)
w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check)
x) Local variables used but none declared
y) Destruction of persistent data (e.g., changing value referenced by parameter)
z) Void method or constructor that returns a value

No Penalty

- Extraneous code with no side-effect (e.g., valid precondition check, no-op)
- Spelling/case discrepancies where there is no ambiguity*
- Local variable not declared provided other variables are declared in some part
- `private` or `public` qualifier on a local variable
- Missing `public` qualifier on class or constructor header
- Keyword used as an identifier
- Common mathematical symbols used for operators (× • ÷ ≤ ≥ <> ≠)
- `[]` vs. `()` vs. `<>`
- `=` instead of `==` and vice versa
- `length/size` confusion for array, `String`, `List`, or `ArrayList`; with or without `( )`
- Extraneous `[]` when referencing entire array
- `[i,j]` instead of `[i][j]`
- Extraneous size in array declaration, e.g., `int[size] nums = new int[size];`
- Missing `;` where structure clearly conveys intent
- Missing `{ }` where indentation clearly conveys intent
- Missing `( )` on parameter-less method or constructor invocations
- Missing `( )` around `if` or `while` conditions

Spelling and case discrepancies for identifiers fall under the "No Penalty" category only if the correction can be **unambiguously inferred from context, for example, "ArayList" instead of "ArrayList". As a counterexample, note that if the code declares "`int G=99, g=0;`", then uses "`while (G < 10)`" instead of "`while (g < 10)`", the context does **not** allow for the reader to assume the use of the lower case variable.*

Question 3: Array / ArrayList — 9 points

Canonical solution

(a) — **3 points**

```
public void addMembers(String[] names, int gradYear)
{
   for (String n : names)
   {
      MemberInfo newM = new MemberInfo(n, gradYear, true);
      memberList.add(newM);
   }
}
```

(b) — **6 points**

```
public ArrayList<MemberInfo> removeMembers(int year)
{
   ArrayList<MemberInfo> removed = new ArrayList<MemberInfo>();

   for (int i = memberList.size() - 1; i >= 0; i--)
   {
      if (memberList.get(i).getGradYear() <= year)
      {
         if (memberList.get(i).inGoodStanding())
         {
            removed.add(memberList.get(i));
         }
         memberList.remove(i);
      }
   }
   return removed;
}
```

(a) `addMembers`

	Scoring Criteria	Decision Rules	
1	Accesses all elements of `names` (*no bounds errors*)	Responses **will not** earn the point if they fail to access elements of the array, even if loop bounds are correct	**1 point**
2	Instantiates a `MemberInfo` object with name from array, provided year, and good standing		**1 point**
3	Adds `MemberInfo` objects to `memberList` (*in the context of a loop*)	Responses **can** earn the point even if they instantiate `MemberInfo` objects incorrectly	**1 point**
		Total for part (a)	**3 points**

(b) `removeMembers`

	Scoring Criteria	Decision Rules	
4	Declares and initializes an `ArrayList` of `MemberInfo` objects	Responses **will not** earn the point if they initialize the variable with a reference to the instance variable	**1 point**
5	Accesses all elements of `memberList` for potential removal (*no bounds errors*)	Responses **will not** earn the point if they • fail to use `get(i)` • fail to attempt to remove an element • skip an element • throw an exception due to removing	**1 point**
6	Calls `getGradYear` or `inGoodStanding`	Responses **can** still earn the point even if they call only one of the methods Responses **will not** earn the point if they • ever include parameters in either method call • ever call either method on an object other than `MemberInfo`	**1 point**
7	Distinguishes any three cases, based on graduation status and standing	Responses **will not** earn the point if they fail to behave differently in all three cases	**1 point**
8	Identifies graduating members	Responses **can** still earn the point even if they • fail to distinguish three cases • fail to access standing at all • access the graduating year incorrectly Responses **will not** earn the point if they confuse `<` and `<=` in the comparison	**1 point**
9	Removes appropriate members from `memberList` and adds appropriate members to the `ArrayList` to be returned	Responses **can** still earn the point even if they • call `getGradYear` or `inGoodStanding` incorrectly • access elements of `memberList` incorrectly • initialize the `ArrayList` incorrectly • fail to return the list that was built (*return is not assessed*) Responses **will not** earn the point if they • fail to declare an `ArrayList` to return • fail to distinguish the correct three cases, with the exception of confusing the `<` and `<=` in the comparison	**1 point**
		Total for part (b)	**6 points**

Question-specific penalties
None

Total for question 3 9 points

Begin your response to each question at the top of a new page.

a)

```
Public void addMembers(String [] names, int gradYear)
{ for(int j=0; j< names.length-1; j++)
  {
    MemberInfo x= new MemberInfo(names[j], gradYear, true);
    MemberList.add(x);
  }
}
```

b

Begin your response to each question at the top of a new page.

b)

```
public ArrayList<MemberInfo> removeMembers(int year)
{ ArrayList<MemberInfo> ret = new ArrayList<MemberInfo>();
  for(MemberInfo c : memberList)
  {
    if(c.getGradYear <= year && c.inGoodStanding == true)
    {
      ret.add(c);
    if(c.getGradYear <= year) false)
      memberList.remove(c);
  }
  return ret;
}
```

Question 1 ○ Question 2 ○ Question 3 ● Question 4 ○

Begin your response to each question at the top of a new page.

```
Public void addMembers(String[] names, int gradYear)
{
  For(int i=0; i< names.length; i++)
  {
    String MemberName = names[i];
    MemberList.add(MemberName, MemberName.getGradYear, true)
  }
}
```

Part a

Begin your response to each question at the top of a new page.

```
Public ArrayList<MemberInfo> removeMembers(int year)
{ ArrayList removed = new ArrayList();
  for(int i = 0; i < MemberList.size(); i++)
  {
    if(MemberList.get(i).getGradYear >= year)
    {
      if(MemberList.get(i).inGoodStanding == true)
      {
        removed.add(MemberList.get(i));
        MemberList.remove(i);
      }
      else
      {
        MemberList.remove(i);
      }
    }
  }
  return removed;
}
```

Part b

Begin your response to each question at the top of a new page.

a)

```
public void addMembers(String[] names, int gradYear)
  {
   for (int i=0; i<names.length; i++)
       {
        memberList.add(new MemberInfo(names[i], gradYear, true);
       }
  }
```

b)

```
public ArrayList<MemberInfo> removeMembers(int year)
   {
    ArrayList<MemberInfo> removed;
    for (int i=0; i<memberList.size(); i++)
        {
        if (memberList.get(i).getGradYear()==year)
            {
             removed.add(memberList.get(i));
             memberList.remove(i);
            }
        }
    return removed;
   }
```

Question 3

Overview

This question tested the student's ability to:

- Write program code to create objects of a class and call methods.
- Write program code to satisfy methods using expressions, conditional statements, and iterative statements.
- Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects.

This question involved the manipulation of both a one-dimensional array containing `String` values and an `ArrayList` containing `MemberInfo` objects. Students were expected to write two methods in the enclosing `ClubMembers` class, making use of its `ArrayList` instance variable as well as two methods from the `MemberInfo` class.

In part (a) students were expected to write a loop to access each element of an array parameter. Inside the loop, students were expected to: (1) Construct a `MemberInfo` object using the `new` keyword and three parameters: a name from the array, `gradYear`, and `true`, in that order; (2) Add the constructed `MemberInfo` object to the `ClubMembers` instance variable `memberList`.

In part (b) students were asked to develop an algorithm to: (1) Identify club members who have graduated and are in good standing and add those club members to an `ArrayList` to be returned; (2) Remove from `memberList` those club members who have graduated, regardless of whether or not they are in good standing; and (3) Leave club members who have not yet graduated in `memberList`. Students had to create an `ArrayList` of `MemberInfo` objects to be returned and write a loop to access each element of the given `ArrayList` instance variable. Inside the loop, students had to call `getGradYear` and correctly compare the `int` return value to the `year` parameter. They also had to call `inGoodStanding` and use the `boolean` return value appropriately.

Sample: 3A
Score: 8

In part (a) point 1 was earned by accessing all elements of `names` with no bounds errors. The response uses a traditional `for` loop with correct lower and upper bounds. Within the context of the loop, the response accesses `names[j]`. Point 2 was earned by instantiating a `MemberInfo` object by using the keyword `new` and the correct parameters. Point 3 was earned by adding `MemberInfo` objects to `memberList` in the context of a loop. The response correctly calls the `add` method for `memberList` with the parameter of an instantiated `MemberInfo` object that has been assigned to a separate variable.

In part (b) point 4 was earned by correctly declaring and initializing an `ArrayList` of `MemberInfo` objects. Point 5 was not earned because the response calls the `remove` method within an enhanced `for` loop, which causes an exception to be thrown. Point 6 was earned because there are correct calls to both the `getGradYear` and `inGoodStanding` methods. Omitting the `()` on each method call falls into the "No Penalty" category. Point 7 was earned because the response distinguishes three cases, based on graduation status and standing. The three identified cases are: (1) members who have graduated in good standing; (2) members who have graduated but are not in good standing; and (3) members who have not yet graduated. Point 8 was earned by identifying graduating members. The response correctly identifies graduating members by checking if the graduation year returned by the method call is less than or equal to the method's `year` parameter. Point 9 was earned because the response first correctly identifies graduating members in good standing and adds them to the `ArrayList` to be returned, then identifies graduating members and removes them from `memberList`. Members who have not yet graduated remain in `memberList`. Note that the

Question 3 (continued)

faint `}` at the end of the loop may have been erased, but because the indentation of the response clearly conveys intent, the possibly missing `}` is one of the minor errors for which no penalty is assessed. (See the "No Penalty" category on page 1 of the Scoring Guidelines for a complete list.)

Sample: 3B
Score: 5

In part (a) point 1 was earned by accessing `names[i]` in a traditional `for` loop with correct bounds. Point 2 was not earned because the response makes no attempt to instantiate a `MemberInfo` object using the keyword `new` and the correct parameters. Point 3 was not earned because the response does not add a `MemberInfo` object to `memberList` within a loop.

In part (b) point 4 was earned because the `ArrayList` is declared and initialized correctly. The response does not declare an object type for the `ArrayList` but it is not always required in a statement of this form; current versions of Java permit the angle-bracketed types to be omitted in certain circumstances when the type can be inferred. When writing a method that returns an `ArrayList<MemberInfo>`, all of the following `ArrayList` declarations and instantiations will work and receive credit:

```
ArrayList<MemberInfo> list1 = new ArrayList<MemberInfo>();
ArrayList<MemberInfo> list2 = new ArrayList();
ArrayList list3 = new ArrayList<MemberInfo>();
ArrayList list4 = new ArrayList();
ArrayList<MemberInfo> list5 = new ArrayList<>();
ArrayList list6 = new ArrayList<>();
ArrayList ArrayList = new ArrayList();
```

Point 5 was not earned because the response does a forward traversal of the `ArrayList` with a call to the `remove` method and does not account for the shift left of elements. Point 6 was earned because the calls to the `getGradYear` and `inGoodStanding` methods are correct. Point 7 was earned because the response distinguishes three cases based on graduation status and standing. The response behaves differently in all three cases. Point 8 was not earned because the response incorrectly determines if the graduation year is greater than or equal to `year`. Point 9 was earned because, following the use of an incorrect operator, a subset of the graduates is removed from `memberList` and some of the removed members are added appropriately to the `ArrayList` to be returned, based on standing.

Sample: 3C
Score: 4

In part (a) point 1 was earned by accessing `names[i]` in a traditional `for` loop with correct bounds. Point 2 was earned by instantiating a `MemberInfo` object using the keyword `new` and the correct parameters. Point 3 was earned by adding `MemberInfo` objects to `memberList` within a loop.

In part (b) point 4 was not earned because even though the `ArrayList` is declared correctly, it is not initialized as an `ArrayList` of `MemberInfo` objects. Point 9 can still be earned because only a declaration is required for that point. Point 5 was not earned because the response does a forward traversal of the `ArrayList` with a call to the `remove` method and does not account for the shift left of elements. Point 6 was earned because there is a correct call to either the `getGradYear` or `inGoodStanding` methods. In this case, a call to `getGradYear` correctly occurs on a `MemberInfo` object. Point 7 was not earned because the response fails to access standing and because the response only distinguishes two cases. Point 8 was not

Question 3 (continued)

earned because the response incorrectly identifies graduating members by checking if the graduation year `== year`. Point 9 was not earned because the response does not distinguish the correct three cases based on graduation status and standing.