AP® COMPUTER SCIENCE A
2013 SCORING GUIDELINES

Question 1: SongList

Part (a)	`getDownloadInfo`	**4 points**

Intent: *Search download list for requested title and return matching* `DownloadInfo` *object if found.*

+1 Accesses all necessary entries in `downloadList` (*no bounds errors*)

+3 Identifies and returns matching entry in `downloadList`, if it exists

- **+1** Calls `getTitle` on `DownloadInfo` object from `downloadList`
- **+1** Checks for equality between title from list object and `title` parameter (*must use String equality check*)
- **+1** Returns reference to matching object if present; `null` if not (*point not awarded for early return*)

Part (b)	`updateDownloads`	**5 points**

Intent: *Update* `downloadList` *with information from list of titles*

+1 Accesses all entries in `titles` *(no bounds error for* `titles`*)*

+1 Calls `getDownloadInfo(`*`title`*`)` to determine whether title from `titles` list exists in `downloadList`

+1 Increments the count in matching `DownloadInfo` object if title is in `downloadList`

+1 Constructs new `DownloadInfo` object (with correct information) if title is not in `downloadList` *(point not awarded if incremented at time of construction)*

+1 Adds constructed object to end of `downloadList` if title is not in `downloadList` *(point not awarded if added more than once)*

Question-Specific Penalties

-1 (g) Uses `getLength`/`getSize` for ArrayList size

-2 (v) Consistently uses incorrect array name instead of `downloadList`/`titles`

-1 (z) Attempts to return a value from `updateDownloads`

1 Aa

Complete method `getDownloadInfo` below.

```
/** Returns a reference to the DownloadInfo object with the requested title if it exists.
 *  @param title the requested title
 *  @return a reference to the DownloadInfo object with the
 *          title that matches the parameter title if it exists in the list;
 *          null otherwise.
 *  Postcondition:
 *    - no changes were made to downloadList.
 */
public DownloadInfo getDownloadInfo(String title)
{
    for (DownloadInfo info : downloadList)
        if (info.getTitle().equals(title))
            return info;

    return null;
}
```

3

Part (b) begins on page 8.

GO ON TO THE NEXT PAGE.

1 Ab

Complete method `updateDownloads` below.

```
/** Updates downloadList with information from titles.
 *  @param titles a list of song titles
 *  Postcondition:
 *    - there are no duplicate titles in downloadList.
 *    - no entries were removed from downloadList.
 *    - all songs in titles are represented in downloadList.
 *    - for each existing entry in downloadList, the download count is increased by
 *        the number of times its title appeared in titles.
 *    - the order of the existing entries in downloadList is not changed.
 *    - the first time an object with a title from titles is added to downloadList, it
 *        is added to the end of the list.
 *    - new entries in downloadList appear in the same order
 *        in which they first appear in titles.
 *    - for each new entry in downloadList, the download count is equal to
 *        the number of times its title appeared in titles.
 */
public void updateDownloads(List<String> titles )
{
    for (String s : titles)
    {
        if (getDownloadInfo(s) != null)
            getDownloadInfo(s).incrementTimesDownloaded()
        else
            downloadList.add(new DownloadInfo(s));
    }
}
```


GO ON TO THE NEXT PAGE.

1Ba

Complete method `getDownloadInfo` below.

```
/** Returns a reference to the DownloadInfo object with the requested title if it exists.
 *  @param title the requested title
 *  @return a reference to the DownloadInfo object with the
 *          title that matches the parameter title if it exists in the list;
 *          null otherwise.
 *  Postcondition:
 *  - no changes were made to downloadList.
 */
public DownloadInfo getDownloadInfo(String title)
```

```
public DownloadInfo getDownloadInfo(String title) {
    DownloadInfo download = null;
    for(int i = 0; i < downloadList.size(); i++) {
        if (downloadList.get(i).getTitle() == title)
        {
            download = downloadList.get(i);
        }
    }
    return download;
}
```

Part (b) begins on page 8.

GO ON TO THE NEXT PAGE.

1 Bb

Complete method updateDownloads bel

```
/** Updates downloadList with inf
 *  @param titles a list of song titl
 *  Postcondition:
 *    - there are no duplicate titles in downloadList.
 *    - no entries were removed from downloadList.
 *    - all songs in titles are represented in downloadList.
 *    - for each existing entry in downloadList, the download count is increased by
 *        the number of times its title appeared in titles.
 *    - the order of the existing entries in downloadList is not changed.
 *    - the first time an object with a title from titles is added to downloadList, it
 *        is added to the end of the list.
 *    - new entries in downloadList appear in the same order
 *        in which they first appear in titles.
 *    - for each new entry in downloadList, the download count is equal to
 *        the number of times its title appeared in titles.
 */
public void updateDownloads(List<String> titles )
```

```
public void updateDownloads(List<String> titles) {
  boolean exists = false;
  for(int i=0; i < titles.size(); i++) {
    for(int n=0; n < downloadList.size(); n++) {
      if (titles.get(i) == downloadList.get(n).getTitle()) {
        exists = true;
        downLoadList.get(n).incrementTimesDownloaded();
      }
    }
    if (!exists) {
      downloadList.add(new DownloadInfo(titles.get(i)
    }
    exists = false;
  }
}
```


GO ON TO THE NEXT PAGE.

1.Ca

Complete method `getDownloadInfo` below.

```
/** Returns a reference to the DownloadInfo object with the requested title if it exists.
 *  @param title the requested title
 *  @return a reference to the DownloadInfo object with the
 *          title that matches the parameter title if it exists in the list;
 *          null otherwise.
 *  Postcondition:
 *    - no changes were made to downloadList.
 */
public DownloadInfo getDownloadInfo(String title)
```

```
for(int i=0 ; i < downloadList.size() ; i++){
    if(downloadList.get(i).equals(title){
        return i ;}
    else {
        return null ;}}
```

Part (b) begins on page 8.

1 C b

Complete method updateDownloads below.

```
/** Updates downloadList with information from titles.
 *  @param titles a list of song titles
 *  Postcondition:
 *    - there are no duplicate titles in downloadList.
 *    - no entries were removed from downloadList.
 *    - all songs in titles are represented in downloadList.
 *    - for each existing entry in downloadList, the download count is increased by
 *        the number of times its title appeared in titles.
 *    - the order of the existing entries in downloadList is not changed.
 *    - the first time an object with a title from titles is added to downloadList, it
 *        is added to the end of the list.
 *    - new entries in downloadList appear in the same order
 *        in which they first appear in titles.
 *    - for each new entry in downloadList, the download count is equal to
 *        the number of times its title appeared in titles.
 */
public void updateDownloads(List<String> titles)
```

adding titles

checking

```
int t= titles.size();
int dd=downloadList.size
for(int i=0; i<t ; i++) {
  for(int j=0; j<dd ; j++) {
    if (downloadList.get().equals(titles.get(i))) {
      downloadList.get(i).incrementTimesDownloaded; }
    else {
      downloadList.add(titles.get(i)); } }
```

Unauthorized copying or reuse of any part of this page is illegal.

GO ON TO THE NEXT PAGE.

AP® COMPUTER SCIENCE A
2013 SCORING COMMENTARY

Question 1

Overview

This question involved object construction, list access and modification, and design and use of a helper method to search a list. In part (a), students were required to implement the method `getDownloadInfo` to search for an entry in the list instance variable having a title matching the parameter. This could be accomplished by traversing the list and using a string comparison to compare the parameter with the result of invoking `getTitle` on each entry. When a match was found, a reference to the object was to be returned. If the traversal completed with no match found, `null` was to be returned. In part (b), the students were required to implement `updateDownloads`, which modified entries in the list instance variable (and added to the list) based on a list parameter containing titles. Students were required to use `getDownloadInfo` in implementing this method. For each title in the parameter list, `getDownloadInfo` would be called with the title as argument. If the result was a reference to a matching `DownloadInfo` object, `incrementTimesDownloaded` would be called on that object. If the result was `null`, a new `DownloadInfo` object would be created for that title. This new object would be added to the end of the list instance variable. The result of this processing would be that the list instance variable would have been updated to include information about each title in the parameter list. Either the existing download count was incremented by one for each occurrence in the title list or, for a new title, a new representation was created with the count set to 1.

Sample: 1A
Score: 9

In part (a), all necessary elements of the `downloadList` instance variable are correctly accessed using an enhanced `for` (for-each) loop. At each iteration of the loop, the `getTitle()` method is applied to the currently selected element of `downloadList` and the title from the `downloadList` element is compared for equality with the `title` parameter using the `String equals()` method. If there is a match, the method immediately returns a reference to the currently selected element of `downloadList`. If the loop terminates without locating the target `title` in `downloadList`, the method correctly returns `null` indicating that the target `title` is not represented in `downloadList`. Part (a) earned 4 points.

In part (b), all elements of the `List` parameter `titles` are accessed using an enhanced `for` (for-each) loop. At each iteration, the `getDownloadInfo()` method is applied, as is required, to the currently selected element of `titles`. If the currently selected `titles` element exists in `downloadList`, then the count of the number of downloads of the `titles` element is incremented by calling the `incrementTimesDownloaded()` method on the matching element of `downloadList`. If the currently selected `titles` element does not exist in `downloadList`, then a new `DownloadInfo` object is constructed for the `titles` element, and this new object is correctly appended to `downloadList`. Part (b) earned 5 points.

Sample: 1B
Score: 6

In part (a), a local `DownloadInfo` reference, `download`, is created and initialized to `null`. All necessary elements of `downloadList` are accessed using an indexed `for` loop. At each iteration of the loop, the `getTitle()` method is applied to the currently selected `downloadList` element, but the attempted comparison for equality with the target `title` is incorrect because the `==` operator is inappropriate for `String` comparison. When an appropriate comparison would have succeeded, the reference to the currently selected `downloadList` element is assigned to

Question 1 (continued)

`download` to be returned when the loop is exited. On exiting the loop, a correct reference (either `null` or a reference to a matching element) is returned in `download`. This part loses only the "equal on title" point. Part (a) earned 3 points.

In part (b), a local variable `exists` is initialized to `false`. It is used as a flag to signal the presence in `downloadList` of a freshly downloaded title (from the `titles` parameter). Using a correct indexed `for` loop, each element of the `titles` parameter is separately selected for processing. Instead of using the `getDownloadInfo()` method as required, reimplementation of `getDownloadInfo()` is attempted to determine the existence of the currently selected `titles` element in `downloadList`. The reimplementation does so using a nested indexed `for` loop to traverse `downloadList`. The reimplementation fails because of an unsuitable comparison (==) of the currently selected `titles` element and the `downloadList` element title, thereby losing the "increment matching" point in addition to the "getDownloadInfo call" point. When the matching would otherwise succeed, the flag `exists` is set to `true` to indicate a successful match, and the count of the number of downloads is incremented via a correct call of `incrementTimesDownloaded()`. On exiting the `downloadList` traversal, the flag `exists` is queried. If `exists` is `false` (indicating that no match occurred during the traversal), then a new `DownloadInfo` object is created to store the freshly downloaded song title from the `titles` parameter; this new object is correctly appended to `downloadList`. The flag is reset to `false` in preparation for the following traversal. Part (b) earned 3 points.

Sample: 1C
Score: 3

In part (a), all necessary elements of `downloadList` are accessed using an indexed `for` loop. The method fails to call `getTitle()` on `downloadList.get(i)`, losing the "getTitle" rubric point. The comparison with the `title` parameter using `equals()` is appropriate and correct. The "returns" point is lost because the method returns an index `i` rather than the object reference and the method otherwise always incorrectly returns `null` at the very first iteration. Thus, part (a) earns only the "access" and "equal on title" points. Part (a) earned 2 points.

In part (b), local variables `t` and `d` are declared and initialized with the sizes of the `titles` parameter and the `downloadList` instance variable respectively. An indexed `for` loop correctly accesses all elements of the `List` parameter, `titles`. The method does not call `getDownloadInfo()` to locate the currently selected title, losing the "getDownloadInfo call" point. The attempted reimplementation of matching the currently selected `title` from `titles` with a `downloadList` element fails because the method omits the necessary call to `getTitle()` on the `downloadList` element that it is attempting to match. The solution thus loses the "increment matching" point. There is never an attempt to create a new `DownloadInfo` object when a new `title` should be added to `downloadList`, and thus both the "construct" and "append" points are lost. The only point earned is for a correct "access all." Part (b) earned 1 point.