1 Explain why the following code fragment does not work as intended. (3)

```
ArrayList names = new ArrayList();
names.add("john");
names.add("greg");
ArrayList initials = new ArrayList();
for (int i = 0; i < names.size(); i++)
  initials.add(names.get(i).substring(0, 1));
```

2 Write a method that takes an `ArrayList<String>` as its sole parameter and returns a new `ArrayList<String>` in which the elements from the given `ArrayList` are stored in reverse order. Your method should not change the original list. (3)

3 Write a method that removes the smallest value from a given `ArrayList<Integer>`. (5)
Hint: The `Integer` class has a `compareTo()` method.

4 Can an `ArrayList<Object>` be an element of itself? Test this hypothesis and explain your results. (5)

5 The following program code is designed to remove all instances of the word "hello" from a populated `ArrayList` of `Strings` called `words`. (10)

```
int i = 0;
while (i < words.size()) {
  if ("hello".compareTo(words.get(i)) == 0)
    words.remove(i);
  i++;
}
```

(a) Explain why the above code fragment does not work as intended.

(b) Without changing any of the existing lines of code, add or insert a single line to enable the code fragment to work as intended.

6 *Failing Silently.* A method or program "failing silently" means that it fails to do what it was designed to without producing an error or any other message. (20)

(a) Create the subclass of `ArrayList`, `SilentArrayList`, that will override each method of `ArrayList` that requires an `index` value and fails silently if that value is not a valid index of the `ArrayList`.
Hint: Remember that the keyword `super` can be used to access methods of the super class.

(b) Describe a situation where failing silently might be the desired outcome for a method or program.