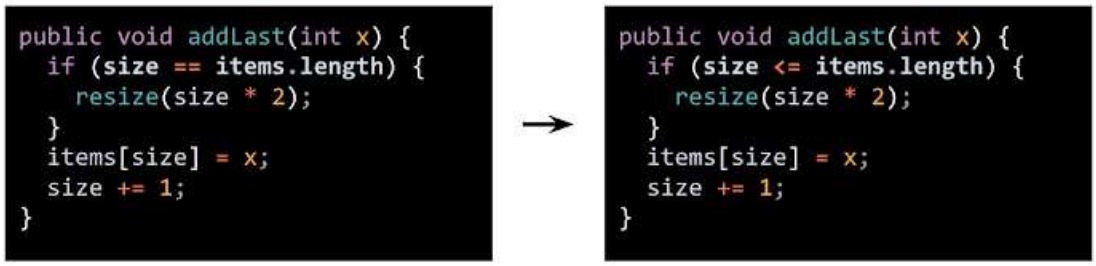
In addLast, suppose we instead decided to resize when **size <= items.length** (instead of **==**). How would this change our AList's behavior? * 2 points

Select all that apply.



```
public void addLast(int x) {
  if (size == items.length) {
    resize(size * 2);
  }
  items[size] = x;
  size += 1;
}
```

→

```
public void addLast(int x) {
  if (size <= items.length) {
    resize(size * 2);
  }
  items[size] = x;
  size += 1;
}
```

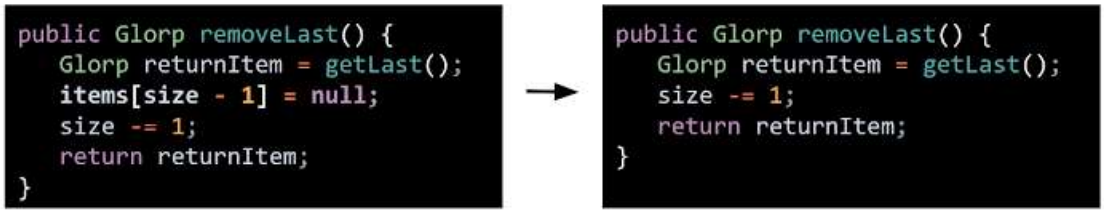
- [ ] The AList will no longer consistently return the correct values for get().
- [ ] The AList will run faster because we ensure there's always space to add new items.
- [ ] The AList will take up significantly more space in memory.
- [ ] The AList will run slower because it has to call System.arrayCopy more frequently.

Why does multiplicative resizing work better than additive resizing? * 1 point

- ( ) For sufficiently large values, multiplication is a faster operation than addition.
- ( ) It ensures we don't use as much space in memory to store our array.
- ( ) We avoid needing to copy the whole array as frequently.

In our Generic AList, what would happen if we forgot to null out the removed item during removeLast? * 2 points

Select all that apply.



- [ ] The old Glorp object would overwrite the next Glorp object we try to add to our list
- [ ] size() will no longer return the correct value
- [ ] The old Glorp would take up space in memory even though we no longer need it
- [ ] Running removeLast would result in a NullPointerException

A copy of your responses will be emailed to yiyunchen@berkeley.edu.

Submit

Clear form