Which of the following are consequences of having both **AList** and **SLList** implementing the **List61B** interface? * 2 points

Select all that apply.

```
public static String longest(List61B<String> inputList) {
        int maxDex = 0;
        for (int i = 0; i < inputList.size(); i+=1 ) {
        ......
}
```

- ☐ Both AList and SLList will support the same set of functionalities/methods (ex: addFirst, addLast, getFirst, etc.)
- ☐ Both AList and SLList can be passed into the "longest" method (see above) as a parameter.
- ☐ AList and SLList will have the same implementations for all of the functionalities/methods listed in the interface.

What will happen to the code below if SLList does **not** implement the List61B interface? * 1 point

```
public static void main(String[] args) {
    List61B<String> someList = new SLList<String>();
    someList.addFirst("elk");
}
```

- ○ Will not compile.
- ○ Will compile, but will cause an error at runtime on the new line.
- ○ When it runs, an SLList is created and its address is stored in the someList variable, but it crashes on someList.addFirst() since the List interface doesn't implement addFirst.
- ○ When it runs, an SLList is created and its address is stored in the someList variable. Then the string "elk" is inserted into the SLList referred to by addFirst.

What is the static and dynamic type of "**a**" ? * 1 point

```
public static void main (String[] args) {
        Dog d = new Corgi();
        Animal a = d;
        d = new Samoyed();
}
```

- ○ static: Animal, dynamic: Corgi
- ○ static: Animal, dynamic: Samoyed
- ○ static: Dog, dynamic: Corgi
- ○ static: Dog, dynamic: Samoyed

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

Submit   Clear form