Homework 10

Q1)(E)

The compiles but throws an exception at runtime. `sb` variable is initialized with an empty `StringBuilder` object. Then the program arguments are iterated. For every given argument, for loop tries to insert the `String` element being iterated to the `sb` `StringBuilder` at the index of character `'c'` in `sb`. But since the the `sb` `StringBuilder` is empty, `sb.indexof("c")` return `-1`, and hence code throws `StringIndexOutOfBoundsException` at runtime.

Q2)(C, E)

Addition operator has higher precedence over assignment operator, so the option A is wrong. For option B, addition operator is listed last, but it has lower precedence over the previous three operators, so it is wrong too. Option D is wrong because the subtraction operator has lower precedence over the multiplication operator. Option C and option E has the correct order of operators according to increasing or the same level of precedence.

Q3)(C, B, F)

Option A is a getter method but it has a parameter in its method signature, so it is not a valid JavaBean. Option D should should return a boolean, looking at the name, at it is prefixed with 'is', so this option is not a valid bean too. Lastly Option E is wrong because there is no such naming convention. C, B, F are valid beans.

Q4)(A, E)

Array has `length` variable instead of a `size()` method. `size()` method is used for ArrayLists. Upon the correction of this one line, code would compile and run without issue. The nested for loop iterates correctly, and without trying to access out of bounds of the crossword array. It only iterates through the half of the every array in the second dimension though. Because the second dimension has `10` arrays in it with every array having length `20`, we are iterating every `10` array but stopping after the ninth element which corresponds to half of the whole elements. And unassigned half has their values set to default value `0`.

Q5)(B, D)

If a file system resource becomes temporarily unavailable, a checked exception must be used. Error is not a subclass of the class `Exception`, but `Throwable` class. If a user enters invalid input, i think an unchecked exception must be used, because we can consider it a programming error. Although it is possible, it is very unlikely that we would've want to catch it.

Q6)(D, A, C)

`import jungle.tree.*` statement imports all of the class inside tree package, allows us to access them without specifying the full package name. This is also true for the import statement `import savana.*`. `import forest.Bird` imports a class, `Bird` class, so again we don't need to use its full package name. `java.lang.Object` package does not need to be imported, since its inherently imported, we can use `Object` class without a full package name. `savana.sand.Wave` class cannot be accessed as asked in the question since the import statement only import classes in the `savana` package. Classes option E and F, should be used with full package name too since only Bird class imported from the forest package. Key point here is that we should not overlook the fact that by using '*', we are not recursively importing all packages.

Q7)(C)

Two of the given four variables represent immutable objects. `Strings` and `LocalDateTime` object are immutable objects, while `Arraylists` and `StringBuilder` classes represent mutable objects.

Q8)(C)

The output of the given code is `"wing"`. Given code iteratively removes characters from the `StringBuilder` unless the length of the `Character` sequence is less than `5`. First iteration leaves the object with the value `"s growing"`, since the length is greater than `5`, iteration runs again, leaving the value with `"wing"`. `while` condition check returns `false` since the length is now less than `5`. And the last value of the variable which is `"wing"` is printed after the `do-while` loop.

Q9)(D)

All of the created `String` objects are new objects, meaning that the all three variables points to different object. `new` keyword always instantiates a new object in the heap. Only the `ceiling` string is interned, but the next two variable are instantiated and initialized with `new` keyword. The `==` operator checks if the references point to the same object, so since none of the three point to the same object as other, there is no any possibility that any equality check on these three references returns `true`. equals method returns `true` with every combination of these three variables, one of them as caller other the argument. Because the `equals` method is implemented to check the elements in the `String` rather than comparing references.

Q10)(C)

The code prints true three times. Since `String` objects are mutable, all the methods which return a `String` after an operation which seems to mutate the string, actually returns a new `String` object. Because again, `Strings` are immutable. Last three statements prints **`true`**. Since they are equality checks on the contents of the strings rather than the references, and the `ignoreCase` is used where the contents are different.

Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()
Q)()

Q)()
Q)()
Q)()
Q)()
Q)()
Q)()