The following classes implements a StackOfCharacters.

```java
public class StackOfCharacters {

    private Character [] list;
    public static final int DEFAULT_INITIAL_CAPACITY = 16;
    private int size;
    private int capacity;

    public StackOfCharacters( int capacity ) {
        if (capacity <= 0 ) capacity = DEFAULT_INITIAL_CAPACITY;
        list = new Character [capacity];
        size = 0;
        this.capacity = capacity;
    }

    public StackOfCharacters( ) {
        this(DEFAULT_INITIAL_CAPACITY);
    }

    /**
     * Determines if this stack is empty or not.
     * @return true if this stack is empty, false otherwise
     */
    public boolean empty ( ) {
        // TODO: implement this method
    }

    public int getSize() {
        return size;
    }

    public int getCapacity() {
        return capacity;
    }

    /**
     * TODO: provide the Javadoc comments for this method
     */
    public void push ( Character value ) {
        // WARNING: this implementation has a bug
        if ( full() )
            makeLarger();
        if (value != null) {
            list[size] = value;
        }
    }

    /*
     */
```
* TODO: provide the Javadoc comments for this method

```java
public Character pop()
{
    if ( empty() ) return null;
    else {
        size--;
        return list[size];
    }
}

public Character peek()
{
    if ( empty() ) return null;
    else {
        return list[size-1];
    }
}

private boolean full()
{
    if (size == capacity )
        return true;
    else
        return false;
}

private void makeLarger ()
{
    Character [] newList = new Character [__________________ ];
    for (int i = 0; i < capacity; i++)
    {
        __________________________
    }
    list = newList;
    capacity = __________________________;
}

public String toString() {
    StringBuilder s = new StringBuilder();
    s.append("bottom [ ");
    for (int i = 0; i < size-1; i++)
        s.append(list[i]+", ");
    s.append(list[size-1]);
    s.append("] top");
    return s.toString();
}
```
Questions:

1. List all the data fields in the class `StackOfCharacters`.

2. Explain the difference between `size` and `capacity` of the stack.

3. Does the class have a default constructor? If so, which line is it on?

4. What are the values of ALL data fields right after a `StackOfCharacters` object is created (right after a call to a constructor)?

5. Implement the `empty` method on line 25.

6. The `peek` method on line 66, uses `empty` method in an `if` statement, and returns `null` if that method returns `true`. Why is this done?

7. The `push` method has a bug. Figure out what it is and suggest a fix.

8. Write the Javadoc comments for the `push` and the `pop` methods.

9. `makeLarger` should create a larger array when the capacity is reached. Complete the blanks in the code to achieve this.

10. Draw the content of the array representing a stack after the following sequence of operations:

```java
StackOfCharacters s = new StackOfCharacters();
s.push('C');
s.push('S');
s.push(' ');
s.push('2');
s.pop();
s.push('1');
s.push('0');
s.peek();
s.push('1');
```