1   State Space as Text

1.

```
public interface SearchListener extends JPFListener {
  void stateAdvanced(Search search);
  void stateProcessed(Search search);
  void stateBacktracked(Search search);
  void statePurged(Search search);
  void stateStored(Search search);
  void stateRestored(Search search);
  void propertyViolated(Search search);
  void searchStarted(Search search);
  void searchConstraintHit(Search search);
  void searchFinished(Search search);
}
```

Implement a listener which prints the states and transitions visited by the search in the following simple format:

```
-1 -> 0
0 -> 1
1 -> 2
0 -> 3
3 -> 4
4 -> 2
```

Which methods of the **SearchListener** interface are relevant?

2. In order to print a transition, what information do we need?

3. How do we store that information?

4.

```
public void stateAdvanced(Search search) {
  this.previous = ???;
  this.current = ???;
}
```

How do we update **this.previous**?

5. How can we use the **Search** parameter of the **stateAdvanced** method to update **this.current**?

6. Where do we initialize the attributes **current** and **previous**?

7. How do we initialize the attributes **current** and **previous**?

8. Complete the following.

```
public class StateSpace extends ListenerAdapter implements SearchListen
  // attributes


  // constructor



  // methods
  public void stateAdvanced(Search search) {





  }

  public void stateBacktracked(Search search) {


  }

  public void stateRestored(Search search) {
```

```
        }
    }
```

2   State Space as Dot File

1. Implement a listener which creates a dot file representing the the states and transitions visited
    by the search.

    ```
    digraph statespace {
    -1 -> 0
    0 -> 1
    1 -> 2
    0 -> 3
    3 -> 4
    4 -> 2
    }
    ```

    Where do we open a file for writing?

2. Where do we print **digraph statespace {**?

3. Where do we print the final **}**?

3   State Space as Dot File with Colours

1. Implement a listener which creates a dot file representing the the states and transitions visited
    by the search. Colour the initial state green and the final states red.

    ```
    digraph statespace {
    -1 [fillcolor=green]
    -1 -> 0
    0 -> 1
    1 -> 2
    2 [fillcolor=red]
    0 -> 3
    3 -> 4
    4 -> 2
    }
    ```

The initial state always has ID -1. Where do we print **-1 [fillcolor=green]**?

2. The class **Search** has a method **isEndState**. How can this method be used?