The `tableswitch` and `lookupswitch` instructions can be found at http://www.cat.nyu.edu/meyer/jvmref. The `tableswitch` is used when the cases of a switch can be efficiently represented as indices into a table of target offsets. Let \( \text{min} \) and \( \text{max} \) be the smallest index and largest index, respectively. Let \( \text{expr} \) be the value of the switch expression, which is available on the top of the operand stack. The default case, if any, will be taken if \( \text{expr} \) is not in the range \([\text{min}..\text{max}]\). Otherwise, the case marked by the \((\text{expr}-\text{min}+1)\)-st label (starting from 1) will be taken.

Where the cases of a switch are sparse, the table representation of the `tableswitch` becomes inefficient in terms of space. The `lookupswitch` is used instead. In this case, a linear or binary search is used to determine the case that should be taken.

1. Consider the following C/Java switch statement:

```java
switch (i) { // assume that the local variable index for i is 1
    case 2: return 2;
    case 6: return 6;
    case 7: return 7;
    default: return -1;
}
```

(a) Translate the switch into Jasmin code `tableswitch`.
(b) Translate the switch into Jasmin code `lookupswitch`.
(c) Translate the switch into Jasmin code as if it were a sequence of if-then-else statements without using `tableswitch` or `lookupswitch`. Is the generated code efficient?