Compiling Loops
Translating While Statement

Consider translation of the **while** statement, which gets 'nextLabel' destination, specifying where to jump when exiting the loop. We assume that the instructions emitted are inside the block that introduced nextLabel.

What is the translation schema?

```
[ while (cond) stmt ] nextLabel =
```
Translating While Statement

Consider translation of the while statement, which gets 'nextLabel' destination, specifying where to jump when exiting the loop. We assume that the instructions emitted are inside the block that introduced nextLabel.

What is the translation schema?

```
[ while (cond) stmt ] nextLabel =
  loop startLabel
  block bodyLabel
    branch(cond, bodyLabel, nextLabel)
  end // bodyLabel
[ stmt ] startLabel
end
```
In many languages, a break statement can be used to exit from the loop. For example, it is possible to write code such as this:

```cpp
while (cond1) {
    code1
    if (cond2) break;
    code2
}
```

Loop executes code1 and checks the condition cond2. If condition holds, it exists. Otherwise, it continues and executes code2 and then goes to the beginning of the loop, repeating the process.

Give translation scheme for this loop construct and explain how the translation of other constructs needs to change.
break Statement - Propagating Exit Label

For a **break** statement to know where to jump, it needs to be given a label indicating the exit of the loop. When we translate a statement (such as **if**) potentially containing **break**, the translation of this statement needs both the parameter to pass on to **break** as well as the parameter to jump to during normal execution. Therefore, each statement needs two destination parameters: the 'nextLabel' and the 'loopExit' label. For example,

```
[ if (cond) thenC else elseC ] nextL loopExitL =
```
For a **break** statement to know where to jump, it needs to be given a label indicating the exit of the loop. When we translate a statement (such as **if**) potentially containing **break**, the translation of this statement needs both the parameter to pass on to **break** as well as the parameter to jump to during normal execution. Therefore, each statement needs two destination parameters: the 'nextLabel' and the 'loopExit' label. For example,

```plaintext
[ if (cond) thenC else elseC ] nextL loopExitL =
  block elseL
  block thenL
    branch (cond, thenL, elseL)
  end // thenL
[thenC] nextL loopExitL
end // elseL
[elseC] nextL loopExitL
```
Translating `break`:

```
[ break ] nextLabel loopExitLabel =
```
Translating **break**:  

```markdown
[break] nextLabel loopExitLabel = br loopExitLabel
```
break Statement - Using and Setting Labels

Translating `break`:

```
[ break ] nextLabel loopExitLabel =
  br loopExitLabel
```

Translating `while`:

```
[ while (cond) stmt ] nextLabel loopExitLabel =
```
break Statement - Using and Setting Labels

Translating `break`:

```plaintext
[ break ] nextLabel loopExitLabel = br loopExitLabel
```

Translating `while`:

```plaintext
[ while (cond) stmt ] nextLabel loopExitLabel =
  loop startLabel
  block bodyLabel
    branch (cond, bodyLabel, nextLabel)
  end // bodyLabel
[ stmt ]
```

What if we want to have `continue` that goes to beginning of the loop?
break Statement - Using and Setting Labels

Translating **break**:

```plaintext
[ break ] nextLabel loopExitLabel = br loopExitLabel
```

Translating **while**:

```plaintext
[ while (cond) stmt ] nextLabel loopExitLabel = loop startLabel
  block bodyLabel
    branch (cond, bodyLabel, nextLabel)
  end // bodyLabel
[ stmt ] startLabel
```
break Statement - Using and Setting Labels

Translating `break`:

```
[ break ] nameLabel loopExitLabel =
  br loopExitLabel
```

Translating `while`:

```
[ while (cond) stmt ] nameLabel loopExitLabel =
  loop startLabel
    block bodyLabel
      branch(cond, bodyLabel, nameLabel)
    end // bodyLabel
  [ stmt ] startLabel nameLabel
end
```
Translating \texttt{break}:

\begin{verbatim}
[ break ] nextLabel loopExitLabel = br loopExitLabel
\end{verbatim}

Translating \texttt{while}:

\begin{verbatim}
[ while (cond) stmt ] nextLabel loopExitLabel =
  loop startLabel
  block bodyLabel
  branch(cond, bodyLabel, nextLabel)
  end // bodyLabel
  [ stmt ] startLabel nextLabel
end
\end{verbatim}

What if we want to have \texttt{continue} that goes to beginning of the loop?
Loops with break and continue

Translating `break`:

```
[ break ] nextL loopExitL loopStartL = br loopExitL
```

Translating `continue`:

```
[ continue ] nextL loopExitL loopStartL = br loopStartL
```

Translating `while`:

```
[ while (cond) stmt ] nextL loopExitL loopStartL =
  loop startLabel
  block bodyLabel
  branch(cond, bodyLabel, nextL)
  end // bodyLabel
[ stmt ] startLabel nextL startLabel
end
```

Explain difference between labels `loopStartL` and `startLabel`