Show that the following problem is undecidable, by giving a reduction from the Hello World problem:

given a program P and an input x, does P eventually halt when it is given x as input?

(Note: this problem is called the Halting problem.)

Solution:

We have to construct a reduction Red from the HWP to the HP.

Red is a program that:
- takes as input an instance \((Q, y)\) of the HWP, and
- produces as output an instance \((P, x)\) of the HP such that

\[
Q(y) = \text{"Hello, World"} \iff \text{HP}(P, x) = \text{"yes"}.
\]

Using Red, we can show that the HP is undecidable. Indeed, assume the HP is decidable, and let \(S_{\text{HP}}\) be a solver for the HP, i.e.

\[
(P, x) \rightarrow S_{\text{HP}} \rightarrow \text{"yes"} \text{ if } P(x) \text{ halts} \rightarrow \text{"no"} \text{ if } P(x) \text{ does not halt}
\]

We now Red and \(S_{\text{HP}}\) to construct a solver \(S_{\text{HWP}}\) for the HWP:

\[
(Q, y) \rightarrow \text{Red} \rightarrow \text{Red}(P, x) \rightarrow S_{\text{HP}} \rightarrow \text{"yes"} \text{ if } Q(y) = \text{"H,W"} \rightarrow \text{"no"} \text{ if } Q(y) \neq \text{"H,W"}
\]

Since \(S_{\text{HWP}}\) does not exist, also \(S_{\text{HP}}\) cannot exist.
We show now how to construct Red by describing what it does:

\[(a, y) \rightarrow \text{Red} \rightarrow (P, x)\]

Red leaves y unchanged, i.e. \(x = y\).
Red performs on Q the following modifications:

1) It makes sure that Q never halts
   (e.g. by inserting \(\text{while} (\text{true}) \{ \}\) at the end of
    \(\text{main()}\) and before every return; in \(\text{main}()\))

2) It modifies the \(\text{print}()\) method so that it stores the
   printed characters in an array and then checks whether
   the array contains "Hello, World". If yes, Q halts.

The resulting program is P.

Note that Red, which computes P from Q can be written
in Java.
Exercise (Example 8.2 from textbook)

Construct a Turing Machine accepting the language

\[ \{0^n1^m | n \geq 1\} \]

Solution

The idea is that the TM \( M \) that we construct needs to read the leftmost 0, turn it into \( X \), and move right until it reaches a 1, that is turned into \( Y \). Then the head moves left again to the leftmost 0 (on the right to a \( X \)), and starts again until all 0's and 1's are turned into \( X \)'s and \( Y \)'s respectively.

If the input is not in \( 0^*1^* \), \( M \) will fail to find a move and it won't accept. If \( M \) changes the last 0 and the last 1 in the same round, it will go into the final state and accept.

\[ Q = \{ q_0, q_1, q_2, q_3, q_4 \} \]
\[ \Sigma = \{ 0, 1 \} \]
\[ \Gamma = \{ 0, 1, X, Y, S \} \quad (S \text{ denotes blank symbol}) \]
\[ q_0 : \text{start state} \]
\[ f = \{ q_4 \} \]

In \( q_0 \) is the state in which \( M \) is when the head precedes the leftmost 0. In state \( q_1 \), \( M \) moves right skipping 0's and 1's until it gets to a 1. In state \( q_2 \), \( M \) moves left while skipping \( Y \)'s and 0's again, until it gets to a \( X \) and goes again in \( q_0 \).
Starting from $q_0$, if a $Y$ is read instead of a $0$, $M$ goes in $q_3$ and moves right; if a $1$ is found, then there are more $1$'s than $0$'s; if a $0$ is read, then the initial string is accepted (transition to $q_4$).

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>X</th>
<th>Y</th>
<th>Ø</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>$(q_1, x, R)$</td>
<td>—</td>
<td>—</td>
<td>$(q_3, Y, R)$</td>
<td>—</td>
</tr>
<tr>
<td>$q_1$</td>
<td>$(q_1, 0, R)$</td>
<td>$(q_2, Y, L)$</td>
<td>—</td>
<td>$(q_4, Y, R)$</td>
<td>—</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$(q_2, 0, L)$</td>
<td>—</td>
<td>$(q_0, X, R)$</td>
<td>$(q_2, Y, L)$</td>
<td>—</td>
</tr>
<tr>
<td>$q_3$</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>$(q_3, Y, R)$</td>
<td>$(q_4, Ø, R)$</td>
</tr>
<tr>
<td>$q_4$</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

**Exercise**

Show the computation of the TM above when the input string is:

(a) 00
(b) 000111

**Solution**

(a) $q_0$ 00 $\rightarrow$ $q_1$ 0 $\rightarrow$ $q_2$ 0 $\rightarrow$ $q_3$ Y $\rightarrow$ $q_4$

and the TM halts

(b) $q_0$ 000111 $\rightarrow$ $q_1$ 00111 $\rightarrow$ $q_2$ 0111 $\rightarrow$ $q_3$ 111 $\rightarrow$ $q_4$

$\cdots$
Exercise (8.1.1 from textbook)

Give a reduction from the hello-world problem to the following problem:

given a program \( P \) and an input \( I \), does the program ever produce any output?

Solution

We modify \( P \) by making it print its output on some array \( A \), capable of storing 12 characters. When \( A \) is full, \( P \) checks whether it stores “hello world”: if it does, \( P \) prints (on the output, not on the array) some character (like @); if not, it does not print anything.

So the modified program prints some output if and only if \( P \) prints “hello, world”: if we are able to determine whether a program produces any output, we can solve the hello-world problem. This ends our reduction.
Exercise (8.2.3 from textbook):

Design a Turing Machine that takes as input a number \( N \) in binary and turns it into \( N+1 \) (in binary); the number \( N \) is preceded by the symbol \( \$ \), which may be destroyed during the computation. For example, \( \$111 \) is turned into \( 1000 \); \( \$1001 \) is turned into \( \$1010 \).

Solution

The idea is to toggle the rightmost digit, and , from right to left, all consecutive 1's until we get to the first 0 (which is also toggled). If there is no 0 to be toggled, a 1 is added on the left of the first digit (i.e., in place of the \( \$ \)).

We need three states, where only \( q_2 \) is the final state; we briefly describe what the TM does in the different states.

- \( q_0 \): the TM goes right until it reaches \( \$ \), after the rightmost digit. When \( \$ \) is reached, the TM goes into \( q_2 \).
- \( q_1 \): goes left toggling all 1's and the first 0 (from right); when 0 or \( \$ \) is reached, the symbol is turned into 1.
- \( q_2 \): final state; the TM does nothing.

<table>
<thead>
<tr>
<th></th>
<th>$</th>
<th>0</th>
<th>1</th>
<th>( b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_0 )</td>
<td>( q_0, #, R )</td>
<td>( q_0, 0, R )</td>
<td>( q_0, 1, R )</td>
<td>( q_1, b, L )</td>
</tr>
<tr>
<td>( q_1 )</td>
<td>( q_1, 1, L )</td>
<td>( q_2, 1, L )</td>
<td>( q_2, 0, L )</td>
<td>_</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
</tr>
</tbody>
</table>
Exercise  (8.22 from textbook)

Design Turing machines accepting the following language:

\[ \{ w \in \{0,1\}^* \mid w \text{ has an equal number of } 0\text{'s and } 1\text{'s} \} \]

**solution**

The idea is that the head of our TM \( M \) moves back and forth on the tape, "deleting" one 0 for each 1, if there are no 0\'s and 1\'s in the end, the string is accepted.

When in state \( q_1 \), \( M \) has found a 1 and looks for a 0; in state \( q_2 \) it is the other way around.

Note that the head never moves left of any \( X \), so that there are never unmatched 0\'s and 1\'s on the left of an \( X \).

From initial state \( q_0 \), \( M \) picks up a 0 or a 1 and turns it into \( X \). The only final state is \( q_4 \). In state \( q_3 \), \( M \) moves back left looking for the rightmost \( X \).

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>( \varepsilon )</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_0 )</td>
<td>( q_2, X, R )</td>
<td>( q_1, X, R )</td>
<td>( q_2, \varepsilon, R )</td>
<td>( q_0, Y, R )</td>
<td></td>
</tr>
<tr>
<td>( q_1 )</td>
<td>( q_3, Y, L )</td>
<td>( q_2, 1, R )</td>
<td></td>
<td>( q_2, Y, R )</td>
<td></td>
</tr>
<tr>
<td>( q_2 )</td>
<td>( q_2, 0, R )</td>
<td>( q_2, Y, L )</td>
<td></td>
<td>( q_2, Y, R )</td>
<td></td>
</tr>
<tr>
<td>( q_3 )</td>
<td>( q_3, 0, L )</td>
<td>( q_3, 1, L )</td>
<td>( q_0, X, R )</td>
<td>( q_3, Y, L )</td>
<td></td>
</tr>
<tr>
<td>( q_4 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>