Question 1: Write your name and student number.

Solution: Christine Sinclair, 12

Question 2: Construct a Turing machine with one tape that gets as input an integer \( x \geq 0 \) and returns as output the integer \( x + 1 \). Integers are represented in binary.

Start of the computation: The tape contains the binary representation of the input \( x \). The tape head is on the leftmost bit of \( x \) and the Turing machine is in the start state.

End of the computation: The tape contains the binary representation of the number \( x + 1 \). The tape head is on the leftmost bit of \( x + 1 \) and the Turing machine is in the final state.

The Turing machine in this question does not have an accept state or a reject state; instead, it has a final state. As soon as this final state is entered, the Turing machine terminates. At termination, the contents of the tape is the output of the Turing machine.

Start by explaining your algorithm in plain English, then mention the states that you are going to use, then explain the meaning of these states, and finally give the list of instructions.

Solution: The Turing machine does the following:

- The tape head moves to the rightmost bit of \( x \), without making any changes on the tape.

- The tape head moves to the left until it reaches a 0 or the blank symbol. While moving to the left, all 1’s are replaced by 0’s, and the first 0 or □ is replaced by 1.

- Finally, the tape head moves further left until it reaches the leftmost bit.

We will use the following states:

- \( q_0 \): start state; tape head moves to the right
- \( q_1 \): final state
- \( q_2 \): tape head moves to the left; on its way to the left it has not read a 0 or a □
- \( q_3 \): \( x + 1 \) has been computed; tape head moves to the left

Here are the instructions:

\[
\begin{align*}
q_00 & \rightarrow q_00R \\
q_01 & \rightarrow q_01R \\
q_0□ & \rightarrow q_2□L \\
q_20 & \rightarrow q_31L \\
q_21 & \rightarrow q_20L \\
q_2□ & \rightarrow q_11N \\
q_30 & \rightarrow q_30L \\
q_31 & \rightarrow q_31L \\
q_3□ & \rightarrow q_1□R
\end{align*}
\]

In Figure 1, you find the states and the contents of the tape during the computation of this Turing machine for the input \( x = 431 \).
Figure 1: The computation of the Turing machine in Question 1 for the input $x = 431$. The boldface symbol indicates the position of the tape head.
Question 3: Construct a Turing machine with three tapes that gets as input two integers $x \geq 0$ and $y \geq 0$, and returns as output the number $x + y$. Integers are represented in binary.

Start of the computation: Tape 1 contains the binary representation of $x$, its head is on the rightmost bit of $x$. Tape 2 contains the binary representation of $y$, its head is on the rightmost bit of $y$. Tape 3 is empty (that is, it contains only □’s), its head is at an arbitrary position. At the start, the Turing machine is in the start state.

End of the computation: Tapes 1 and 2 are empty, and tape 3 contains the binary representation of the number $x + y$. The head of tape 3 is on the rightmost bit of $x + y$. The Turing machine is in the final state.

The Turing machine in this question does not have an accept state or a reject state; instead, it has a final state. As soon as this final state is entered, the Turing machine terminates. At termination, the contents of tape 3 is the output of the Turing machine.

Start by explaining your algorithm in plain English, then mention the states that you are going to use, then explain the meaning of these states, and finally give the list of instructions.

Solution: The Turing machine will do the following.

Stage 1: Walk (simultaneously on all tapes) from right to left, and perform the addition. While doing this, write $x + y$ on tape 3, and delete all bits from tapes 1 and 2. Use two states $q_0$ and $q_1$ to remember whether or not there is a carry.

Stage 2: In this stage, the sum $x + y$ has been computed, and tapes 1 and 2 are already empty. In this stage, head 3 moves to the rightmost bit on its tape.

We use the following states:

- $q_0$: start state; we are in stage 1; there is no carry.
- $q_1$: we are in stage 1; there is a carry.
- $q_2$: we are in stage 2.
- $q_3$: final state.
Here are the instructions:

$q_0$00□ → $q_0$□□0LLL
$q_0$01□ → $q_0$□□1LLL
$q_0$10□ → $q_0$□□1LLL
$q_0$11□ → $q_1$□□0LLL
$q_0$□0□ → $q_0$□□0LLL
$q_0$□1□ → $q_0$□□1LLL
$q_0$0□□ → $q_0$□□0LLL
$q_0$1□□ → $q_0$□□1LLL
$q_0$□□□ → $q_2$□□□RRR
$q_1$00□ → $q_0$□□1LLL
$q_1$01□ → $q_1$□□0LLL
$q_1$10□ → $q_1$□□0LLL
$q_1$11□ → $q_1$□□1LLL
$q_1$□0□ → $q_0$□□1LLL
$q_1$□1□ → $q_1$□□0LLL
$q_1$0□□ → $q_0$□□1LLL
$q_1$1□□ → $q_1$□□0LLL
$q_1$□□□ → $q_2$□□□1NNN
$q_2$□□0 → $q_2$□□0RRR
$q_2$□□1 → $q_2$□□1RRR
$q_2$□□□ → $q_3$□□□LLL

**Question 4:** In class, we have seen that the language 

\[ Halt = \{ \langle P, w \rangle : \text{P is a Java program that terminates on the binary input string } w \} \]

is undecidable.

A Java program $P$ is called a *Hello-World-program*, if the following is true: When given the empty string $\epsilon$ as input, $P$ can do whatever it wants, as long as it outputs the string *Hello World* and terminates. (We do not care what $P$ does when the input string is non-empty.) Consider the language

\[ HW = \{ \langle P \rangle : \text{P is a Hello-World-program} \} \]

The questions below will lead you through a proof of the claim that the language $HW$ is undecidable.

(4.1) Consider a fixed Java program $P$ and a fixed binary string $w$.

We write a new Java program $J_{Pw}$, which takes as input an arbitrary binary string $x$. On such an input $x$, the Java program $J_{Pw}$ does the following:

**Algorithm** $J_{Pw}(x)$:
run $P$ on the input $w$;
print *Hello World*
• Argue that $P$ terminates on input $w$ if and only if $\langle J_{Pw} \rangle \in HW$.

Solution:

• Assume that $P$ terminates on input $w$.

Let $x$ be an arbitrary input string for $J_{Pw}$. We go through the pseudocode for $J_{Pw}$ and see what happens: First, we run $P$ on the input $w$. Because of our assumption, this part of the pseudocode terminates. Then, in the next line, Hello World is printed and $J_{Pw}$ terminates.

Thus, for any $x$, the computation of $J_{Pw}(x)$ prints Hello World and terminates. In particular, this is true for $x = \epsilon$. It follows that $J_{Pw}$ is a Hello-World-program and, thus, $\langle J_{Pw} \rangle \in HW$.

• Assume that $P$ does not terminate on input $w$.

Let $x$ be an arbitrary input string for $J_{Pw}$. We go through the pseudocode for $J_{Pw}$ and see what happens: First, we run $P$ on the input $w$. Because of our assumption, this part of the pseudocode does not terminate. As a result, $J_{Pw}$ does not terminate.

Thus, for any $x$, the computation of $J_{Pw}(x)$ does not terminate. In particular, this is true for $x = \epsilon$. It follows that $J_{Pw}$ is not a Hello-World-program and, thus, $\langle J_{Pw} \rangle \notin HW$.

(4.2) The goal is to prove that the language $HW$ is undecidable. We will prove this by contradiction. Thus, we assume that $H$ is a Java program that decides $HW$. Recall what this means:

• If $P$ is a Hello-World-program, then $H$, when given $\langle P \rangle$ as input, will terminate in the accept state.

• If $P$ is not a Hello-World-program, then $H$, when given $\langle P \rangle$ as input, will terminate in the reject state.

We write a new Java program $H'$ which takes as input the binary encoding $\langle P, w \rangle$ of an arbitrary Java program $P$ and an arbitrary binary string $w$. On such an input $\langle P, w \rangle$, the Java program $H'$ does the following:

\textbf{Algorithm} $H'(\langle P, w \rangle)$:

construct the Java program $J_{Pw}$ described above;
run $H$ on the input $\langle J_{Pw} \rangle$;
if $H$ terminates in the accept state
then terminate in the accept state
else terminate in the reject state
endif

5
Argue that the following are true:

- For any input $\langle P, w \rangle$, $H'$ terminates.

  **Solution:** This follows from the fact that $H$ terminates on any input.

- If $P$ terminates on input $w$, then $H'$ (when given $\langle P, w \rangle$ as input), terminates in the accept state.

  **Solution:** We assume that $P$ terminates on input $w$. We know from the first part of the question that $\langle J_{Pw} \rangle \in HW$. Since $H$ decides the language $HW$, it follows that, on input $\langle J_{Pw} \rangle$, $H$ terminates in the accept state. It then follows from the pseudocode for $H'$ that this program, on input $\langle P, w \rangle$, terminates in the accept state.

- If $P$ does not terminate on input $w$, then $H'$ (when given $\langle P, w \rangle$ as input), terminates in the reject state.

  **Solution:** We assume that $P$ does not terminate on input $w$. We know from the first part of the question that $\langle J_{Pw} \rangle \notin HW$. Since $H$ decides the language $HW$, it follows that, on input $\langle J_{Pw} \rangle$, $H$ terminates in the reject state. It then follows from the pseudocode for $H'$, together with the fact that $H$ terminates on any input, that $H'$, on input $\langle P, w \rangle$, terminates in the reject state.

(4.3) Now finish the proof by arguing that the language $HW$ is undecidable.

**Solution:** Above, we have assumed that $HW$ is decidable. Based on this assumption, we have constructed a Java program $H'$ that has the following property:

- $H'$ terminates on any input string $\langle P, w \rangle$.

- $H'$ accepts the input string $\langle P, w \rangle$ if and only if $P$ terminates on the input string $w$.

- This means: $H'$ accepts $\langle P, w \rangle$ if and only if $\langle P, w \rangle \in Halt$.

- But, by definition, this means that the language $Halt$ is decidable.

- However, $Halt$ is undecidable. Therefore, $HW$ is undecidable.

**Question 5:** Consider the two languages

\[ Empty = \{ \langle M \rangle : M \text{ is a Turing machine for which } L(M) = \emptyset \} \]

and

\[ UselessState = \{ \langle M, q \rangle : M \text{ is a Turing machine, } q \text{ is a state of } M, \text{ for every input string } w, \text{ the computation of } M \text{ on input } w \text{ never visits state } q \}. \]

(5.1) Use Rice’s Theorem to show that $Empty$ is undecidable.

**Solutions:** We verify the three conditions in Rice’s theorem:
Let $M$ be the Turing machine that does the following: In the start state, and no matter which symbol is read, $M$ switches to the reject state. This Turing machine rejects every input string and, therefore, $L(M) = \emptyset$. This implies that $\langle M \rangle \in \text{Empty}$. Thus, there exists a Turing machine $M$ such that $\langle M \rangle \in \text{Empty}$.

In class, we have seen several Turing machines $N$ for which $L(N) \neq \emptyset$. Thus, there exists a Turing machine $N$ such that $\langle N \rangle \notin \text{Empty}$.

It is obvious that for any two Turing machines $M_1$ and $M_2$ with $L(M_1) = L(M_2)$, either both $\langle M_1 \rangle$ and $\langle M_2 \rangle$ are in $\text{Empty}$ or none of them is in $\text{Empty}$. (In other words, whether or not $\langle M \rangle$ is in $\text{Empty}$ only depends on the language of $M$.)

Since all three conditions in Rice’s theorem are satisfied, it follows that $\text{Empty}$ is undecidable.

\textbf{(5.2) Use (5.1) to show that $\text{UselessState}$ is undecidable.}

\textbf{Solution:} Let $M$ be a Turing machine and let $q$ be a state of $M$. We say that $q$ is a useless state if for every input string $w$, the computation of $M$ on input $w$ never visits state $q$.

Here is the main observation: Let $q_{\text{accept}}$ be the accept state of the Turing machine $M$. Then

$$L(M) = \emptyset$$

if and only if $q_{\text{accept}}$ is a useless state.

We assume that the language $\text{UselessState}$ is decidable. Let $H$ be a Turing machine that decides $\text{UselessState}$. Consider the following algorithm $H'$, which takes as input the binary encoding $\langle M \rangle$ of a Turing machine $M$:

\begin{algorithm}
\textbf{Algorithm $H'((M))$}:
\begin{itemize}
  \item let $q_{\text{accept}}$ be the accept state of $M$;
  \item run $H$ on the input $\langle M, q_{\text{accept}} \rangle$;
  \item if $H$ terminates in the accept state
  \begin{itemize}
    \item then accept and terminate
  \end{itemize}
  \item else reject and terminate
\end{itemize}
\end{algorithm}

This new algorithm $H'$ decides the language $\text{Empty}$. This is a contradiction, because we saw in (5.1) that $\text{Empty}$ is undecidable.