Question 1: Write your name and student number.

Solution: Aitana Bonmatí Conca, 14

Question 2: Construct a Turing machine with one tape, input alphabet \( \Sigma = \{a, b, c\} \), and tape alphabet \( \Gamma = \{a, b, c, \square\} \) that accepts the language

\[
\{a^m b^n c^{m+n} : m \geq 0, n \geq 0\}.
\]

You may assume that the input string belongs to the language described by the regular expression \( a^* b^* c^* \). At the start of the computation, the tape head is at the leftmost symbol of the input string. If the input string is empty, then the tape head is at an arbitrary position.

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 repeat the following:

- Starting at the leftmost symbol, if it is \( a \) or \( b \), delete it.
- Walk to the rightmost symbol. If it is \( c \), delete it.
- Walk to the leftmost symbol.

The input alphabet is \( \Sigma = \{a, b, c\} \) and the tape alphabet is \( \Gamma = \{a, b, c, \square\} \). We will use the following states:

- \( q_0 \): start state, we are at the leftmost symbol.
- \( q_r \): the leftmost symbol was \( a \) or \( b \), it has been deleted, we are walking to the rightmost symbol and hope that it is \( c \).
- \( q_R \): we are at the rightmost symbol and are still hoping that it is \( c \).
- \( q_\ell \): the rightmost symbol was \( c \), it has been deleted, we are walking to the leftmost symbol.
- \( q_{accept} \) and \( q_{reject} \).

Here are the instructions:
Question 3: Construct a Turing machine with one tape and input alphabet \( \Sigma = \{a\} \), that “doubles” a non-empty string of a’s.

**Start of the computation:** The tape contains a string of the form \( a^n \), for some \( n \geq 1 \), and the tape head is at the leftmost \( a \). The Turing machine is in the start state.

**End of the computation:** The tape contains the string \( a^{2n} \) and its head is at the rightmost \( a \). At the end, 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.

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, where \( b \) is a new symbol:

- Starting at the leftmost \( a \), walk to the right and replace every \( a \) by \( b \). At the first \( \square \), move one cell left and stop at the rightmost \( b \).

- Repeat the following:
  - Starting at the rightmost symbol, walk to the left until you see \( b \). Replace this \( b \) by \( a \).
  - Walk to the right and replace the first \( \square \) by \( a \).
  - Walk to the left until you see \( b \).
• After the string has been doubled, walk to the rightmost $a$.

The input alphabet is $\Sigma = \{a\}$ and the tape alphabet is $\Gamma = \{a, b, \square\}$. We will use the following states:

• $q_0$: start state, walk to the right and replace every $a$ by $b$, stop at the rightmost symbol (which must be $b$, because $n \geq 1$).

• $q_\ell$: walk to the left until you see $b$ or $\square$. If you see $b$ first, replace it by $a$ and switch to $q_r$. If you see $\square$ first, the input string has been doubled and we switch to $q_R$.

• $q_r$: walk to the right until you see $\square$; replace it by $a$ and switch to $q_\ell$. (Note: we cannot read $b$.)

• $q_R$: the string has been doubled; walk to the rightmost symbol. (Note: we cannot read $a$.)

• $q_f$: final state.

Here are the instructions:

$q_0a \to q_0bR$
$q_0b \to $ cannot happen
$q_0\square \to q_0\square L$
$q_\ell a \to q_\ell aL$
$q_\ell b \to q_\ell aR$
$q_\ell \square \to q_\ell \square R$
$q_r a \to q_r aR$
$q_r b \to $ cannot happen
$q_r \square \to q_r aL$
$q_R a \to q_R aR$
$q_R b \to $ cannot happen
$q_R \square \to q_f$

Below you see a simplified version that terminates at the leftmost $a$. The Turing machine has been implemented using the iPhone app Turing.
Question 4: We have seen in class that the language

$$Halt = \{ \langle M, w \rangle : M \text{ is a Turing machine that terminates on the input string } w \}$$

is undecidable. In the language $Print_B$ that is defined below, $\Sigma$ denotes the input alphabet of the Turing machine $M$, and $\Gamma$ denotes its tape alphabet.
\( \text{PrintB} = \{ \langle M, w, b \rangle : M \text{ is a Turing machine, } w \in \Sigma^*, b \in \Gamma, \text{ when running } M \text{ on input } w, M \text{ writes } b \text{ on the tape at least once} \}. \)

Prove that \( \text{PrintB} \) is undecidable.

\textit{Hint:} Given an input \( \langle M, w \rangle \) for \( \text{Halt} \), modify \( M \) such that the resulting Turing machine prints a new symbol, say \#s, at the moment it terminates.

\textbf{Solution:} As always, the proof is by contradiction. We assume that \( \text{PrintB} \) is decidable. Let \( H \) be a Turing machine that decides \( \text{PrintB} \): For any string \( \langle M, w, b \rangle \),

- if the Turing machine \( M \), on input \( w \), writes \( b \) at least once, then \( H \) terminates and accepts \( \langle M, w, b \rangle \).
- if the Turing machine \( M \), on input \( w \), never writes \( b \), then \( H \) terminates and rejects \( \langle M, w, b \rangle \).
- Note: It may happen that \( M \) does not terminate on input \( w \). In this case, \( H \) is still able to find out whether or not \( M \) writes \( b \) at least once. In particular, \( H \) terminates.

We are going to show that \( \text{Halt} \) is decidable. Consider the following algorithm \( H' \), which takes as input \( \langle M, w \rangle \):

\textbf{Step 1:} Modify the Turing machine \( M \) as follows, and denote the resulting Turing machine by \( M' \):

- Add a new symbol \# to the tape alphabet.
- Replace each instruction

\[ ra \rightarrow q_{\text{accept}} *_1 *_2, \]

where \( *_1 \) is in the old tape alphabet and \( *_2 \) is \( R, L, \) or \( N \), by

\[ ra \rightarrow q_{\text{accept}} \# *_2. \]

- Replace each instruction

\[ ra \rightarrow q_{\text{reject}} *_1 *_2, \]

where \( *_1 \) is in the old tape alphabet and \( *_2 \) is \( R, L, \) or \( N \), by

\[ ra \rightarrow q_{\text{reject}} \# *_2. \]

- For each \( r \notin \{ q_{\text{accept}}, q_{\text{reject}} \} \), add instructions

\[ r\# \rightarrow r\#N. \]

Note: \( M \) enters the accept or reject state (and, thus, terminates), if and only if \( M' \) prints \#.

\textbf{Step 2:} Run algorithm \( H \) on input \( \langle M', w, \# \rangle \).
• If $H$ terminates in the accept state, then $H'$ terminates in the accept state.
• If $H$ terminates in the reject state, then $H'$ terminates in the reject state.

Since $H$ terminates on every input, $H'$ also terminates on every input. Since the following are equivalent

- $\langle M, w \rangle \in \text{Halt}$,
- $M$ terminates on input $w$,
- on input $w$, $M'$ prints $#$ at least once,
- $\langle M', w, \# \rangle \in \text{PrintB}$,
- $H$ accepts $\langle M', w, \# \rangle$,
- $H'$ accepts $\langle M, w \rangle$,

algorithm $H'$ decides $\text{Halt}$. Thus, $\text{Halt}$ is decidable, which is a contradiction. We conclude that $\text{PrintB}$ is undecidable.

**Question 5:** Let $A$ be an arbitrary language that is enumerable, but not decidable. Recall what it means to enumerable: There exists a Turing machine $M$, such that for any input string $w$:

- If $w \in A$, then, on input $w$, $M$ terminates in the accept state.
- If $w \notin A$, then, on input $w$, $M$ either terminates in the reject state or does not terminate.

Consider the following function $f : \{0,1\}^* \rightarrow \mathbb{N}$:

$$f(w) = \begin{cases} 
\text{the number of steps made by } M \text{ on input } w, & \text{if } M \text{ terminates on } w, \\
0, & \text{if } M \text{ does not terminate on } w. 
\end{cases}$$

In this question, you will prove that the function $f$ is not computable, i.e., there does not exist an algorithm that, for any input string $w \in \{0,1\}^*$, terminates and returns the value of $f(w)$.

(5.1) Let $g : \{0,1\}^* \rightarrow \mathbb{N}$ be an arbitrary computable function. Prove that there exists a string $w \in \{0,1\}^*$ such that $f(w) > g(w)$.

*Hint:* As you can expect, the proof is by contradiction. Thus, you assume that the claim is not true. Define a new Turing machine $N$ that, for any input string $w$ in $\{0,1\}^*$, runs the Turing machine $M$ for $g(w)$ steps and then “does something”.

(5.2) Prove that the function $f$ is not computable.

**Solution:** We start with the first part. We take an arbitrary computable function $g : \{0,1\}^* \rightarrow \mathbb{N}$ and assume that for every string $w$ in $\{0,1\}^*$, $f(w) \leq g(w)$. 
Consider the following algorithm $N$, which takes as input an arbitrary string $w$ in $\{0, 1\}^*$:

**Step 1:** Compute $g(w)$ and store the value in the variable $k$. (Note: Since $g$ is computable, there exists an algorithm that computes $g(w)$.)

**Step 2:** Run the Turing machine $M$ on input $w$ and stop as soon as $M$ terminates or $M$ has made $k$ steps.

**Step 3:**

- If $M$ terminates in the accept state within $k$ steps, then $N$ terminates and accepts the string $w$.
- Otherwise, $M$ did not terminate in the accept state after $k$ steps. In this case, $N$ terminates and rejects the string $w$.

Let us see what is going on here:

- Assume that $w \in A$. We know that, on input $w$, $M$ terminates in the accept state. By the definition of $f$, the number of steps made by $M$ is equal to $f(w)$. By our assumption, $f(w) \leq g(w) = k$. Thus, $M$ terminates in the accept state within $k$ steps. From Step 3, $N$ terminates and accepts the string $w$.

- Now assume that $w \not\in A$. Then $M$ does not accept $w$. In particular, $M$ does not accept $w$ within $k$ steps. From Step 3, $N$ terminates and rejects the string $w$.

The two items above imply that algorithm $N$ decides the language $A$. This is a contradiction, because $A$ is undecidable. We conclude that there exists a string $w$ in $\{0, 1\}^*$ such that $f(w) > g(w)$.

Now the second part. We assume that the function $f$ is computable. In the first part, we take $g = f$. Then we know that there exists a string $w$ in $\{0, 1\}^*$ such that $f(w) > f(w)$. This is a contradiction. Thus, $f$ is not computable.