Question 1: Write your name and student number.

Solution: Johan Cruijff, 14

Question 2: Consider the language $A$ consisting of all strings over the alphabet \{a, b\} that do not contain $bb$ as a substring. Give a regular expression that describes the language $A$. As always, justify your answer.

First Solution: Each string in the language $A$ is of the following form:

- It starts with zero or more $a$’s.
  - This is described by the regular expression $a^*$.
- Then we see zero or more times a string described by $baa^*$. This guarantees that every $b$ that is not the rightmost symbol is followed by at least one $a$.
  - This is described by the regular expression $(baa^*)^*$.
- The string may end with $b$.
  - This is described by the regular expression $\varepsilon \cup b$.

By concatenating these, we get the regular expression

$$a^* (baa^*)^* (\varepsilon \cup b).$$

Second Solution: You can check that the following regular expression is also correct:

$$(a \cup ba)^* (\varepsilon \cup b).$$

Using Prof. Swift’s result in Question 3, we can write this as

$$(a^* (ba)^*)^* (\varepsilon \cup b).$$

Third Solution: Here is a DFA whose language is the set of all strings in \{a, b\} that contain $bb$:
If we “flip” the states, then we get a DFA whose language is the set of all strings in \( \{a, b\}^* \) that do not contain \( bb \). Now we can use the technique seen in class to convert this to a regular expression.

**Question 3:** Let \( R_1 \) and \( R_2 \) be two arbitrary regular expressions over the same alphabet. Professor Taylor Swift claims that the regular expressions 
\[
(R_1 \cup R_2)^*
\]
and 
\[
(R_1^* R_2^*)^*
\]
describe the same language. If Professor Swift’s claim correct? As always, justify your answer.

**Solution:** As all Swifties know, Taylor is always right. Giving a formal proof is vey painful: You would have to use induction on \( R_1 \) and then, inside, induction on \( R_2 \). Those of you who come to class know that it is enough to give a proof in English.

**Left is contained in right:** We show that the language described by \((R_1 \cup R_2)^*\) is contained in the language described by \((R_1^* R_2^*)^*\).

Take an arbitrary string in the language described by \((R_1 \cup R_2)^*\). There is an integer \( k \geq 0 \), such that this string is described by “\( k \) times, do \( R_1 \) or \( R_2 \)”. Think of this as “do \( A_1, A_2, \ldots, A_k \)”, where each \( A_i \) is either \( R_1 \) or \( R_2 \). We show that this is contained in \((R_1^* R_2^*)^*\):

- For \( i = 1, 2, \ldots, k \):
  - if \( A_i = \text{“do } R_1 \text{”} \): We do \( R_1 \) once and \( R_2 \) zero times. This is contained in \( R_1^* R_2^* \).
  - if \( A_i = \text{“do } R_2 \text{”} \): We do \( R_1 \) zero times and \( R_2 \) once. This is contained in \( R_1^* R_2^* \).
- The entire for-loop is contained in \((R_1^* R_2^*)^*\).

**Right is contained in left:** Now we show that the language described by \((R_1^* R_2^*)^*\) is contained in the language described by \((R_1 \cup R_2)^*\).

Take an arbitrary string in the language described by \((R_1^* R_2^*)^*\). There is an integer \( k \geq 0 \), such that this string is described by “\( k \) times, do \( R_1^* R_2^* \)”. Think of this as “do \( A_1, A_2, \ldots, A_k \)”, where each \( A_i \) is \( R_1^* R_2^* \). We show that this is contained in \((R_1 \cup R_2)^*\):

- For \( i = 1, 2, \ldots, k \):
  - Since \( A_i \) is “do \( R_1^* R_2^* \)”, there are integers \( m_i \geq 0 \) and \( n_i \geq 0 \), such that \( A_i \) is “\( m_i \) times, do \( R_1 \), followed by \( n_i \) times, do \( R_2 \)”. This is contained in “\( m_i + n_i \) times, do \( R_1 \cup R_2 \)”. Thus, \( A_i \) is contained in \((R_1 \cup R_2)^*\).
- The entire for-loop is contained in \((R_1 \cup R_2)^*\).
**Question 4:** In this question, the alphabet is \{0, 1\}. Let \( A \) be the language consisting of all bitstrings that are the binary representation of an integer at least equal to 40. (Assume that the leftmost bit in the binary representation of a positive integer is 1. For example, the integer 41 in binary is 101001 and not 0101001.) Give a regular expression that describes the language \( A \). As always, justify your answer.

**Solution:** The solution will be based on the following observations:

- Every integer at least equal to 40 has at least six bits in its binary representation.
- Every integer that has at least seven bits in its binary representation is at least equal to \( 2^6 = 64 \) and, therefore, at least equal to 40.
- Every integer at least equal to 40 that has exactly six bits in its binary representation is
  - either \( 11**\), where each \( * \) is 0 or 1,
  - or \( 101**\), where each \( * \) is 0 or 1.

This leads to the regular expression

\[
1(0 \cup 1)(0 \cup 1)(0 \cup 1)(0 \cup 1)(0 \cup 1)(0 \cup 1)^* \\
\cup \\
11(0 \cup 1)(0 \cup 1)(0 \cup 1)(0 \cup 1) \\
\cup \\
101(0 \cup 1)(0 \cup 1)(0 \cup 1)
\]

**Question 5:** Use the construction given in class to convert the regular expression

\[
(a \cup bb)^* (ba^* \cup \varepsilon)
\]

to an NFA. Do not simplify your NFA; just apply the construction rules “without thinking”.

**Solution:** We first consider how the regular expression is “built”:

- Take the regular expression \( a \).
- Take the regular expression \( b \).
- Take the regular expressions \( b \) and \( b \), and turn them into the regular expression \( bb \).
- Take the regular expressions \( a \) and \( bb \), and turn them into the regular expression \( a \cup bb \).
- Take the regular expression \( a \cup bb \), and turn it into the regular expression \( (a \cup bb)^* \).
• Take the regular expression $a$, and turn it into the regular expression $a^*$.  

• Take the regular expressions $b$ and $a^*$, and turn them into the regular expression $ba^*$.  

• Take the regular expression $\varepsilon$.  

• Take the regular expressions $ba^*$ and $\varepsilon$, and turn them into the regular expression $ba^* \cup \varepsilon$.  

• Take the regular expressions $(a \cup bb)^*$ and $ba^* \cup \varepsilon$, and turn them into the regular expression $(a \cup bb)^*(ba^* \cup \varepsilon)$.

First, we construct an NFA $M_1$ that accepts the language described by the regular expression $a$:

$$M_1$$

Next, we construct an NFA $M_2$ that accepts the language described by the regular expression $b$:

$$M_2$$

Next, we apply the concatenate construction to $M_2$ and $M_2$. This gives an NFA $M_3$ that accepts the language described by the regular expression $bb$:

$$M_3$$

Next, we apply the union construction to $M_1$ and $M_3$. This gives an NFA $M_4$ that accepts the language described by the regular expression $a \cup bb$:

$$M_4$$

Next, we apply the star construction to $M_4$. This gives an NFA $M_5$ that accepts the language described by the regular expression $(a \cup bb)^*$:
Next, we apply the star construction to $M_1$. This gives an NFA $M_6$ that accepts the language described by the regular expression $a^*$:

Next, we apply the concatenate construction to $M_2$ and $M_6$. This gives an NFA $M_7$ that accepts the language described by the regular expression $ba^*$:

Next, we construct an NFA $M_8$ that accepts the language described by the regular expression $\varepsilon$:

Next, we apply the union construction to $M_7$ and $M_8$. This gives an NFA $M_9$ that accepts the language described by the regular expression $ba^* \cup \varepsilon$:

Finally, we apply the concatenate construction to $M_5$ and $M_9$. This gives an NFA $M_{10}$ that accepts the language described by the regular expression $(a \cup bb)^*(ba^* \cup \varepsilon)$:
This was fun eh!

**Question 6:** Use the construction given in class to convert the following DFA to a regular expression.

**Solution:** For each state $q_i$, $i = 0, 1, 2$, we define $L_i$ to be the set of all strings $w$ in $\{a, b\}^*$ such that the path in the state diagram that starts in state $q_i$ and corresponds to $w$ ends in the accept state $q_1$. We obtain the following set of equations:

\[
L_0 = aL_0 \cup bL_1 \\
L_1 = \varepsilon \cup aL_0 \cup bL_2 \\
L_2 = aL_2 \cup bL_1
\]

Since $q_0$ is the start state, we need a regular expression for $L_0$.

We use the following tool to solve these equations:

If $L = BL \cup C$ and $\varepsilon \not\in B$, then $L = B^*C$.  \hspace{1cm} (4)

We solve the equations (1), (2), and (3), in the following way: Equation (3) is in the form of (4). This gives

\[
L_2 = a^*bL_1.
\]
By substituting this into (2), we obtain
\[ L_1 = \varepsilon \cup aL_0 \cup ba^*bL_1, \]
which we rewrite as
\[ L_1 = ba^*bL_1 \cup (\varepsilon \cup aL_0). \]
This equation is in the form of (4). This gives
\[ L_1 = (ba^*b)^*(\varepsilon \cup aL_0) = (ba^*b)^* \cup (ba^*b)^*aL_0. \]
By substituting this into (1), we obtain
\[ L_0 = aL_0 \cup b (ba^*b)^* \cup b (ba^*b)^*aL_0 = (a \cup b (ba^*b)^*a) L_0 \cup b (ba^*b)^*. \]
This equation is in the form of (4). This gives
\[ L_0 = (a \cup b (ba^*b)^*a)^* b (ba^*b)^*. \]

**Question 7:** Is the language
\[ L = \{a^kb^la^m : k \geq 0, \ell \geq 0, m \geq 0, k + \ell + m \geq 5\} \]
regular? As always, justify your answer.

**Solution:** The language \( L \) is regular. To prove this, we are going to write \( L \) as the union of finitely many languages, each of which is regular.

- For each triple \((x, y, z)\) of non-negative integers for which \( x + y + z = 5 \), consider the regular expression
  \[ R(x, y, z) = \underbrace{a \cdots a}_{x} a^* b \underbrace{\cdots b}_{y} a^* b \underbrace{\cdots a}_{z} a^*. \]
  This regular expression describes a regular language. Each string in this language belongs to \( L \).

- In COMP 2804, you have learned that the number of such triples \((x, y, z)\) is equal to \( \binom{5}{2} = 21 \); in particular, this number is finite.

It remains to show that each string in \( L \) is in the language of some regular expression \( R(x, y, z) \).

Let \( k \geq 0, \ell \geq 0, \) and \( m \geq 0 \) be such that \( k + \ell + m \geq 5 \). Consider the string \( a^kb^la^m \).

We may assume without loss of generality that \( m \leq \ell \leq k \). Observe that \( k \geq 2 \).

- Assume that \( k = 2 \). Then \( \ell = 2 \) and \( m \geq 1 \). The string \( a^kb^la^m \) is in the language described by the regular expression \( R(2, 2, 1) \).
• Assume that $k = 3$. Then $\ell \geq 1$.
  
  – Assume that $\ell = 1$. Then $m = 1$. The string $a^k b^\ell a^m$ is in the language described by the regular expression $R(3, 1, 1)$.
  
  – Assume that $\ell \geq 2$. The string $a^k b^\ell a^m$ is in the language described by the regular expression $R(3, 2, 0)$.

• Assume that $k = 4$. Then $\ell \geq 1$. The string $a^k b^\ell a^m$ is in the language described by the regular expression $R(4, 1, 0)$.

• Assume that $k \geq 5$. The string $a^k b^\ell a^m$ is in the language described by the regular expression $R(5, 0, 0)$.

**Question 8:** For any string $w \in \{a, b\}^*$, we denote the number of $a$’s in $w$ by $N_a(w)$, and we denote the number of $b$’s in $w$ by $N_b(w)$. Consider the language

$$A = \{w \in \{a, b\}^* : N_a(w) = N_b(w)\}.$$ 

Assume that we are going to use the Pumping Lemma to prove that $A$ is not regular. As always, we assume that $A$ is regular. The Pumping Lemma gives us a pumping length $p$.

1. Explain in a few sentences why we may assume that $p$ is even.

2. Given that $p$ is even, can we choose the string $s = a^{p/2}b^{p/2}$ to obtain a contradiction?

**Solution:** For the first part, the Pumping Lemma gives us an integer $p$ such that every string in $A$ having length at least $p$ can be “pumped”. This implies that every string in $A$ having length at least $p + 1$ can also be “pumped”. If $p$ happens to be odd, then we can take $p + 1$ to be the pumping length.

For the second part, we assume that $p$ is even. Consider the string $s = a^{p/2}b^{p/2}$. Then $s \in A$ and $|s| = p$. By the Pumping Lemma, we can write $s = xyz$, where

1. $y \neq \epsilon$,

2. $|xy| \leq p$, and

3. $xy^iz \in A$, for all $i \geq 0$.

It may happen that $y = a^k b^k$ for some $k$ with $1 \leq k \leq p/2$. In this case, for every $i \geq 0$, the string $xy^iz$ contains as many $a$’s as $b$’s. Thus, each such string is in $A$. In other words, this string $s$ is the wrong string.

**Question 9:** Use the Pumping Lemma to prove that the following languages are not regular. In all cases, the alphabet is $\{a, b\}$.

1. $\{a^k b^\ell a^m : k \geq 0, \ell \geq 0, m \geq 0, \text{ and } k = \ell \text{ or } \ell \neq m\}$.
2. \( \{a^mb^n : m \geq 0, n \geq 0, \ m + n \text{ is a prime number} \} \).

**Solution:** First, we do

\[
A = \{a^k b^\ell a^m : k \geq 0, \ell \geq 0, m \geq 0, \text{ and } k = \ell \text{ or } \ell \neq m \}.
\]

Assume the language \( A \) is regular. Let \( p \geq 1 \) be the pumping length, as given by the Pumping Lemma. Let \( s = a^p b^p a^p \). Then \( s \in A \) and \( |s| = 3p \geq p \). By the Pumping Lemma, we can write \( s = xyz \), where

1. \( y \neq \epsilon \),
2. \( |xy| \leq p \), and
3. \( xy^iz \in A \), for all \( i \geq 0 \).

Since \( |xy| \leq p \), the string \( y \) contains only \( a \)'s from the left block of \( a \)'s in \( s \). Since \( y \neq \epsilon \), the string \( y \) contains at least one \( a \). Let \( k = |y| \), so that \( y = a^k \). Note that \( 1 \leq k \leq p \). By the Pumping Lemma, the string

\[
s' = xy^2z = a^{p+k} b^p a^p
\]

is in \( A \). However, \( s' \) is not in \( A \), because the left \( a \)-block is longer than the \( b \)-block, and the \( b \)-block has the same length as the right \( a \)-block. Thus, we have a contradiction, and we can conclude that \( A \) is not regular.

Next we do

\[
B = \{a^m b^n : m \geq 0, n \geq 0, \ m + n \text{ is a prime number} \}.
\]

Since the value of \( n \) can be zero, we can copy the proof done in class: Take the string \( s = a^m b^0 = a^m \), where \( m \) is a prime number with \( m \geq p \) and \( p \) is the pumping length .

Here is a different proof (it is basically the same). Assume the language \( B \) is regular. Let \( p \geq 1 \) be the pumping length, as given by the Pumping Lemma. Let \( n \) be a prime number such that \( n \geq p \). Let \( s = a^p b^{n-p} \). Then \( s \in B \) and \( |s| = n \geq p \). By the Pumping Lemma, we can write \( s = xyz \), where

1. \( y \neq \epsilon \),
2. \( |xy| \leq p \), and
3. \( xy^iz \in B \), for all \( i \geq 0 \).

Since \( |xy| \leq p \), the string \( y \) contains only \( a \)'s. Since \( y \neq \epsilon \), the string \( y \) contains at least one \( a \). Let \( k = |y| \), so that \( y = a^k \). Note that \( 1 \leq k \leq p \). By the Pumping Lemma, for each \( i \geq 0 \), the string

\[
xy^iz = a^{p+(i-1)k} b^{n-p}
\]
is in \( B \). Therefore, for each \( i \geq 0 \),
\[
p + (i - 1)k + n - p = (i - 1)k + n
\]
is a prime number. However, for \( i = n + 1 \) we have
\[
(i - 1)k + n = nk + n = n(k + 1),
\]
which is not a prime number because \( n \geq 2 \) and \( k + 1 \geq 2 \). Thus, we have a contradiction, and we can conclude that \( A \) is not regular.

**Question 10:** In this question, the alphabet is \( \{a, b\} \).

1. Explain in a few sentences why the language
\[
\{a^n b^n : n \geq 1\}
\]
is not regular, where the “bar” denotes the complement. You may use any result that was shown in class.

2. Explain in a few sentences why every finite language is regular. You may use any result that was shown in class.

3. Give an example of two languages \( A \) and \( B \), such that \( A \) is finite (and, thus, regular), \( AB \) is regular, but \( B \) is not regular.

4. Give an example of two languages \( A \) and \( B \), such that \( A \) is infinite and regular, \( AB \) is regular, but \( B \) is not regular.

**Solution:** For the first part:

- We have seen in class that \( \{a^n b^n : n \geq 1\} \) is not regular.

- We have seen in class that the complement of a regular language is regular. This implies that the complement of a non-regular language is non-regular.

For the second part:

- Let \( A \) be the language consisting of one single string, say, \( w_1 w_2 \ldots w_n \). Then \( A \) is regular, because it is described by the regular expression \( w_1 w_2 \ldots w_n \).

- Let \( B \) be any finite language, consisting of, say, \( N \) strings. Then, \( B \) is the union of \( N \) many one-string languages. Since the union of finitely many regular languages is regular, it follows that \( B \) is regular.

For the third part, take \( A = \emptyset \) and take for \( B \) your favorite non-regular language. Note that \( AB = \emptyset \), which is regular.

Alternatively, we can take \( A = \{\varepsilon, a\} \) and take for \( B \) the non-regular language in the first part. You can check that \( AB = \{a, b\}^* \), which is regular.

For the fourth part, take \( A = \{a, b\}^* \) and \( B = \{a^n b^n : n \geq 0\} \). You can check that \( AB = \{a, b\}^* \), which is regular.