Problem 1. Why do we need to study automata? Provide two main reasons.

Problem 2–4. Let us consider the automaton that has two states: \( n \) (the student is in normal mood) and \( h \) (the student is happy); \( n \) is the starting state, \( h \) is the final state. The only two symbols are \( g \) (the student got a good grade on this test) and \( t \) (time passed).

- From \( n \), \( g \) leads to \( h \), and \( t \) leads back to \( n \).
- From \( h \), \( g \) leads back to \( h \) and \( t \) leads to \( n \).

Problem 2. Trace, step-by-step, how this finite automaton will check that the word \( gtg \) belongs to this language. Use the tracing to find the parts \( x \), \( y \), and \( z \) of the word \( gtg \) corresponding to the Pumping Lemma. Check that the “pumped” word \( xyz \) will also be accepted by this automaton.

Problem 3. Write down the tuple \( (Q, \Sigma, \delta, q_0, F) \) corresponding to this automaton:

- \( Q \) is the set of all the states,
- \( \Sigma \) is the alphabet, i.e., the set of all the symbols that this automaton can encounter;
- \( \delta : Q \times \Sigma \rightarrow Q \) is the function that describes, for each state \( q \) and for each symbol \( s \), the state \( \delta(q, s) \) to which the automaton that was originally in the state \( q \) moves when it sees the symbol \( s \) (you do not need to describe all possible transitions this way, just describe two of them);
- \( q_0 \) is the staring state, and
- \( F \) is the set of all final states.

Problem 4. Use a general algorithm that we had in class to generate a context-free grammar corresponding to this automaton. Show how this grammar will generate the word \( gtg \).

Problem 5. Let \( A_1 \) be the automaton described in Problem 2. Let \( A_2 \) be an automaton that accepts all the strings that contain at least one symbol \( g \) – indicating good news. This automaton has two states: the starting state \( s \), and the final state \( f \). The transitions are as follows:
• from the start state, \( t \) lead back to the start state, while \( g \) leads to the final state \( f \);

• from the final state \( f \), any symbol leads back to this state.

Use the algorithm that we had in class to describe the following two new automata:

• the automaton that recognizes the union \( A_1 \cup A_2 \) of the two corresponding languages, and

• the automaton that recognizes the intersection of the languages \( A_1 \) and \( A_2 \).

**Problem 6.** Use the general algorithm that we learned in class to design a non-deterministic finite automaton that recognizes the language \((g \cup t)^*g\):

• first, describe the automata for recognizing \( g \) and \( t \);

• then, combine them into the automata for recognizing the union \( g \cup t \), and the Kleene star \((g \cup t)^*\);

• finally, combine the automata for \((g \cup t)^*\) and \( g \) into an automaton for recognizing the desired composition of the two languages.

**Problem 7.** Use the general algorithm to transform the resulting non-deterministic finite automaton into a deterministic one.

**Problem 8–9.** Use a general algorithm to transform the finite automaton from Problem 2 into the corresponding regular expression. Start with eliminating the state \( n \).

**Problem 10.** Prove that the language \( L \) of all the words that have more \( g \)'s than \( t \)'s is not regular.