Name:

General comments:

- you are allowed up to 5 pages of handwritten notes;
- if you need extra pages, place your name on each extra page;
- the main goal of most questions is to show that you know the corresponding algorithms; so, if you are running out of time, just follow the few first steps of the corresponding algorithm.

Good luck!
1. Design a pushdown automaton that would recognize the words of the type $a^{2n}b^n$, i.e., the language $L = \{a, aab, aaaabb, \ldots\}$. Show, step by step, how your pushdown automaton will recognize the word $aab$. *Hint:* use a pushdown automaton for recognizing the words of the type $a^nb^n$ as a sample. The difference is that in that case, when we saw a letter 'b', we popped one symbol 'a' from the stack. Now it is slightly different.
2. Design a context-free grammar that would generate all the words of the type $a^{2n}b^n$, i.e., the language $L = \{\Lambda, aab, aaaaab, \ldots\}$. Show, step by step, how your grammar will generate the word $aab$. 

*Hint:* use a context-free grammar for generating the words of the type $a^nb^n$ as a sample. There, we had rules $S \rightarrow \varepsilon$ and $S \rightarrow aSb$, now a slight modification is needed.
3. Use a general algorithm that we had in class to generate a context-free grammar that corresponds to the following finite automaton for recognizing signed or unsigned binary integers. This automaton has three states: the starting state s, the final state f, and the sink state k.

- From s, any symbol (+, −, 0, or 1) lead to f.
- From f, 0 or 1 lead to f, while + or − lead to sink.

Show, step by step, how your grammar generates a string +01.
4. Transform, step by step, the grammar with rules \( S \rightarrow \varepsilon \) and \( S \rightarrow 1S0 \) to Chomsky normal form. Show how the word 10 will be generated in the resulting Chomsky-normal-form grammar.
5. Use a general algorithm for transforming CFG into PDA to design a pushdown automaton which is equivalent to the grammar with rules $S \rightarrow \varepsilon$ and $S \rightarrow 1S0$. Show, step by step, how the word 10 will be accepted by the resulting pushdown automaton.
6. Use the general stack-based algorithms to show:

- how the compiler will transform the expression \((11 + 1) / (20 - 17)\) into postfix form, and
- how it will compute the value of the resulting postfix expression.
7. Use a general algorithm for transforming PDA into CFG to design a CFG that corresponds to the following pushdown automaton. This automaton has two states: the 1-state $s_1$ and the 0-state $s_0$. Both states are final. The transitions are:

- From $s_1$ to $s_1$, the transition is 1, $\varepsilon \rightarrow 1$.
- From $s_1$ to $s_0$, the transition is 0, $1 \rightarrow \varepsilon$.
- From $s_0$ to $s_0$, the transition is 0, $1 \rightarrow \varepsilon$.

Show, step by step, how the word 10 will be generated by the resulting grammar.