1. **General hints**

1. Start early! Homework assignments take much longer than just an hour to finish. Starting early allows you to better understand the problem statements and write correct and complete solutions. Furthermore you can get help on the course forums in case you need it. Seeing the homework sheet for the first time on the day it is due is highly discouraged.
2. Read the course notes before starting work on the homework. Many of the questions posted on the course forums are already answered in the slides and course notes. Save time and unnecessary discussions by reading the provided material first.
3. Do not write sloppy solutions. Presenting a single scanned graph with no explanations is not sufficient.
4. Try to minimize the submission of scanned problems. There are plenty of free tools available that can help you design circuits, graphs, tables and so on. Learn to write your solution on a computer early on as this will bring benefits for you.

2. **Math tasks**

When writing a math proof:

1. Think about the solution and make sure you understand the problem.
2. Check if your solution is complete and correctly deals with special/border cases.
3. Write clear and rigorous proofs.
4. Do not write essays. Write concise solutions and use math talk.
5. If you introduce new concepts or symbols, clearly define their meaning.

3. **Programming tasks**

When writing a program:

1. If you are given a function name make sure you use it in the exact same way in your solution. If a function *foo* is required, don't provide a solution with a name *Foo, FOO* or *myfoo*.
2. Whenever a function signature is given, you must adhere to it. Even if you think a different way might be better.
3. For all programming assignments (SML, Assembler, TM, and Prolog), always write your solution on a computer, compile it and make sure it is correct.
4. Comment your code well. Always explain more complicated parts of your program with meaningful comments. Unless your program consists of a few lines, have a comment in the beginning explaining what the idea of the solution is. For more details on this point see the code structure guidelines below.
5. If you cannot complete a programming task before the deadline, make sure to at least describe your idea and comment on what the possible reasons for your program not working might be.

4. **Code formatting guidelines**

SML programs:

- Always include a comment header in your source files. The header should at least include your name, homework and problem number.
- Explain briefly how more complicated parts of your code work. If you are writing a program of more than just a few functions explain the general idea of your algorithm.
- Use new lines to separate logical parts of your program (e.g. function declarations).
- Do not cram long functions on a single line. Write each new function case on a new line and use white spaces in the function body.
- Indent your programs! If a piece of code is a subpart of a bigger block indent it to the right. This includes function bodies, the inside blocks of if...then...else, let...in...end, etc.
- Name your functions and variables according to their meaning. At the same time do not use very long names.

- Be consistent. Use the same formatting style throughout your homework. Please, see also a good and a bad example of SML code!

6. **Homework checklist**

Here is a brief checklist for submitting your homework:

- All tasks:
  - ✔All submitted files have a header with your name and problem number
  - ✔Details of the solutions are consistent with the course lecture notes
- Math tasks:
  - ✔The proofs are concise and use math talk
  - ✔Special cases have been considered
- Programming tasks:
  - ✔The code compiles and has been tested
  - ✔Function signatures are identical to the problem statement
  - ✔The solution is appropriately documented and easy to read
  - ✔For SML programs, exceptional situations have been identified and handled properly

7. **Point deduction**

Each solution will be graded individually based on its correctness and marked appropriately. After that further points will be deducted according to the following scheme:

- 10%
  - ✗Missing file header
  - ✗Extensive math proof which hardly uses math talk
- 20%
  - ✗Missing or irrelevant comments for a long programming exercise
- 30%
  - ✗Function signatures/names are different from the problem statement
  - ✗Code formatting is inconsistent and clearly doesn't follow the guidelines
- 50%
  - ✗A programming solution doesn't compile and contains no explanation as to what the problem might be

These are percentages from the maximal achievable points for each problem. If more than one of the guidelines above is not followed only the biggest penalty will apply.

References:

[1] Anca Drăgan, Lucia Ambrošová, Dimitar Asenov: General CS I & II homework writing guidelines, v2.0 – 9 September 2008