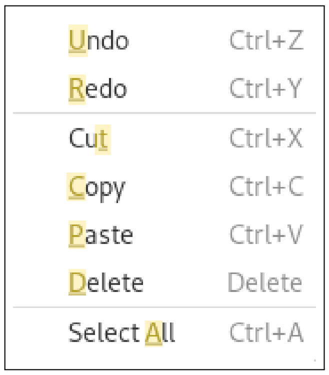
Assignment 1
Keyboard Mnemonics

Static Program Analysis and Constraint Solving
Master's Degree in Formal Methods in Computer Science

Submission deadline: October, 3rd

Submission instructions: Students are required to submit the source code of the program and (if applicable) the generated SMT-LIB script. The program has to contain comments explaining how the problem is formalized in propositional logic, and which constraints have been generated.

When you work with any computer application, you may have realized that some of the letters that appear in the names of the menus, buttons, etc. are underlined. Maybe these underlined letters may not be visible at first sight but you have to press a certain key (for example, ALT) to activate them. These letters are named *mnemonics*, and they allow us to quickly select, by means of the keyboard, an option of the menu.



Normally, the person who designs the user interface of an application is responsible for assigning a mnemonic to each of the options. Ideally, the mnemonic would be the first letter that appears in the option's text, so that it is easier to remember. In this way, we would have the mnemonic *F* for the *F̲ile* option, mnemonic *E* for the *E̲dit* option, and so on. Unfortunately this is not always possible, since we do not want two options in the same menu to have the same mnemonic. This happens, for example, in a menu containing *Copy* and *Cut*. Since both options start with *C*, you have to use, in one of them, another letter as a mnemonic. For example: *C̲opy* and *Cut̲*.

Given a set of options in a menu, we want to assign a mnemonic to every option such that there are not two options with the same mnemonic. This problem can be reduced to SAT. Assume a menu with n options and, for every $i \in \{1, \ldots, n\}$ let us denote by $\mathit{Chars}(i)$ the set of characters that belong to the i-th option's text. We encode our problem as a set of propositional variables $\{u_{c,i} \mid 1 \leq i \leq n, c \in \mathit{Chars}(i)\}$ so that $u_{c,i}$ is true if and only if the character c is the mnemonic of the i-th option.

Given this encoding, our solution has to meet the following constraints:

1. Each option must have a mnemonic.

2. An option cannot have more than one mnemonic.

3. A given letter cannot be a mnemonic of two different options.

Formalize these restrictions as sentences in propositional logic. These sentences should mention only the $u_{c,i}$ variables, and cannot contain quantifiers. Then, in your favorite programming language, implement a program that receives a list of strings as input, each one denoting the text of an option, and determines a possible assignment of mnemonics to options. You can use one of these approaches:

- Your program generates an SMT-LIB script that is subsequently executed by Z3, **or**
- Your program uses language-specific bindings to Z3 API. Currently supported languages are C, C++, Java, Python, ML/OCaml, and those targeting the .NET platform.