# C Developer preliminary test

**Task:** Develop a simple bash-like scripting language interpreter for testing shared functions libraries. A program should be developed with C language usage as a console application for the OS GNU/Linux (x86 processor architecture) with API POSIX.1-2001 usage.

**Scripting language syntax:** The script file is a text file with an arbitrary name. The rule applies: one

line is one command.

There are following commands in the language.

1. Loading the library

   *use <so_path> as <alias>*

   *<so_path>* - the path to the file *.so

   *<alias>* – alias name for the loaded library ( to be referred to in other commands)

   Possible errors:

   - The file not found
   - The file is not a shared library;
   - Alias is already used
2. Unloading the library

   *rem <alias>*

   *<alias>* - the alias of the loaded library

   Possible errors:

   - The alias is not found from list of loaded libraries
3. Library function call

   *call [alias.]<func_name> [args]*

   *<func_name>* - the name of the function

   *[alias.]* – optionally, the alias of a loaded library can be provided, followed by a dot (**.**) to separate from func_name, to explicitly indicate where to look for the function. If alias is not provided, the most recently loaded library which contains the function should be used.

   *[args]* – optional input arguments ( each argument is a text string ) of the function, number of arguments may vary

   Possible errors:

   - the function is not found
   - the library ( referred by alias ) is not loaded

- function returned an error code ( a non-zero value )

All errors must not cause the script to stop executing.

Support of comments and line breaks should also be provided:

- Comments: lines starting with a hash ( **#** ) or the ending part of a line after a **#** should be treated as comment texts and hence ignored by interpreter
- Line breaks: if a line ends with a backslash ( **\** ), the next line should be appended into current line ( after removing backslash ) before being processed

Optionally, following additional features may be provided:

- Interactive mode of the application (entering commands directly from the terminal): in this mode, a new command 'quit' should be supported to allow clean exit of the application.
- Introduce ability to use variables:
    - New command 'set' should be introduced to define a variable: set <var_name>=<value>
    - Variable values should be passed as argument of a function call by using **$**<var_name>

**Description of test libraries:**
All test libraries should be developed with C language usage. All exported functions should have prototype same as main() function:

*int example (int argc, char *argv[]);*

**Requirements to the application:**
The application should be launched from the command line with only one parameter – the path to the file script. An example:

*sotest test.sc*

If no parameter is provided, the application may enter Interactive mode ( if the feature is already supported ).

**Example of the script:**

```
use test.so as test
call example
call test1_func abc 123 #2 arguments are provided
use ./other.so as test2
call test2.func1 \
def456 # this line should belong to above line
call test.example
rem test2
#Expect error in below command
call test2.func1
```

**The result of the test task:**

- Application source code (compilation instruction should be included)
- Self-testing result of the application (to demonstrate that the application works properly in candidate's own machine), including:
    - Screenshot/console output of the application execution
    - The script ( .sc file ) used for testing
    - The source of the libraries used for testing