Algorithm 1 isSingleton

Input: *list* $\in \mathsf{STRING}^N$, *where* $\in \mathsf{STRING}$ (with $N \in \mathsf{N}_{\geq 0}$)
Output: *bool* $\in \mathsf{BOOL}$

```
if list contains no dublications then
    bool ← true
else
    bool ← false
    print("ERROR: name used more than once in " + where)
end if
```

Algorithm 2 parserLevelOne

Input: *file* with ending .Cunt
Output: *func_names*, *func_args*, *func_bodys*, *is_poss*

```
is_poss ← true
N ← number of "func" infile
for i = 1 : N do
    name ← next_word_after_func
    args ← evythg_in_brc
    is_poss ← is_poss * isSingleton(args, "the args of function " + name)
    body ← evythg_in_cbrc
    func_names ← name
    func_args ← (name, args)
    func_bodys ← (name, body)
end for
is_poss ← is_poss * isSingleton(args, "naming functions")
if "main" is not in func_names then
    is_poss ← false
    print("ERROR: no main to execute")
end if
```

Algorithm 3 parserLevelOne

Input: *file* with ending .Cunt
Output: *function_list*

```
is_poss ← true
N ← number of "func" in file
for i = 1 : N do
    name ← next_word_after_func
    args ← evythg_in_brc
    is_poss ← is_poss * isSingleton(args, "the args of function " + name)
    body ← evythg_in_cbrc
    func_names ← name
    func_args ← (name, args)
    func_bodys ← (name, body)
end for
is_poss ← is_poss * isSingleton(args, "naming functions")
if "main" is not in func_names then
    is_poss ← false
    print("ERROR: no main to execute")
end if
```