https://github.com/onodibianca/FLCD/tree/master/Lab6

```
def expand(
    state,              # -- String
    index,              # -- Integer
    working_stack,      # -- list of pair of strings
    input_stack,        # -- list of strings
    production,         # -- String
    number_of_production,   # -- Integer
    grammer
    ):
    working_stack.add((grammer.P[number_of_production], number_of_production))
    input_stack.insert(0, production)
    return state, index, working_stack, input_stack
#called only if variable is Terminal
def advance(
    state,              # the state of the parsing                  -- String
    index,              # the current index of the parsing          -- Integer
    working_stack,      # the working_stack of the parsing          -- list of strings and pair of string
    input_stack         # the input_stack of the parsing            -- list of strings
    ):
    working_stack.add(input_stack[index])
    del input_stack[index]
    index += 1
    return state, index, working_stack, input_stack
def momentary_insuccess(
    state,              # the state of the parsing                  -- String
    index,              # the current index of the parsing          -- Integer
    working_stack,      # the working_stack of the parsing          -- list of strings and pair of string
    input_stack         # the input_stack of the parsing            -- list of strings
```

```
    ):
    state = 'b' # (this changes the current state of the parsing to "back" state)
    return state, index, working_stack, input_stack
def back(
    state,              # the state of the parsing                    -- String
    index,              # the current index of the parsing            -- Integer
    working_stack,      # the working_stack of the parsing            -- list of strings and pair of string
    input_stack         # the input_stack of the parsing              -- list of strings
    ):
    index -= 1
    head_working_stack = working_stack.pop()
    input_stack.insert(0, head_working_stack)
        return state, index, working_stack, input_stack
#called when nonTerminal
def another_try(
    state,              # the state of the parsing                    -- String
    index,              # the current index of the parsing            -- Integer
    working_stack,      # the working_stack of the parsing            -- list of strings and pair of string
    input_stack,        # the input_stack of the parsing              -- list of strings
    productions_list    # the list with all productions               -- list of Strings
    ):
    index_aux = working_stack[len(working_stack) - 1][1]
    if index_aux == 1 and input_stack[0] == "A":
        state = 'e'
        return state, index, working_stack, input_stack
    if index_aux <= len(productions_list) - 1 :
        state = 'q'
        working_stack.append(("A", index_aux))
        input_stack.insert(0, productions_list[index_aux])
```

```
        return state, index, working_stack, input_stack
    else:
        input_stack.insert(0, "S")
        return state, index, working_stack, input_stack
def success(
    state,              # the state of the parsing                    -- String
    index,              # the current index of the parsing            -- Integer
    working_stack,      # the working_stack of the parsing            -- list of strings and pair of string
    input_stack         # the input_stack of the parsing              -- list of strings
    ):
    state = 'f' # (this changes the current state of the parsing to "final" state)
    return state, index, working_stack, input_stack
```