Team Members:

1. احمد اسامة عبدالخالق الشامي
2. يوسف حمادة ابراهيم محمود
3. اسلام عامر فؤاد محمد

The Idea: Our project is about a simple language that gives a robot some commands to move (forward or back) and turn (right or left), the language must have a compiler (Scanner and Parser) to check the correctness of it.

Description For the Language: Our language is to give the robot some commands and map to move around. Mapping: containing 1-Table: is 2D array that contain some symbols {_, =} to simulate the room that robot moves in 2D coordinate systems (for the computer screen), underscore '_' is referred to empty place and '=' is referred to full space.

2- containing coordinates where the robot's first place.

_	_	=	_	=
#	_	_	_	_
_	_	=	_	_
_	=	=	_	_

(1, 2) is Robot's first place (#)

for mapping {[_ _ = _ =], [_ _ _ _ _], [_ _ = _ _],[_ = = _ _] (1, 2) LEFT}

so if it moves forward the result at 2nd row will be [_ # _ _ _]

3- R-Direction: is the initialized Robot Forward direction (Is the direction that the robot looks to at the start).

Forward Direction: is the Direction that the robot looks to.

move (forward, backward) and turn (left, right), but you must use STOP key word after any movement or movements and you cannot use it except after movements, and you can repeat some commands more than one time using repeat statement.

We NOTE that the mapping is to simulation but Commands is to move robot so the movement directions of robots (Forward and backward) but at mapping (simulation) we have 4 directions (up, down , right, left).

BNF Grammar:

<Program> ::={ <Mapping > } { <Commands> }

<Mapping>::=<Table> (<Digits>,<Digits>) <R-Direction>

<R-Direction>::=Up | Down | <Direction>

<Table>::= <Table> , [<Row>] | [<Row>]

<Row>::= <Row> <Char> | <Char>

<Char>::= _ | =

<Commands> ::= <Commands> , <Command> | <Command>

<Command> ::= <Move> STOP | <Turn> | <Repeat>

<Move> ::= <Move> MOVE <M-Direction> | MOVE <M-Direction>

<Turn> ::= TURN <Direction>

<M-Direction> ::= FORWARD | BACK

<Direction> ::= RIGHT | LEFT

<Repeat> ::= REPEAT (<Digits>) {<Commands>}

<Digits> ::= <Digits> <Digit> | <Digit>

<Digit> ::= 0 | 1 | 2 | …| 9

Sample Examples of accepted inputs:

- {[_ , _ , =] [_ , _ , _] (0 , 1) RIGHT } {MOVE FORWARD, STOP, TURN RIGHT, MOVE BACK, STOP}
- {[_ , = , =] [_ , = , _] (2 , 1) BACK } {REPEAT (12) {MOVE BACK, STOP, TURN LEFT}, TURN RIGHT, MOVE FORWARD, STOP, MOVE FORWARD, MOVE FORWARD, STOP}

Sample Examples of rejected inputs:

- {[_ , _ , =] [_ , _ , _] (0 , 1) RIGHT } {REPEAT(5) {MOVE BACK, TURN LEFT}, TURN RIGHT}
- {MOVE FORWARD, MOVE BACK, STOP, TURN RIGHT, MOVE BACK}