Robot Parser:

Implementation:
Stage 0:
Stage 0 was implemented as specified on the handout, according to the grammar.

Stage 1:
Stage 1 was also implemented as specified on the handout, according to the grammar.

Stage 2:
Stage 2 was where the grammar and test's started to become different. I followed the specification in the "full" test cases, rather than the grammar.

Changes made in part 2 were as follows:
- COND ::= RELOP "(" EXP"," EXP ")

Stage 3:
As in Stage 2, the grammar and tests did not line up, changes to the grammar were made as follows:
- NUM ::= "-?[1-9][0-9]*|0" // this is from stage 2, stage 3 reversed this grammar back to stage 1, which I did not do.

Stage 4:
I only attempted part of stage 4:
"Require variables to be declared before they can be used… To handle declarations, you will need to build a set of declared variables when the declaration is parsed, and then check that any variables used in the program are in this set. You might consider making this set part of the map used to store values when the program is executed. Note that a block can be now occur anywhere that a statement can."

The way I did this, is that DECL created a declaration node, which could be done right at the start of the program, or inside any block, as specified in the grammar. A variable could then be assigned later, as long as it was declared. A block can now occur anywhere that a statement can, as requested in the above paragraph.

A test for Stage 4:

```
{vars $count , $test ; $count = 5; }
while (gt($count, 0)){
  move;
  $count = sub($count, 1);
  $test = 2;
}
turnL;
```

My program does not:
Allow infix operators and optional parentheses for both arithmetic and logical expressions.
Allow nested scope, so that variables declared inside a block (i) are only accessible within the block, and (ii) "shadow" any variables of the same name declared in the program or outer blocks.

I have not included any tests which show this, as I did not attempt it at all, hence I did not think I would require tests to show that this does not work.

NOTE: It never specifies how many times to loop, so for the part 3, it loops the same amount of times as statements in the block, while in part 4, it loops infinity.