MAU11602 Assignment 0
Due 2026-01-29

1. (a) Evaluate the postfix expression `1 2 3 4 * + 5 - 6 * +` using the stack based evaluation method described in lecture, showing the stack contents at each step.

   (b) The method given in lecture for parsing postfix expressions involved keeping a running count of the number of integers seen minus the number of operators seen. You don't have to parse the expression above but just list that count.

   (c) Do you see a relation between this running count and the stack? Can you prove that this relation holds in general, not just in the example?

   (d) For simplicity I left out one commonly used arithmetic operation, the unary minus, i.e the operator which takes a single integer as argument and changes its sign. Using a `-` to denote this, as is usual, causes problems do to confusion with the `-` for subtraction. One solution to this problem, adopted in SML, is to use `~` for unary minus. We then need to add a rule to our stack based evaluator to say that when we read a `~` then we pop the top element off the stack and push its additive inverse. We also need to change the parsing algorithm as well though. How should we change it?

2. Consider the postfix grammar for a language with arithmetic operators and relations, as discussed at the end of Lecture 2, i.e.

```
expr ::= bexp | iexp
iexp ::= iexp ws iexp ws oper | int
bexp ::= iexp ws iexp ws rel
oper ::= "*" | "+" | "-"
 rel ::= "<" | "<=" | "=" | ">" | ">="
```

   (a) Prove that for every expression consistent with this grammar the number of integers is the number of operators plus the number of relations plus one.

   (b) Show that the condition which was shown to be necessary in the previous part is not sufficient by giving a list of tokens which can't be parsed even though the number of integers is the number of operators plus the number of relations plus one.