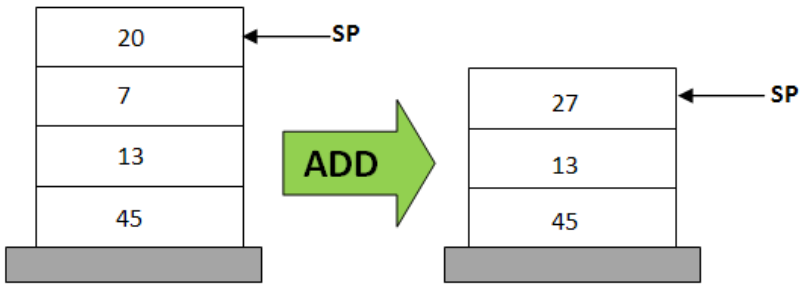
Stack-Based Machine

Description

Stack-base machine is a kind of CPU design architecture which sometimes be compared with register-based machine. Unlike register-based machine which used 3 address code taking 1 opcode and 2 corresponding operands e.g. add %eax, 3, stack based machine was designed with opcode with one or zero operand and one long stack which served as registers to take temporary value in calculation.
It works as follow:



Like the figure above, we may first execute 4 instructions: load 45, load 13, load 7, load 20, then the stack will layout like the stack on left side (note that the "load" instruction is the only operator that takes 1 operand). At the moment, if we continue to execute instruction 'add' which have 0 operand, it will pop two numbers (20, 7) from the top of stack, then execute 'add' with those numbers. Finally push the result (27) back to the stack which will have final layout like the stack on the right side.

In this problem, you will be given a program for stack-based machine. You need to write a yacc program to check if the format of the program and get its result. The instruction set of our machine are:

Instruction	Description
add, sub, mul, div, mod	Pop 2 operands from stack, perform addition, subtraction, multiplication, division and modulation and push back the result. If anything divide zero (e.g. 5/0), output "Divide by zero"
inc, dec	Pop 1 operand from stack increase/decrease operand by 1, then push back the result.
load <number>	Push a constant <number> on the top of the stack.
copy	Copy a constant <number> from stack top, then push it back.
delete	Delete 1 operand from stack.
switch	Pop 2 operands from stack, switch the order of them, and then push them back to the stack.

You will get zero points if you use c or c++ to solve this problem instead of lex and yacc.

Sample Input

load 1
load 1
dec
add
load 10
inc
sub
load 5
inc
inc
mul

Sample Output

70

Sample Input

load 6
load 5
delete
inc
load 0
div

Sample Output

0

Sample Input

load 0
load 4
div

Sample Output

Divide by zero

Sample Input

load 5
load 6
delete
switch

Sample Output

Invalid format