Evaluating infix arithmetic expressions with the 2-stack method.

Stack A is a stack of operators ( use Stack1gen<Character>). The operators are given below.

Stack B is a stack of operands (use Stack1gen<Double>).

Assume that the input infix expression is “correct” (balanced parentheses, etc.) In this first version of the problem, you many assume that each operand is a single digit, 1, 2, …, 9 (no negative values). (In a second version, we will use multi-digit signed decimal values, and then the operand has to be built up from the individual digits and a possible decimal point and/or minus sign.) There may also be blanks, which the program ignores.

Ex. (3 \* (4 + 2 \* (6 – 4) + 1) + 2 ^ 3) / 5

Assume that the expression is examined from right to left.

Assume that a is at the top of the operand stack and b is just below it.

Priority (precedence) order for operators:

Exponentiation a^b

Multiplication, division a\*b, a/b

Addition, subtraction a + b, a - b

Right parenthesis

To process an operand, push it onto the operand stack (stack B).

If the current token is an operator, op, process op as follows:

If op is ), push ) onto A stack

If op has equal priority or strictly higher priority than the operator at the top of stack A, push op onto stack A

If op has strictly lower priority than operator opTop at the top of stack A, execute opTop:

Pop a and b from stack B, perform a opTop b, and push the result onto stack B.

Then re-process op.

If op = ( : In a loop, “execute” (as above) all of the operators on stack A down to the topmost

), which is also popped off stack A. Discard ( and process the character to its left.

After processing the leftmost character in the expression, execute any remaining operators on stack A. The final answer is the single value on stack B, which can be popped and returned as the value of the expression.