```
# qaud_sol.s
# This assembly program calculates the integer solutions of a quadratic polynomial.
# Inputs : The coefficients a,b,c of the equation a*x^2 + b*x + c = 0
# Output : The two integer solutions.
#
# All numbers are 32 bit integers



        .globl main
main:   # Read all inputs and put them in floating point registers.
                li           $v0, 4    # Load print_string syscall code to register v0 for the
1st string.
                la           $a0, str1 # Load actual string to register $a0
                syscall                    # Make the syscall
                li           $v0, 5        # Load read_int syscall code to register v0 for the
coefficient a of a quadratic polynomial
                syscall            # Make the syscall
                move     $t1, $v0  # Move input from register $v0 to register $t1
                li           $v0, 4        # Load print_string syscall code to register v0 for
the 2nd string.
                la           $a0, str2 # Load actual string to register $a0
                syscall                    # Make the syscall
                li           $v0, 5        # Load read_int syscall code to register v0 for the
coefficient a of a quadratic polynomial
                syscall                    # Make the syscall
                move     $t2, $v0  # Move input from register $v0 to register $t2
                li           $v0, 4        # Load print_string syscall code to register v0 for
the 3rd string.
                la           $a0, str3 # Load actual string to register $a0
                syscall                    # Make the syscall
                li           $v0, 5        # Load read_int syscall code to register v0 for the
coefficient a of a quadratic polynomial
                syscall            # Make the syscall
                move     $t3, $v0  # Move input from register $v0 to register $t3

                # In the following lines all the necessary steps are taken to
                # calculate the discriminant of the quadratic equation.
                # As is known D = b^2 - 4*a*c

                li              $t0, 2       # Load constant number to integer register
                mul             $t4,$t2,$t2  # t4 = t2*t2, where t2 holds b
                mul             $t5,$t1,$t3      # t5 = t1*t3, where t1 holds a and t3 holds c
                mul             $t5,$t5,4    # Multiply value of s0 with 4, creating 4*a*c
                sub             $t6,$t4,$t5  # Calculate D = b^2-4*a*c
                tlt             $t6,$0           # If D is less than 0 issue an exception

                # The following lines calculate the Integer result of the square root
                # of  a positive integer number D with a recursive algorithm.
                # x[n+1] = x[n] -(1+2*n), where n is the integer square root of an integer
                # number. x[0], of the step before the loop is D. The algorithm stops
                # when x[n+1] is less than zero.
                li              $s0, 0           # Square Root Partial Result, sqrt(D).
                li      $t7, 1       # Decrement step.
                move    $s1,$t6      # Move value in register t6 to register s1 for safety
purposes.
                sqrtloop:
                sub      $s1,$s1,$t7     # Subtract the decrement step from the x[n]
                bltz $s1,endsqrt         # Check if x[n+1] is less than zero, if yes stop
                addi    $s0,$s0,1        # Increase partial result
                addi    $t7,$t7,2        # Increase by 2 the decrement step
                b sqrtloop                       # Branch unconditionally to sqrtloop label


                endsqrt:
                neg             $s2,$t2          # Calculate -b and save it to s2
                add             $s3,$s2,$s0  # Calculate -b+sqrt(D) and save it to s3
                sub             $s4,$s2,$s0  # Calculate -b-sqrt(D) and save it to s4
                mul             $s5,$t1,$t0  # Calculate 2*a and save it to s5
                div             $s6,$s3,$s5  # Calculate first integer solution
                div             $s7,$s4,$s5  # Calculate second integer solution

                #Print the calculated solutions.
                li      $v0,4               # Load print_string syscall code to register v0 for
the 1st result string.
                la  $a0, str4       # Load actual string to register $a0
                syscall                     # Make the syscall
                li      $v0, 1              # Load print_int syscall code to register v0 for the
1st result string.
                move    $a0, $s6        # Load actual integer to register $a0
                syscall                     # Make the syscall

                li      $v0,4               # Load print_string syscall code to register v0 for
the 1st result string.
                la  $a0, str5       # Load actual string to register $a0
                syscall                     # Make the syscall
                li      $v0, 1              # Load print_float syscall code to register v0 for
the 1st result string.
                move    $a0, $s7        # Load actual float to register $f12
                syscall                     # Make the syscall
                li      $v0, 10             # Load exit syscall code to register v0.
                syscall                     # Make the syscall


.data
str1 : .asciiz "Please enter coefficient a of equation a*x^2 + b*x + c: "
str2 : .asciiz "Please enter coefficient b of equation a*x^2 + b*x + c: "
str3 : .asciiz "Please enter coefficient c of equation a*x^2 + b*x + c: "
str4 : .asciiz "The first integer solution is: "
str5 : .asciiz "\nThe second integer solution is: "
```