In this part of the exam you will write two functions. The first function called `hip` computes the value of a function of variables x and y. The second function invokes `minhip` to scan an specified area in the $x \times y$ plan and returns the minimum value of the function within the specified region.

Question 4 (20 points): Write MIPS assembly code for the function `hip` that computes the value of the following function:

$$f(x, y) = k - x^2 + y^2 \tag{1}$$

The specification for the `hip` function is as follows.

- **parameters:**

  `$a0: k`

  `$a1: x`

  `$a2: y`

- **return value:**
  - `$v0`: $k - x^2 + y^2$

- **guarantee:**
  - The values of k, x and y are such that all the intermediate values and the result to be returned fit into 32-bit integers.

Your implementation of `hip` must follow all the MIPS calling conventions for saving/restoring registers.

```
.text
hip:
mul $t1, $a1, $a1   # $t1 <-- x*x
mul $t2, $a2, $a2   # $t2 <-- y*y
sub $t3, $a0, $t1   # $t3 <-- k - x*x
add $v0, $t3, $t2   # f(x,y) <-- k = x*x + y*y
jr  $ra
```

Figure 1: A solution for `hip`.

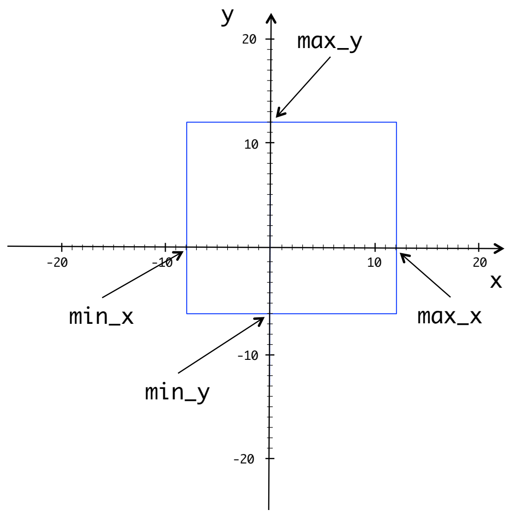

Figure 2: Illustration of the definition of a region in the XY plan.

Question 5 (30 points): In this question you will write MIPS assembly code for `minhip`, a subroutine that evaluates the function defined by equation 1 in every integer point in an specified region of the $x \times y$ plan and returns the minimum value encountered in that region. The definition of a region is illustrated in Figure 1. The `minhip` function will vary x in the interval `[min_x, max_x]` and will vary y in the interval `[min_y, max_y]`. The `[ ]` indicates that the ends of the interval are also included. Both x and y are integer variables that vary in increments of one. The `minhip` function will invoke the `hip` function to evaluate the value of $f(x, y)$ for every combination of integer values of x and y within the specified region.

- **parameters:**

  `$a0`: `min_x`

  `$a1`: `max_x`

  `$a2`: `min_y`

  `$a3`: `max_y`

  memory location with label `const_k`: contains the value of constant `k`

- **return value:**

  `$v0`: minimum value of $f(x, y)$

- **guarantee**

  - the value of the parameters is such that all intermediate and final results fit within 32-bit integers

```
const_k: .word 0
# minhip returns the minimum value of the hiperbola within an
# area defined in the plan
# parameters:   $a0: min_x
#               $a1: max_x
#               $a2: min_y
#               $a3: max_y
#               Memory const_ k contains the value of the constant k.
# return value:
#               $v0: minimum value of f(x,y) = k - x*x + y*y
# guarantee: the value of the parameters is such that all intermediate
#            and final results fit within 32-bit integers
# register usage:  $s0: min
#                  $s1: max_x
#                  $s2: min_y
#                  $s3: max_y
#                  $s4: x
#                  $s5: y
.text
minhip:
        addi $sp, $sp, -28
        sw   $ra, 0($sp)
        sw   $s0, 4($sp)
        sw   $s1, 8($sp)
        sw   $s2, 12($sp)
        sw   $s3, 16($sp)
        sw   $s4, 20($sp)
        sw   $s5, 24($sp)
        lui  $s0, 0x7FFF
        ori  $s0, 0xFFFF       # min <- 0x7FFF FFFF  (max int)
        add  $s1, $0, $a1      # $s1 <- max_x
        add  $s2, $0, $a2      # $s2 <- min_y
        add  $s3, $0, $a3      # $s3 <- max_y
        add  $s4, $0, $a0      # x <- min_x
loop_x: bgt  $s4, $s1, end_x   # if x > max_x
        add  $s5, $0, $s2      # y <- min_y
loop_y: bgt  $s5, $s3, end_y   # if y > max_y
        la   $t1, const_k      # $t1 <- address of k
        lw   $a0, 0($t1)       # $a0 <- k
        add  $a1, $0, $s4      # a1 <- x
        add  $a2, $0, $s5      # a2 <- y
        jal  hip
        addi $s5, $s5, 1      # y <- y+1
        bge  $v0, $s0, loop_y
        add  $s0, $0, $v0     # min <- $v0
        j    loop_y
end_y:  addi  $s4, $s4, 1     # x <- x+1
        j    loop_x
end_x:  add  $v0, $0, $s0
        lw   $ra, 0($sp)
        lw   $s0, 4($sp)
        lw   $s1, 8($sp)
        lw   $s2, 12($sp)
        lw   $s3, 16($sp)
        lw   $s4, 20($sp)
        lw   $s5, 24($sp)
        addi $sp, $sp, 28
        jr   $ra
```


Figure 3: A solution to the `minhip` function.