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.

Code for `hip`

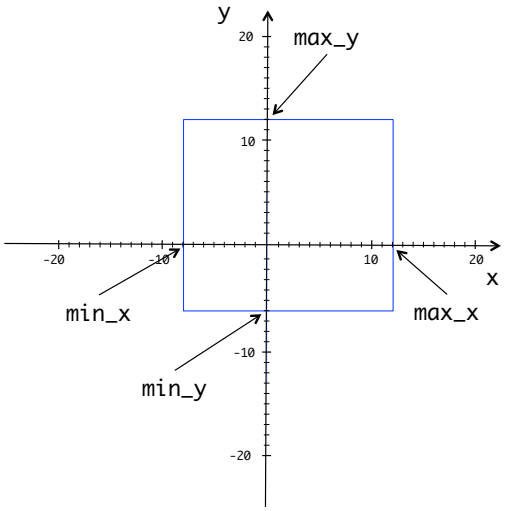

Figure 1: Illustration of the definition of a region in the XY plan.

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 2. 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

MIPS code for `binhip`