* m := 1e9 + 7
* Define a function add(), this will take a, b,
* return ((a mod m) + (b mod m)) mod m
* Define a function sub(), this will take a, b,
* return ((a mod m) - (b mod m) + m) mod m
* Define a function mul(), this will take a, b,
* return ((a mod m) \* (b mod m)) mod m
* Define an array power
* Define a function ok(), this will take x, s,
* if x is same as 0, then −
  + return empty string
* Define one map called hash
* current := 0
* for initialize i := 0, when i < x, update (increase i by 1), do −
  + current := add(mul(current, 26), s[i] - 'a')
* hash[current] := Define an array (1, 0)
* n := size of s
* for initialize i := x, when i < n, update (increase i by 1), do −
  + current := sub(current, mul(power[x - 1], s[i - x] - 'a'))
  + current := add(mul(current, 26), s[i] - 'a')
  + if count is member of hash, then −
    - for all it in hash[current] −
      * if substring of s from it to x - 1 is same as substring of s from i - x + 1 to x - 1, then −
        + return substring of s from it to x - 1
  + Otherwise
    - insert i - x + 1 at the end of hash[current]
* return empty string
* From the main method, do the following −
* ret := empty string
* n := size of S
* power := Define an array of size n and fill this with 1
* for initialize i := 1, when i < n, update (increase i by 1), do −
  + power[i] := mul(power[i - 1], 26)
* low := 0, high := n - 1
* while low <= high, do −
  + mid := low + (high - low) /2
  + temp := ok(mid, S)
  + if size of temp is same as 0, then −
    - high := mid - 1
  + Otherwise
    - if size of temp > size of ret, then −
      * ret := temp
    - low := mid + 1
* return ret