```
#
# Instructions:
#
# A string is a pangram if it includes all letters of the alphabet.
#
# * Determine if var.some_string is a pangram of local.alphabet
# * local.actual should equal the sorted set of letters NOT in var.some_string
# * local.pangram should equal true if var.some_string is a pangram
#
variable "some_string" {
  type = string
  default = "the quick brown fox jumps over lazy dog" # <- is a pangram, try others
}

locals {
  alphabet = "abcdefghijklmnopqrstuvwxyz"
}

################################################################################
#
# Your work goes here in one or more local blocks.
#
# * Try not to spend more than hour on it
# * Have fun
#
locals {

}
################################################################################

#
# To test:
#
# terraform init
# terraform apply -auto-approve
#
# Example output on success:
#
#   > terraform apply -auto-approve
#
#   Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
#
#   Outputs:
#
#   actual = tolist([])
#   pangram = true
#
#
# Example output on failure:
#
```

```
#   tf apply -auto-approve -var some_string=abde
#
#   Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
#
#   Outputs:
#
#   actual = tolist([
#     "c",
#     "f",
#     "g",
#     "h",
#     "i",
#     "j",
#     "k",
#     "l",
#     "m",
#     "n",
#     "o",
#     "p",
#     "q",
#     "r",
#     "s",
#     "t",
#     "u",
#     "v",
#     "w",
#     "x",
#     "y",
#     "z",
#   ])
#   pangram = false
#
locals {
  pangram = length(local.actual) == 0
}

output "actual" {
  value = local.actual
}

output "pangram" {
  value = local.pangram
}
```