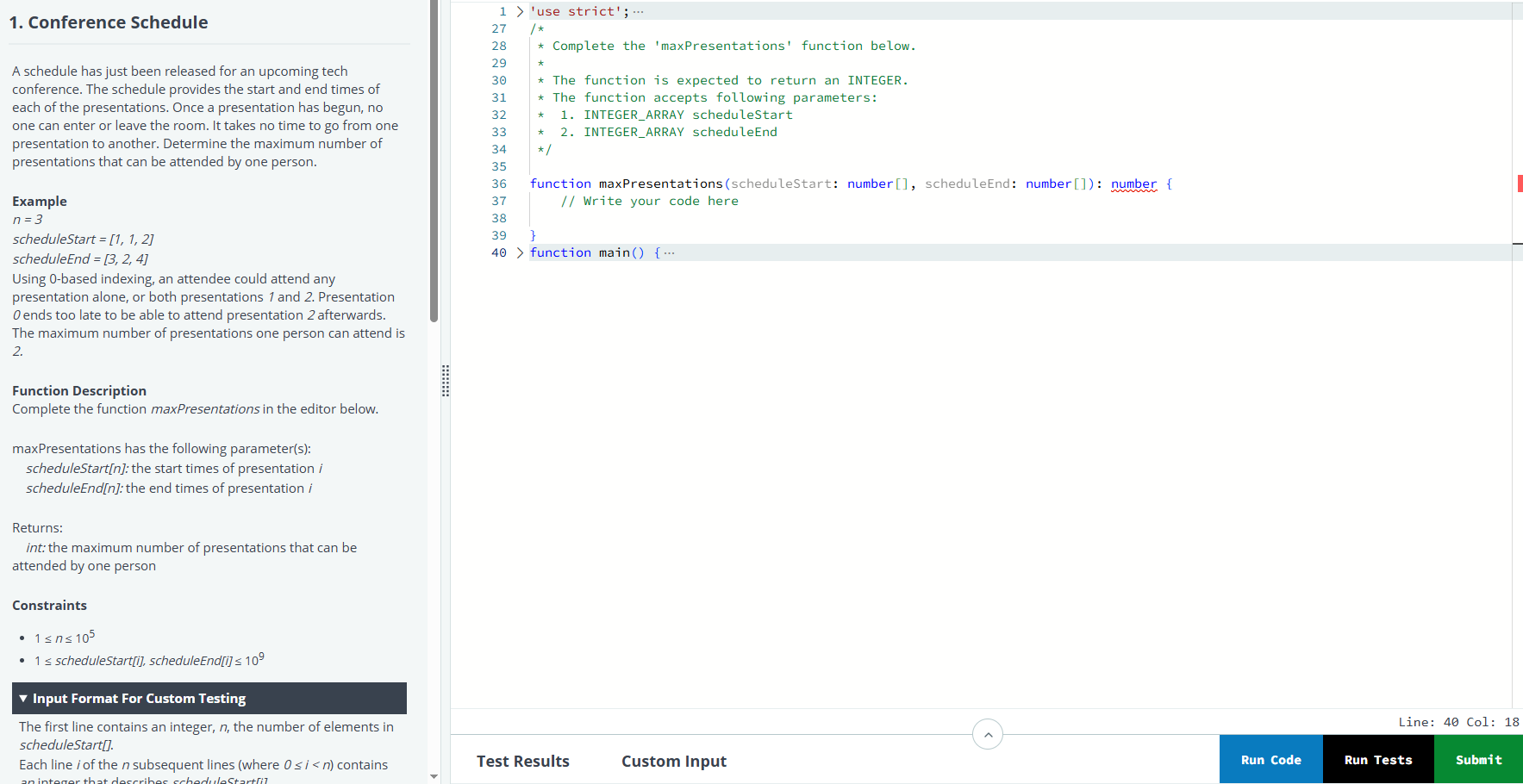
# Conference Schedule



function maxPresentations(scheduleStart: number[], scheduleEnd: number[]): number {

const n = scheduleStart.length;

// Pair up start and end times

const presentations = scheduleStart.map((start, i) => ({

start,

end: scheduleEnd[i]

}));

// Sort by end time

presentations.sort((a, b) => a.end - b.end);

let count = 0;

let lastEndTime = 0;

for (const presentation of presentations) {

if (presentation.start >= lastEndTime) {

count++;

lastEndTime = presentation.end;

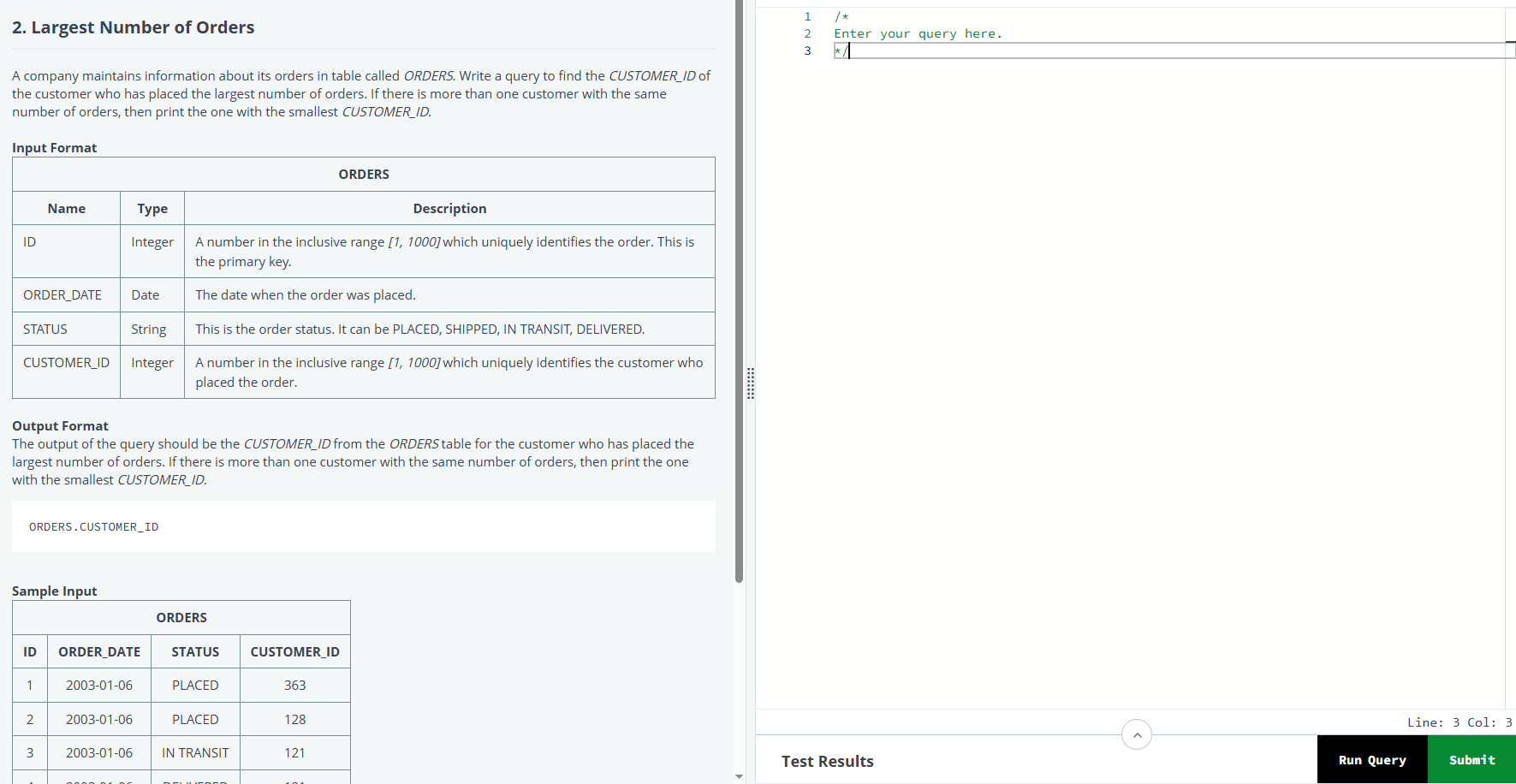
}

}

return count;

}

# Largest Number of Orders



SELECT CUSTOMER\_ID

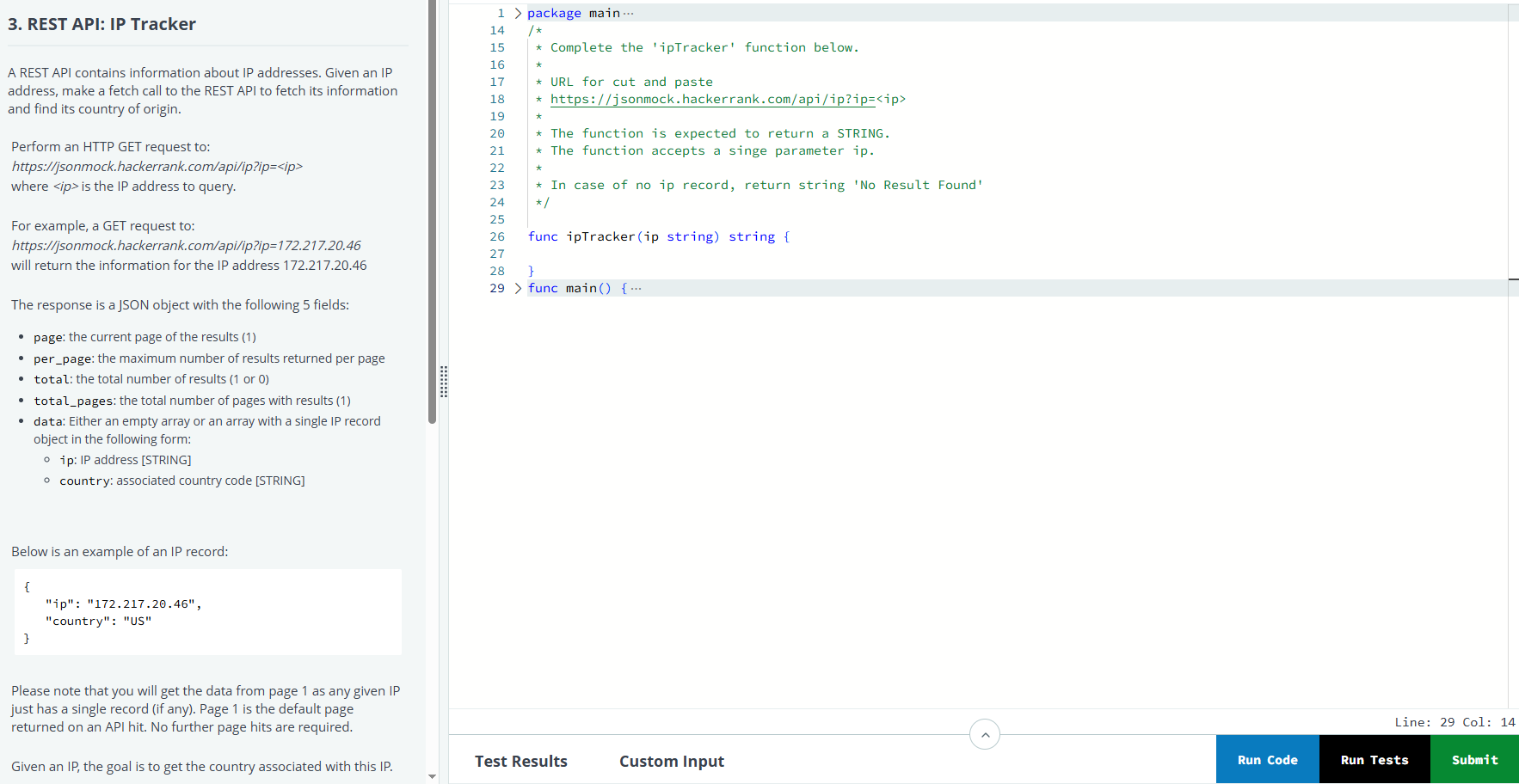
FROM ORDERS

GROUP BY CUSTOMER\_ID

ORDER BY COUNT(\*) DESC, CUSTOMER\_ID ASC

LIMIT 1;

# REST API: IP Tracker



package main

import (

"bufio"

"encoding/json"

"fmt"

"io"

"net/http"

"os"

"strings"

)

type IPResponse struct {

Page int `json:"page"`

PerPage int `json:"per\_page"`

Total int `json:"total"`

TotalPages int `json:"total\_pages"`

Data []struct {

IP string `json:"ip"`

Country string `json:"country"`

} `json:"data"`

}

func ipTracker(ip string) string {

url := fmt.Sprintf("https://jsonmock.hackerrank.com/api/ip?ip=%s", ip)

resp, err := http.Get(url)

if err != nil {

return "No Result Found"

}

defer resp.Body.Close()

var ipResponse IPResponse

if err := json.NewDecoder(resp.Body).Decode(&ipResponse); err != nil {

return "No Result Found"

}

if ipResponse.Total == 0 || len(ipResponse.Data) == 0 {

return "No Result Found"

}

return ipResponse.Data[0].Country

}

func main() {

reader := bufio.NewReaderSize(os.Stdin, 16\*1024\*1024)

stdout, err := os.Create(os.Getenv("OUTPUT\_PATH"))

checkError(err)

defer stdout.Close()

writer := bufio.NewWriterSize(stdout, 16\*1024\*1024)

ip := readLine(reader)

result := ipTracker(ip)

fmt.Fprintf(writer, "%s\n", result)

writer.Flush()

}

func readLine(reader \*bufio.Reader) string {

str, \_, err := reader.ReadLine()

if err == io.EOF {

return ""

}

return strings.TrimRight(string(str), "\r\n")

}

func checkError(err error) {

if err != nil {

panic(err)

}

}