محمد بهرامی - 400243024

گزارش تمرین اول کامپیوتری شبکه های کامپیوتری

فایل server :

```
from socket import *
from hashlib import sha256
from random import randint
from pprint import pprint
```

همانند کتاب از کتابخانه سوکت استفاده کرده تا تابع های مورد نیاز برنامه نویسی سوکت را داشته باشیم.
از کتابخانه hashlib الگوریتم sha256 را انتخاب کردم چون در گذشته با آن کار کرده بوده و آشنا بودم.
برای تولید عدد رندوم از تابع randint کتابخانه رندوم استفاده کردم.
برای نمایش بهتر لاگ ها از تابع pprint استفاده کردم.

```
ip = '127.0.0.1'
port = 8090
server = socket(AF_INET, SOCK_STREAM)
server.bind((ip, port))
server.listen(1)
pprint(f'listening on: {ip}:{port}')
```

مقدار ip که لوکال هاست و port را بر اساس فایل تمرین ست کرده و پس از بایند کردن و listen کردن یک log برای مشخصات سرور میزنیم.
سرور همیشه بالا میماند و کلاینت ها میتوانند به آن یکی یکی درخواست بدهند:

```
while True:
    connection, address = server.accept()
    authentication = connection.recv(1024).decode()
    pprint(f'get authentication message from {address}')
    random_number = randint(0, 31)
    pprint(f'server number: {random_number}')
    answer = ip + str(random_number)
    answer_hash = sha256(answer.encode()).hexdigest()
    cannot_guess = True
    while cannot_guess:
        connection.send('Send Hash!'.encode())
        client_guess = connection.recv(1024).decode()
        pprint(f'guess: {client_guess}  || answer: {answer_hash}')
        if answer_hash == client_guess:
            cannot_guess = False
    connection.send('Hash Found'.encode())
    connection.close()
    pprint('connection closed!')
    print()
```

در یک حلقه بی نهایت برای اینکه سرور بالا بماند تا بتوانیم چند درخواست بدیم کد را قرار میدهیم.
ابتدا پیام authentication کلاینت را دریافت کرده و بعد از آن عددی بین 0 تا 31 رندوم تولید کرده و دو لاگ برای ثبت پورت سوکت کلاینت و عددی که سرور در نظر گرفته برای دیباگ میزنیم.
عدد را به ip اضافه کرده و هش میکنیم و در answer_hash نگه میداریم.
سپس در یک حلقه بی نهایت دیگر تا زمانی که کاربر حدس نزده باشد برای او پیام ارسال هش را ارسال کرده و از او هش دریافت میکنیم و با مقدار اصلی مقایسه میکنیم و اگر برابر بود از حلقه خارج شده پیام موفقیت را ارسال میکنیم و یک لاگ برای اینکه متوجه بشیم این کلاینت دیگه قطع شده ثبت میکنیم.

کد کلاینت:

کتابخانه ها یکسان می‌باشند.

```
ip = '127.0.0.1'
port = 8090
client = socket(AF_INET, SOCK_STREAM)
client.connect((ip, port))
pprint(f'connected to {ip}:{port}')
client.send('Auth'.encode())
order = client.recv(1024).decode()
final = 'Failure'
```

مقادیر id و port سرور را به تابع connect داده تا به سرور وصل شود و پس از وصل شدن یک لاگ ثبت میکنیم.

سپس پیام authentication را ارسال میکنیم و منتظر دستور سرور میشویم

مقدار final رشته ای است که در نهایت چاپ میکنیم و وقتی حدس درست زده میشود آن را به Hash Found تغییر میدهیم. برای این مقدار اولیه شکست را به آن داده ایم تا بتوانیم دیباگ کنیم اگر هیچوقت پیدا نکرد.

```
for i in range(32):
    guess = ip + str(i)
    guess_hash = sha256(guess.encode()).hexdigest()
    client.send(guess_hash.encode())
    result = client.recv(1024).decode()
    pprint(f'attempt {i} || hash: {guess_hash} || result: {result}')
    if result == 'Hash Found':
        client.close()
        final = result
        break
pprint(f'result: {final}')
```

از صفر تا 31 را امتحان کرده (به انتهای ip اضافه میکنیم و هش میکنیم و با استفاده از send ارسال میکنیم). بعد از دریافت جواب سرور یک لاگ برای دیباگ زده و سپس بررسی میکنیم اگر برابر با پیام پیدا شدن بود مقدار final را به Hash Found تغییر داده و از حلقه خارج میشویم.
در انتها پیام موفقیت را چاپ میکنیم.(اگر کد مشکل داشته باشه و بعد از32 تا عددی که چک کرده پیدا نکرده باشه، Failure چاپ میشه که بشه دیباگ کرد)