Dear applicant,

Thank you for your application and your interest in working for Exasol.

We would like to invite you to take the first step of our application process. At this point you can convince us of your problem solving ability. Starting from now you have 14 days to complete the task described bellow.

Before we explain the test in detail, you can find further information in our FAQs:

1) What happens after completing the task?

   After successful processing, we will evaluate your test results. We will come back to you with feedback.

2) Who can I contact if I have technical problems or questions about the test?

   Please contact one our research & development employees:

   Oleksandr Kozachuk <oleksandr.kozachuk@exasol.com>
   Ruslan Rusinov <ruslan.rusinov@exasol.com>
   Igor Chubin <igor.chubin@exasol.com>

   They are available Monday to Friday from 10:00 to 17:00 CET time.

3) If I only have an old/slow computer, is it possible to solve the test using it?

   The test is designed to be solvable on difficulty 9 with a regular Raspberry PI and the regular Python language. The solution program usually runs for more than an hour with difficulty 9, although highly optimized code may need only several minutes or even only seconds on regular PC hardware. Some tips for the solution:

   - Use a programming language and environment, with which you are experienced enough to write efficient code.
   - Test your code carefully before running it against our server.
   - Use profilers to find bottlenecks and optimize your code.
   - Think about the efficiency of your random string generator.
   - Think about the probabilities of finding a correct solution in different situations. Try to maximize the probability.
   - Additional hardware like GPUs can speedup the calculation.

In the following you will find the test instructions. Good luck!

As part of our application process, we would like you to write a test program that we could later discuss in your interview. The data you send using the program will be used for further communication with you. You can write your program in any programming language you prefer, but you should be able to show and explain your solution later in the interview. To connect to the server, you need the keys included in this README.

The following pseudocode represents the program you need to write. It is a full implementation with all required elements. It is written with Python language syntax and semantics in mind, but it is not a correct implementation and needs to be extended to actually run in a Python interpreter. The main purpose of this pseudocode is to give you an idea of what you need to develop.

```
# === BEGIN ===
conn = tls_connect("18.202.148.130:3336", cert, key)
authdata = ""
while true:
    args = conn.read().strip().split(' ')
    if args[0] == "HELO":
        conn.write("EHLO\n")
    elif args[0] == "ERROR":
        print("ERROR: " + " ".join(args[1:]))
        break
    elif args[0] == "POW":
        authdata, difficulty = args[1], args[2]
        while true:
            # generate short random string, server accepts all utf-8 characters,
            # except [\n\r\t ], it means that the suffix should not contain the
            # characters: newline, carriege return, tab and space
            suffix = random_string()
            cksum_in_hex = SHA1(authdata + suffix)
            # check if the checksum has enough leading zeros
            # (length of leading zeros should be equal to the difficulty)
            if cksum_in_hex.startswith("0"*difficulty):
                conn.write(suffix + "\n")
                break
    elif args[0] == "END":
        # if you get this command, then your data was submitted
        conn.write("OK\n")
        break
    # the rest of the data server requests are required to identify you
    # and get basic contact information
    elif args[0] == "NAME":
        # as the response to the NAME request you should send your full name
        # including first and last name separated by single space
        conn.write(SHA1(authdata + args[1]) + " " + "My name\n")
    elif args[0] == "MAILNUM":
        # here you specify, how many email addresses you want to send
        # each email is asked separately up to the number specified in MAILNUM
        conn.write(SHA1(authdata + args[1]) + " " + "2\n")
    elif args[0] == "MAIL1":
        conn.write(SHA1(authdata + args[1]) + " " + "my.name@example.com\n")
    elif args[0] == "MAIL2":
        conn.write(SHA1(authdata + args[1]) + " " + "my.name2@example.com\n")
    elif args[0] == "SKYPE":
        # here please specify your Skype account for the interview, or N/A
        # in case you have no Skype account
        conn.write(SHA1(authdata + args[1]) + " " + "my.name@example.com\n")
    elif args[0] == "BIRTHDATE":
        # here please specify your birthdate in the format %d.%m.%Y
        conn.write(SHA1(authdata + args[1]) + " " + "01.02.2017\n")
    elif args[0] == "COUNTRY":
        # country where you currently live and where the specified address is
        # please use only the names from this web site:
        #   https://www.countries-ofthe-world.com/all-countries.html
        conn.write(SHA1(authdata + args[1]) + " " + "Germany\n")
    elif args[0] == "ADDRNUM":
        # specifies how many lines your address has, this address should
        # be in the specified country
        conn.write(SHA1(authdata + args[1]) + " " + "2\n")
    elif args[0] == "ADDRLINE1":
        conn.write(SHA1(authdata + args[1]) + " " + "Long street 3\n")
    elif args[0] == "ADDRLINE2":
        conn.write(SHA1(authdata + args[1]) + " " + "32345 Big city\n")
conn.close()
# === END ===
```

Notes:
- This pseudocode is written with Python semantics in mind
- Only TLS connections with valid keys are allowed
- All communication needs to be in valid UTF-8
- Protocol is line oriented and each line should end with \n
- Only if you see the END request from the server is the data fully sent
- On problems, the server sends the ERROR command and closes the connection
- If the data is not fully sent, no data will be recorded on server
- There are no logs about connections on the server side
- HELO and POW commands always come first (handshake)
- END command is always the last command and confirms successful application
- Other commands come from the server in random order
- List of acceptable country names:
  https://www.countries-ofthe-world.com/all-countries.html
- The timeout of the POW command is 2 hours
- All other commands have a timeout of 6 seconds
- It is possible to reach this service on the following ports:
  3336, 8083, 8446, 49155, 3481, 65532

Tue Nov 30 14:01:19 UTC 2021