نام و نام خانوادگی: میثم خزاعی صبور **تاریخ: ۱۴۰۳/۰۱/۲۶**

گزارش پروژه:

پیاده سازی کد به زبان سی

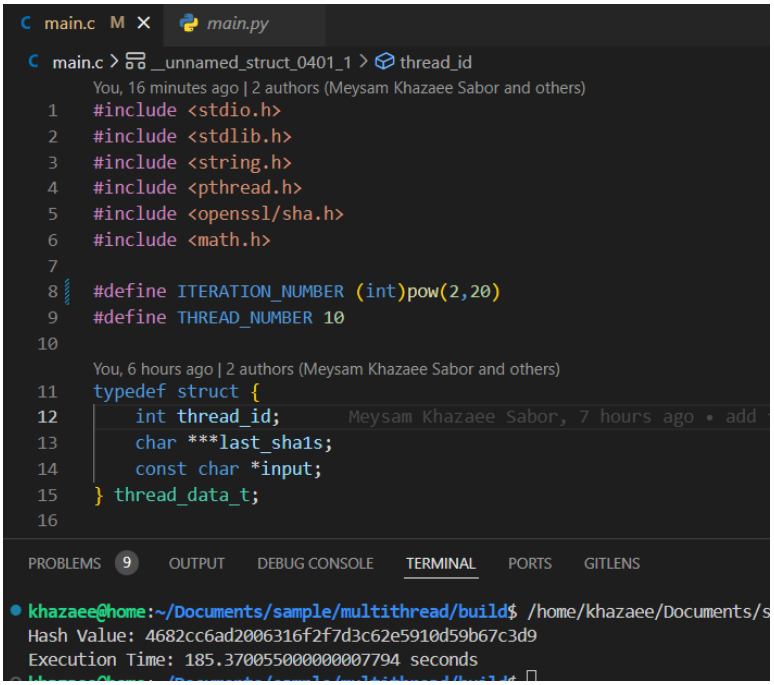
(توجه شود که تعداد thread ها و ورودی thread ها ثابت در نظر گرفته شده است.)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <openssl/sha.h>
#include <math.h>

#define ITERATION_NUMBER (int)pow(2,20)
#define THREAD_NUMBER 10

typedef struct {
    int thread_id;
    char ***last_sha1s;
    const char *input;
} thread_data_t;

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
// pthread_cond_t condition = PTHREAD_COND_INITIALIZER;

void *thread_function(void *arg) {
    thread_data_t *data = (thread_data_t *)arg;
    int group_mate = data->thread_id ^ 1;  // XOR with 1 for finding group
mate
    for (int counter = 0; counter < ITERATION_NUMBER; counter++) {
        char data_str[(sizeof(data->thread_id)/sizeof(int)) +
(SHA_DIGEST_LENGTH * 2 + 1) + (sizeof(counter)/sizeof(int)) + strlen(data-
>input) + 1];

        if (counter == 0) {
            snprintf(data_str, sizeof(data_str), "%d%d%s", data->thread_id,
counter, data->input);
        }
        else if (strlen(data->last_sha1s[group_mate][counter - 1]) != 0) {
            pthread_mutex_lock(&mutex);
            snprintf(data_str, sizeof(data_str), "%d%s%d%s", data->thread_id,
data->last_sha1s[group_mate][counter - 1], counter, data->input);
            pthread_mutex_unlock(&mutex);
        }
        else{
            //thread_cond_wait(&condition, &mutex);
            do {
                //printf(".");
            } while(strlen(data->last_sha1s[group_mate][counter - 1]) == 0);
            pthread_mutex_lock(&mutex);
            snprintf(data_str, sizeof(data_str), "%d%s%d%s", data->thread_id,
data->last_sha1s[group_mate][counter - 1], counter, data->input);
            pthread_mutex_unlock(&mutex);
        }

        unsigned char sha1[SHA_DIGEST_LENGTH + 1];
        SHA1((unsigned char *)data_str, strlen(data_str), sha1);

        char hex_sha1[SHA_DIGEST_LENGTH * 2 + 1];
```

```
        for (int i = 0; i < SHA_DIGEST_LENGTH; i++) {
            snprintf(hex_sha1 + i * 2, 3, "%02x", sha1[i]);
        }

        strcpy(data->last_sha1s[data->thread_id][counter], hex_sha1);
    }
    pthread_mutex_unlock(&mutex);
    pthread_exit(NULL);
}

char *new_hash_function(char *input, int k) {
    pthread_t threads[k];
    char*** last_sha1s = (char***)calloc(k,sizeof(char**));

    for (int i = 0; i < k; i++) {
        last_sha1s[i] = (char**)calloc(ITERATION_NUMBER, sizeof(char*));
        for (int j = 0; j < ITERATION_NUMBER; j++) {
            last_sha1s[i][j] = (char *)calloc((SHA_DIGEST_LENGTH * 2 +
1),sizeof(char));
        }
    }

    thread_data_t data[k];
    for (int i = 0; i < k; i++) {
        data[i].thread_id = i;
        data[i].last_sha1s = last_sha1s;
        data[i].input = input;
        pthread_create(&threads[i], NULL, thread_function, &data[i]);
    }

    for (int i = 0; i < k; i++) {
        pthread_join(threads[i], NULL);
    }

    char final_hash[k * SHA_DIGEST_LENGTH * 2 + 1];
    memset(final_hash, 0, sizeof(final_hash));
    for (int i = 0; i < k; i++) {
        strcat(final_hash, last_sha1s[i][ITERATION_NUMBER - 1]);
    }

    unsigned char overall_sha1[SHA_DIGEST_LENGTH + 1];
    SHA1((unsigned char *)final_hash, strlen(final_hash), overall_sha1);

    char hex_overall_sha1[SHA_DIGEST_LENGTH * 2 + 1];
    for (int i = 0; i < SHA_DIGEST_LENGTH; i++) {
        snprintf(hex_overall_sha1 + i * 2, 3, "%02x", overall_sha1[i]);
    }

    for (int i = 0; i < k; i++) {
        for (int j = 0; j < ITERATION_NUMBER; j++) {
            if (NULL != last_sha1s[i][j]){
                free(last_sha1s[i][j]);
            }
        }
        free(last_sha1s[i]);
    }
    free(last_sha1s);

    return strdup(hex_overall_sha1);
}

int main() {
    double start_time = clock();
```

```c
    char *hash_result = new_hash_function("meysam_khazaee", THREAD_NUMBER);

    double end_time = clock();
    double execution_time = (end_time - start_time) / CLOCKS_PER_SEC;

    printf("Hash Value: %s\n", hash_result);
    printf("Execution Time: %.18f seconds\n", execution_time);

    free(hash_result);
    return 0;
}
```

خروجی اجرای کد در زبان سی



پیاده سازی کد به زبان پایتون

(توجه شود که تعداد thread ها و ورودی thread ها ثابت در نظر گرفته شده است.)

```python
import threading
import hashlib
import time

itratetion_number = 2**20
thread_number = 10

def new_hash_function(input, k):
    threads = []
    last_sha1s = [["" for element in range(itratetion_number)] for element in
range(k)]
    mutex = threading.Lock()
```

```
    def thread_function(thread_id):
        mutex.acquire()
        nonlocal last_sha1s  # Access last_sha1s in the enclosing scope
        group_mate = thread_id ^ 1  # XOR with 1 to get the group mate
        for counter in range(itratetion_number):
            if counter == 0 :
                data = f"{thread_id}{counter}{input}"
            else:
                while(last_sha1s[group_mate][counter-1] == ""):
                    pass
                data = f"{thread_id}{last_sha1s[group_mate][counter-
1]}{counter}{input}"
            sha1 = hashlib.sha1(data.encode()).hexdigest()
            last_sha1s[thread_id][counter] = sha1
        mutex.release()


    # Create threads and start their execution
    for i in range(k):
        thread = threading.Thread(target=thread_function, args=(i,))
        threads.append(thread)
        thread.start()

    # Wait for all threads to finish
    for thread in threads:
        thread.join()

    # Collect final SHA1s from each thread
    final_hash = []
    for row in last_sha1s:
    # Append the last element to the row (modifies original matrix)
        final_hash.append(row[-1])
    final_sha1s = "".join(final_hash)

    # Calculate the overall SHA1
    overall_sha1 = hashlib.sha1(final_sha1s.encode()).hexdigest()

    return overall_sha1

def main():
    start_time = time.time()
    hash_result = new_hash_function("meysam_khazaee", thread_number)
    end_time = time.time()
    execution_time = end_time - start_time
    print("Hash Value:", hash_result)
    print("Execution Time:", execution_time, "seconds")

if __name__ == "__main__":
    main()
```

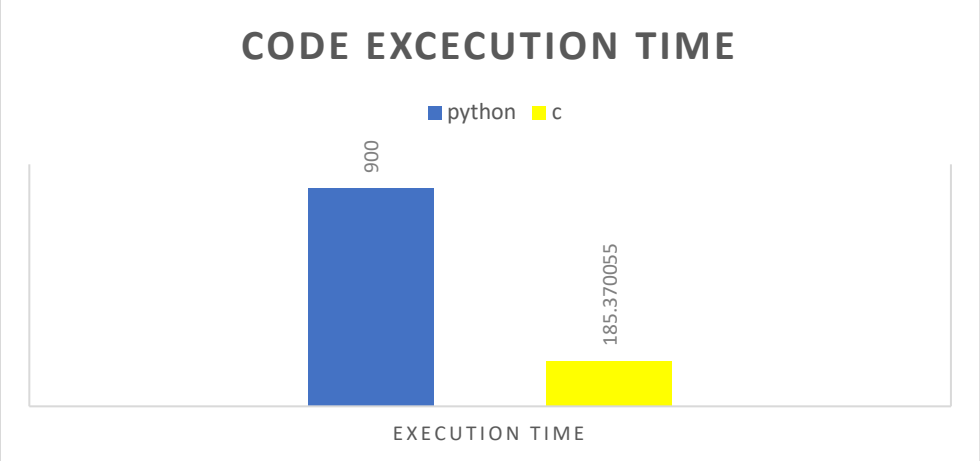
خروجی اجرای کد در زبان پایتون

```python
itratetion_number = 2**20
thread_number = 10

def new_hash_function(input, k):
    threads = []
    last_sha1s = [["" for element in range(itratetion_num
    mutex = threading.Lock()

    def thread_function(thread_id):
        nonlocal last_sha1s  # Access last_sha1s in the e
        group_mate = thread_id ^ 1  # XOR with 1 to get t
        for counter in range(itratetion_number):
            print("counter = {}".format(counter))
            if counter == 0 :
                data = f"{thread_id}{counter}{input}"
            elif len(last_sha1s[group_mate][counter - 1])
```

PROBLEMS 9 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
counter = 979
counter = 980
counter = 969
counter = 1004
counter = 1003
counter = 1004
counter = 1005
counter = 981
counter = 968
counter = 969
counter = 980
counter = 981
counter = 982
```

مقایسه زمان اجرا



زمان اجرای کد پایتون خیلی زیاد بود و عدد تقریبی است.(برای زمان دقیق می بایست اسکریپت پایتونی مدت زمان زیادی اجرا شود که بنده به خاطر ضیق وقت صرف نظر کردم)

۳- آیا این برنامه را می توان به صورت microservice پیاده سازی کرد؟ چه مزایایی دارد؟ روی زمان اجرا چه اثری دارد؟ لطفا پاسختان را به صورت تشریحی و به طور مفصل بیان کنید. در صورت تمایل خود تان، این برنامه را به صورت microservice پیاده سازی وزمان اجرای آن را با حالت های قبلی مقایسه کنید. (پیاده سازی، امتیاز مثبت در نظر گرفته می شود)

کد پیاده سازی شده یک هش SHA-1 را محاسبه می‌کند و مستقیماً چندین ورودی مستقل را پردازش نمی‌کند، اگر کد را بازسازی کنیم تا به عنوان پایه‌ای برای یک میکروسرویس با تغییرات احتمالی می بایست کد را به‌گونه‌ای بازسازی کنیم که:

- ورودی‌های چندگانه: برای پردازش چندین ورودی مستقل، کد را طوری تنظیم کرد که آرایه‌ای از رشته‌ها را به عنوان ورودی بپذیرد یا به مشتریان اجازه دهید رشته‌های جداگانه را از طریق یک API ارسال کنند.
- هش(های) محاسبه شده را به عنوان ساختمان داده برگرداند.

<< پایان گزارش >>