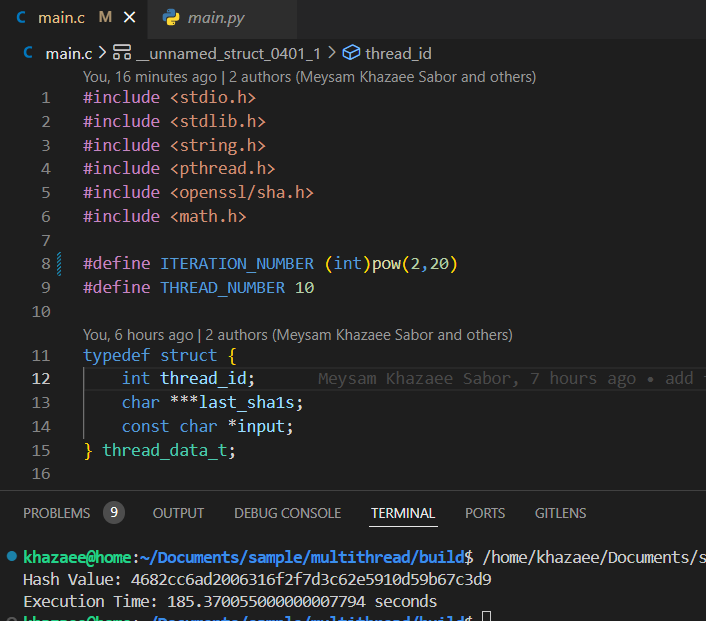
**نام و نام خانوادگی: میثم خزاعی صبور تاریخ: 26/01/1403**

**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**();**    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.)*

**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

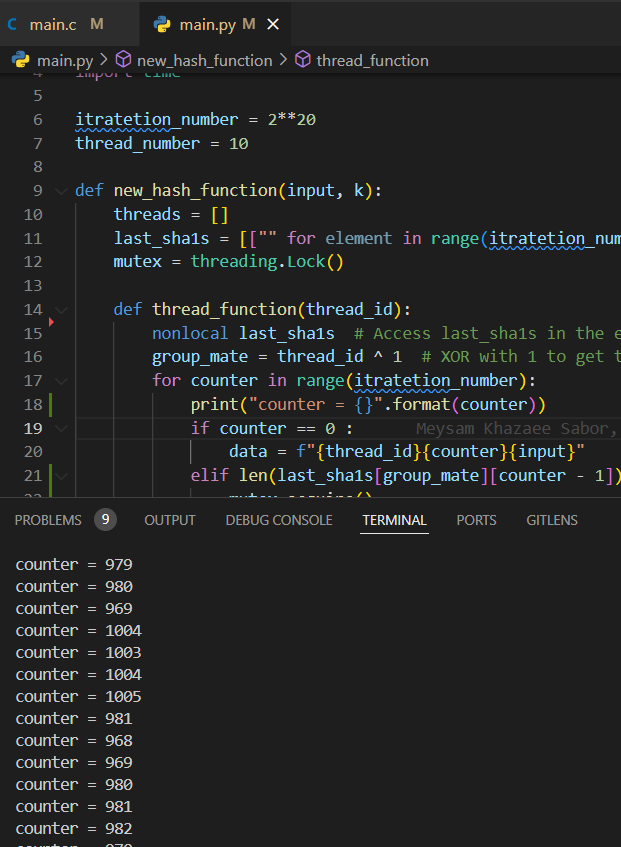
**print(**"Hash Value:"**,** hash\_result**)**

**print(**"Execution Time:"**,** execution\_time**,** "seconds"**)**

**if** \_\_name\_\_ **==** "\_\_main\_\_"**:**

main**()**

خروجی اجرای کد در زبان پایتون



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)*

**٣- آیا این برنامه را مى توان به صورت microservice پياده سازی كرد؟ 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 پياده سازى وزمان اجراى آن را با حالت هاى قبلى مقايسه كنيد. (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<<**