SARDAR PATEL INSTITUTE OF TECHNOLOGY

| | |
|---|---|
| NAME | Karthik Iyer |
| UID | 2021700028 |
| CLASS | SY BTECH CSE-DS B BATCH |
| EXP NO | 4 |

## AIM: Multithreading

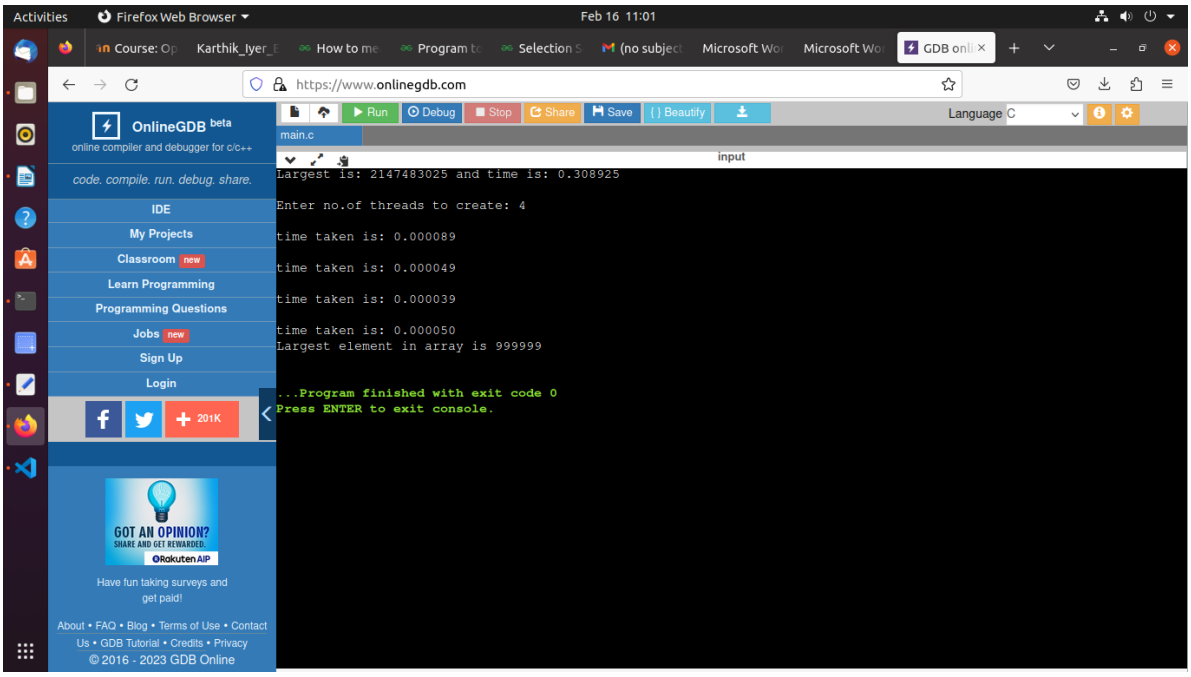
Read the array of size 10,000,000 . Write the function findlargest(). Compare the time required to perform the given operation using various numbers of threads. Create the table mentioning no of threads and time required. Write down the conclusion.

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

#define ARRAY_SIZE 10000000
// #define NUM_THREADS 4
 int
array[ARRAY_SIZE];
 struct thread_data
{     int thread_id;
int start;     int
end;
};  int largest
= 0;
 void *find_largest(void *thread_arg) {
struct thread_data *my_data;     my_data =
(struct thread_data *) thread_arg;
        int start = my_data-
>start;     int end = my_data-
>end;     int i;
    int local_largest = 0;

    for (i = start; i < end; i++) {
if (array[i] > local_largest) {
local_largest = array[i];
        }
    }         if (local_largest >
largest) {         largest =
local_largest;
    }
    pthread_exit(NULL);
} int findLargest(int arr[],int
n) {
```

```
    int lower=1, upper = n, count=n;     for(int
i=0;i<n;i++)
    {         int num=(rand());
arr[i]=num;
    }
    int largest=arr[0];     for(int
i=0;i<n;i++)
    {         if(arr[i]>largest)
largest=arr[i];
    }     return largest;
} int main() {
     int *tp=(int *)malloc(sizeof(int)*10000000);     int
lower=1, upper = 10000000, count=10000000;
     clock_t start, end;     start = clock();     int
ans=findLargest(tp,count);     end = clock();     double duration =
((double)end - start)/CLOCKS_PER_SEC;
     printf("Largest is: %d and time is: %lf\n",ans,duration);


    int i, NUM_THREADS;
    printf("\nEnter no.of threads to create: ");
scanf("%d",&NUM_THREADS);     pthread_t
threads[NUM_THREADS];
    struct thread_data thread_args[NUM_THREADS];

    for (i = 0; i < ARRAY_SIZE; i++) {         array[i] =
rand() % 1000000;
    }
    for (i = 0; i < NUM_THREADS; i++) {         thread_args[i].thread_id =
i;         thread_args[i].start = i * (ARRAY_SIZE / NUM_THREADS);
thread_args[i].end = (i + 1) * (ARRAY_SIZE / NUM_THREADS);
         clock_t start, end;         start = clock();
pthread_create(&threads[i], NULL, find_largest, (void *)
&thread_args[i]);         end =
clock();
```

```
        double duration = ((double)end - start)/CLOCKS_PER_SEC;
printf("\ntime taken is: %f\n", duration);
pthread_create(&threads[i], NULL, find_largest, (void *)
&thread_args[i]);
    }         for (i = 0; i <
NUM_THREADS; i++) {
pthread_join(threads[i], NULL);
    }

    // printf("Largest element in array is %d\n", largest);

    return 0;
}
```

OUTPUT:




CONCLUSION:

When thread is not used, 1000000 elements are calculated in one iteration.

Therefore it takes greater amount of time to complete.

When threads are used, the input size drastically decreases and the program runs concurrently

Therefore the time decreases for the reduced input size.