HOMEWORK ASSIGNMENT 4

Date-17-10-25

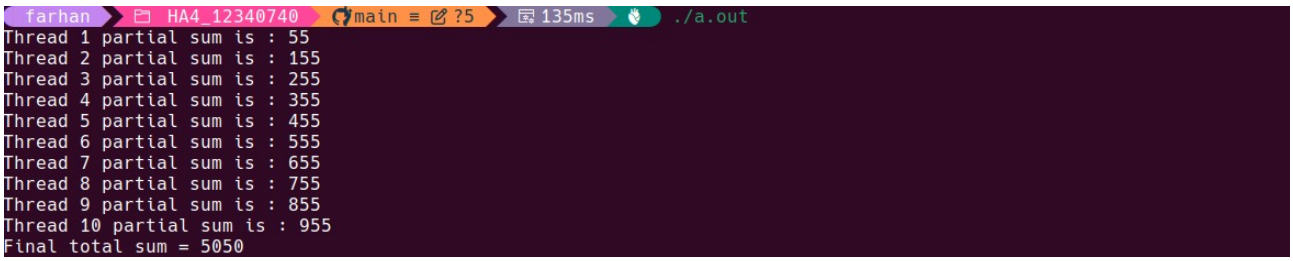
Question 1)

The code is as follows:-

HA4_Q1_12340740.c > main()

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#define ARRAY_SIZE 100
#define SEGMENTS 10
#define SEGMENT_SIZE (ARRAY_SIZE / SEGMENTS)
int arr[ARRAY_SIZE];
typedef struct {
    int* arr;
    int start;
} args;
void* partial_sum(void* arg) {
    args* a = (args*)arg;
    int sum = 0;
    for (int i = a->start; i < a->start + SEGMENT_SIZE; ++i) {
        sum += a->arr[i];
    }
    int* result = (int*)malloc(sizeof(int));
    if (result == NULL) {
        return NULL;
    }
    *result = sum;
    return (void*)result;
}
int main(){
    pthread_t threads[SEGMENTS + 1];
    args thread_args[SEGMENTS];
    int ids[SEGMENTS + 1] = {0};
    int total_sum = 0;
    for (int i = 0; i < ARRAY_SIZE; i++) {
        arr[i] = i + 1;
    }
    for (int i = 0; i < SEGMENTS; i++) {
        thread_args[i].arr = arr;
        thread_args[i].start = i * SEGMENT_SIZE;
        pthread_create(&threads[i], NULL, partial_sum, &thread_args[i]);
    }
    for (int i = 0; i < SEGMENTS; i++){
        void* partial_result_ptr;
        pthread_join(threads[i], &partial_result_ptr);
        ids[i] = *((int*)partial_result_ptr);
        total_sum += ids[i];
        free(partial_result_ptr);
    }
    ids[SEGMENTS] = total_sum;
    for(int i = 0; i < SEGMENTS; i++){
        printf("Thread %d partial sum is : %d \n", i + 1, ids[i]);
    }
    printf("Final total sum = %d\n", ids[SEGMENTS]);
    return 0;
}
```

The output is as follows-



The code first creates an array of 100 elements listed from 1 to 100 . Then it divides the array in 10 parts based on the index the index in then passed to the thread which then calculates the sum of that partition now for the final thread the final thread takes the result of all the partial sums and adds it thus giving us the final output .

Question 2)
The code is as follows:-

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#define ARRAY_SIZE 100
#define SEGMENTS 10
#define SEGMENT_SIZE (ARRAY_SIZE / SEGMENTS)
int arr[ARRAY_SIZE];
typedef struct {
    int* arr;
    int start;
} args;
void* partial_sum(void* arg) {
    args* a = (args*)arg;
    int sum = 0;
    for (int i = a->start; i < a->start + SEGMENT_SIZE; ++i) {
        sum += a->arr[i];
    }
    int* result = (int*)malloc(sizeof(int));
    if (result == NULL) {
        return NULL;
    }
    *result = sum;
    return (void*)result;
}
int main(){
    pthread_t threads[SEGMENTS + 1];
    args thread_args[SEGMENTS];
    int ids[SEGMENTS + 1] = {0};
    int total_sum = 0;

    for (int i = 0; i < ARRAY_SIZE; i++) {
        arr[i] = i + 1;
    }
    for (int i = 0; i < SEGMENTS; i++) {
        thread_args[i].arr = arr;
        thread_args[i].start = i * SEGMENT_SIZE;
        pthread_create(&threads[i], NULL, partial_sum, &thread_args[i]);
    }
    for (int i = 0; i < SEGMENTS; i++){
        void* partial_result_ptr;
        pthread_join(threads[i], &partial_result_ptr);
        ids[i] = *((int*)partial_result_ptr);
        total_sum += ids[i];
        free(partial_result_ptr);
    }
    ids[SEGMENTS] = total_sum;
    for(int i = 0; i < SEGMENTS; i++){
        printf("Thread %d partial sum is : %d \n", i + 1, ids[i]);

    }
    printf("Final total sum = %d\n", ids[SEGMENTS]);
    printf("Average = %.2f\n", (float)ids[SEGMENTS]/ (float)ARRAY_SIZE);


    return 0;
}
```

The output is as follows:-

```
Thread 1 partial sum is : 55
Thread 2 partial sum is : 155
Thread 3 partial sum is : 255
Thread 4 partial sum is : 355
Thread 5 partial sum is : 455
Thread 6 partial sum is : 555
Thread 7 partial sum is : 655
Thread 8 partial sum is : 755
Thread 9 partial sum is : 855
Thread 10 partial sum is : 955
Final total sum = 5050
Average = 50.50
```

The only change being adding an average in the last.

Question 3)-

The code is as follows:-

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <time.h>
#include <limits.h>
#define ARRAY_SIZE 100
#define NUM_WORKER_THREADS 10
#define NUM_TOTAL_THREADS (NUM_WORKER_THREADS + 1)
#define SEGMENT_SIZE (ARRAY_SIZE / NUM_WORKER_THREADS)
int numbers[ARRAY_SIZE];
int segment_maxima[NUM_WORKER_THREADS];
typedef struct {
    int thread_id;
    int start_index;
    int end_index;} thread_data_t;
void* find_segment_max(void* arg) {
    thread_data_t* data = (thread_data_t*)arg;
    int local_max = INT_MIN;
    for (int i = data->start_index; i < data->end_index; i++) {
        if (numbers[i] > local_max) {
            local_max = numbers[i];}}
    segment_maxima[data->thread_id - 1] = local_max;
    printf("Thread %2d max: %3d (Segment Indices %3d to %3d)\n",
           data->thread_id,
           local_max,
           data->start_index,
           data->end_index - 1);
    pthread_exit(NULL);}
void* find_overall_max(void* arg) {
    int overall_max = INT_MIN;
    for (int i = 0; i < NUM_WORKER_THREADS; i++) {
        if (segment_maxima[i] > overall_max) {
            overall_max = segment_maxima[i];}}
    return (void*)(long)overall_max;}
int main() {
    pthread_t worker_threads[NUM_WORKER_THREADS];
    pthread_t coordinator_thread;
    thread_data_t worker_data[NUM_WORKER_THREADS];
    srand(time(NULL));
    printf("--- Initializing Array with Random Values (1 to 100) ---\n");
    for (int i = 0; i < ARRAY_SIZE; i++) {
        numbers[i] = (rand() % 100) + 1;}
    printf("Array initialized. Total size: %d elements.\n\n", ARRAY_SIZE);
    for (int i = 0; i < NUM_WORKER_THREADS; i++) {
        worker_data[i].thread_id = i + 1;
        worker_data[i].start_index = i * SEGMENT_SIZE;
        worker_data[i].end_index = (i + 1) * SEGMENT_SIZE;
        pthread_create(&worker_threads[i],
                       NULL,
                       find_segment_max,
                       &worker_data[i]);}
    for (int i = 0; i < NUM_WORKER_THREADS; i++) {pthread_join(worker_threads[i], NULL);}
    printf("\n--- Worker threads finished and results collected ---\n\n");
    void* coordinator_result;
    pthread_create(&coordinator_thread,
                   NULL,
                   find_overall_max,
                   NULL);
    pthread_join(coordinator_thread, &coordinator_result);
    int overall_max = (int)(long)coordinator_result;
    printf("Overall max: %d\n", overall_max);
    return 0;}
```

The output is as follows:-

```
--- Initializing Array with Random Values (1 to 100) ---
Array initialized. Total size: 100 elements.

Thread  1 max:  95 (Segment Indices   0 to   9)
Thread  2 max:  90 (Segment Indices  10 to  19)
Thread  4 max:  91 (Segment Indices  30 to  39)
Thread  3 max:  95 (Segment Indices  20 to  29)
Thread  5 max:  96 (Segment Indices  40 to  49)
Thread  6 max:  90 (Segment Indices  50 to  59)
Thread  7 max:  98 (Segment Indices  60 to  69)
Thread  8 max:  92 (Segment Indices  70 to  79)
Thread  9 max:  92 (Segment Indices  80 to  89)
Thread 10 max:  96 (Segment Indices  90 to  99)

--- Worker threads finished and results collected ---

Overall max: 98
```

In this we randomly allocate each index of an array numbers from 1 to 100 then we divide it in a segment of 10 no. each now we find maximum for each of those segement s by passing them into threads and then overall maximum by comparing the maximums we received.