HW8.3. Threadsafe Programming

For each of the following code snippets, select one of the following that best describes its performance when run with 8 threads, compared to a single-threaded solution:

Incorrect result: The code runs into a data race, or otherwise has some chance of yielding an incorrect result. A solution with any chance of incorrect results is not considered correct, and as such is considered to be worse performance than a single-threaded solution.

Correct result, slower than serial: The code reaches the correct result, but does not speed up the computation.

Correct result, faster than serial: The code reaches the correct result, and speeds up the computation.

Q1.1:

```
//Set element i of arr to i
int A[1048576];
#pragma omp parallel
{
    for (int i = 0; i < 1048576; i++){
        A[i] = i;
    }
}
```

○ (a) Incorrect result

○ (b) Correct result, slower than serial

○ (c) Correct result, faster than serial

Q1.2:

```
//Set all elements in arr to 17
int A[1048576];
#pragma omp parallel
{
    #pragma omp for
    for (int i = 0; i < 1048576; i++){
        A[i] = 17;
    }
}
```

○ (a) Incorrect result

○ (b) Correct result, slower than serial

○ (c) Correct result, faster than serial

Q1.3:

```
//Set all elements in arr to 17
int A[1048576];
#pragma omp parallel for
    for (int i = 0; i < 1048576; i++){
        A[i] = 17;
    }
```

○ (a) Incorrect result

○ (b) Correct result, slower than serial

○ (c) Correct result, faster than serial

Q1.4:

Homework 8

Assessment overview

Total points: 0/100

Score: 0%

Question

Value: 20

History:

Awarded points: 0/20

Report an error in this question

Previous question

Next question

Attached files

No attached files

Attach a file

Attach text

```
//Set arr to be an array of Fibonacci numbers.
int A[1048576];
A[0] = 0;
A[1] = 1;
#pragma omp parallel for
    for (int i = 2; i < 1048576; i++){
        A[i] = A[i - 1] + A[i - 2];
    }
```

○ (a) Incorrect result

○ (b) Correct result, slower than serial

○ (c) Correct result, faster than serial

Q1.5:

```
int A[1048576];
initializedata(&A);
int j;
#pragma omp parallel for
    for (int i = 0; i < 1024; i++) {
        for (j = 0; j < 1024; j++) {
            A[i + j*1024] = i + j*1024;
        }
    }
```

○ (a) Incorrect result

○ (b) Correct result, slower than serial

○ (c) Correct result, faster than serial

Q1.6:

```
int A[1048576];
initializedata(&A);
#pragma omp parallel for
    for (int i = 0; i < 1024; i++) {
        for (int j = 0; j < 1024; j++) {
            A[i + j*1024] = i + j*1024;
        }
    }
```

○ (a) Incorrect result

○ (b) Correct result, slower than serial

○ (c) Correct result, faster than serial

Q1.7:

```
//Negates all elements of A
int A[1048576];
initializedata(&A);
#pragma omp parallel
{
    for (int i = 0; i < 1048576; i++)
        if(i%omp_get_num_threads() == omp_get_thread_num())
            A[i] = -A[i];
}
```

○ (a) Incorrect result

○ (b) Correct result, slower than serial

○ (c) Correct result, faster than serial

Q1.8:

```
int A[1048576];
initializedata(&A);
int sum = 0;
#pragma omp parallel for
    for (int i = 0; i < 1048576; i++){
        sum += A[i];
    }
```

○ (a) Incorrect result

○ (b) Correct result, slower than serial

○ (c) Correct result, faster than serial

Q1.9:

```
int A[1048576];
initializedata(&A);
int sum = 0;
#pragma omp parallel for
    for (int i = 0; i < 1048576; i++){
        #pragma omp critical
        sum += A[i];
    }
```

○ (a) Incorrect result

○ (b) Correct result, slower than serial

○ (c) Correct result, faster than serial

Q1.10:

```
int A[1048576];
initializedata(&A);
int sum = 0;
#pragma omp parallel
{
    int private_sum = 0;
    #pragma omp for
    for (int i = 0; i < 1048576; i++){
        private_sum += A[i];
    }
    #pragma omp critical
    sum += private_sum;
}
```

○ (a) Incorrect result

○ (b) Correct result, slower than serial

○ (c) Correct result, faster than serial

Save & Grade *20 attempts left* Save only *Additional attempts available with new variants* ?