NAME	JACOB JOHN
REGISTER NO.	16BCE2205
E-MAIL	jacob.john2016@vitstudent.ac.in

LAB ASSESSMENT #4

SCENARIO – 1

Write a simple OpenMP program to employ a ‘Work Sharing’ clause to assign each thread an independent set of iterations. In order to explore its practical use, you are advised to read and understand the following statements.

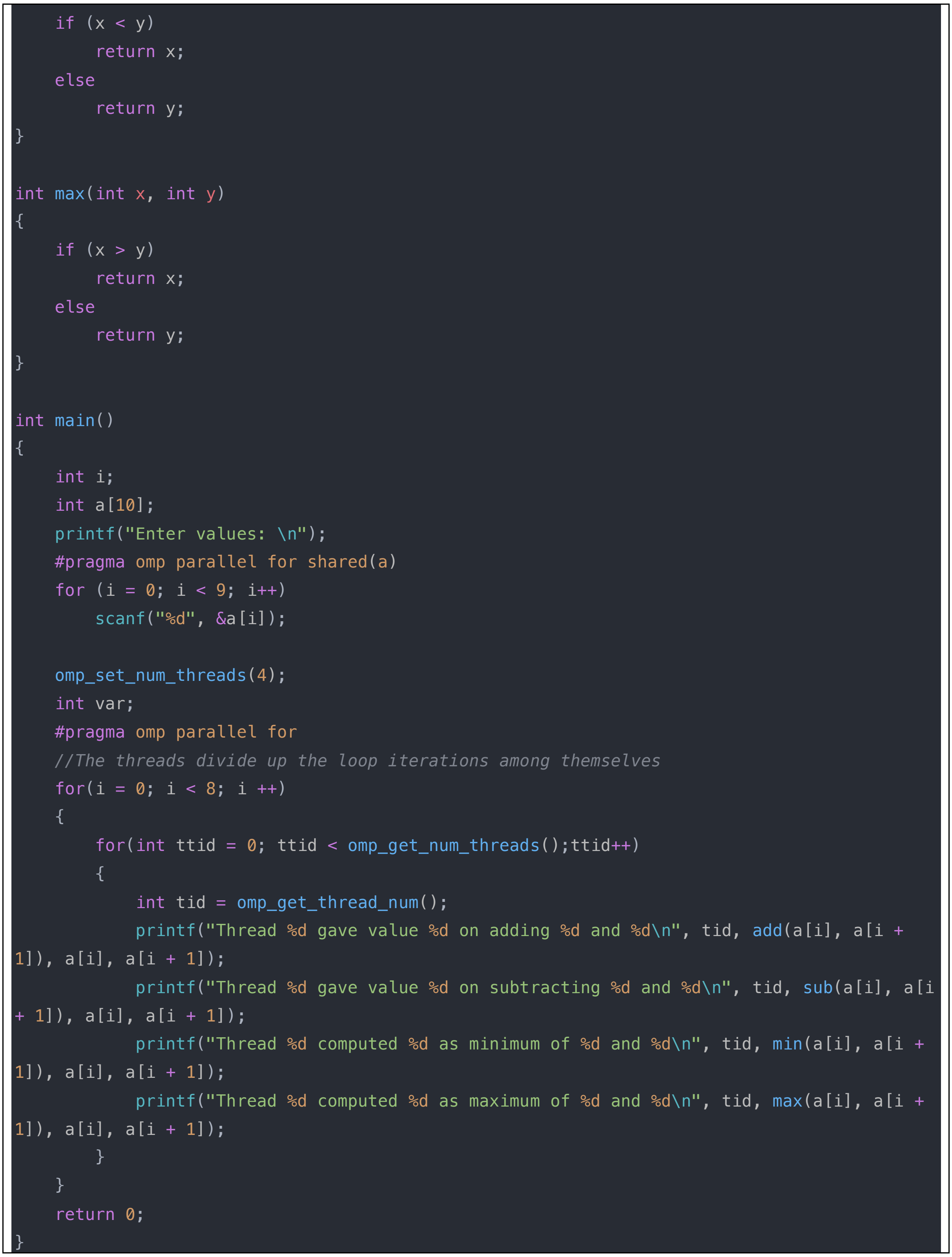
1. Assign each thread an independent set of iterations;
2. Threads must wait at the end
3. Can combine the directives:
4. #pragma omp parallel for
5. Only simple kinds of for loops:
    a. Only one signed integer variable
    b. Initialization: var=init
    c. Comparison: var op last op: , <=, >=
    d. Increment: var++, var--, var+=incr, var-=incr, etc.

Brief about your approach:

The code below uses a parallel pragma for loop to distribute the work among three threads. Each thread executes every task, i.e., the four functions defined on all the threads – addition, subtraction, minimum and maximum. The threads divide up the loop iterations among themselves using the for clause.

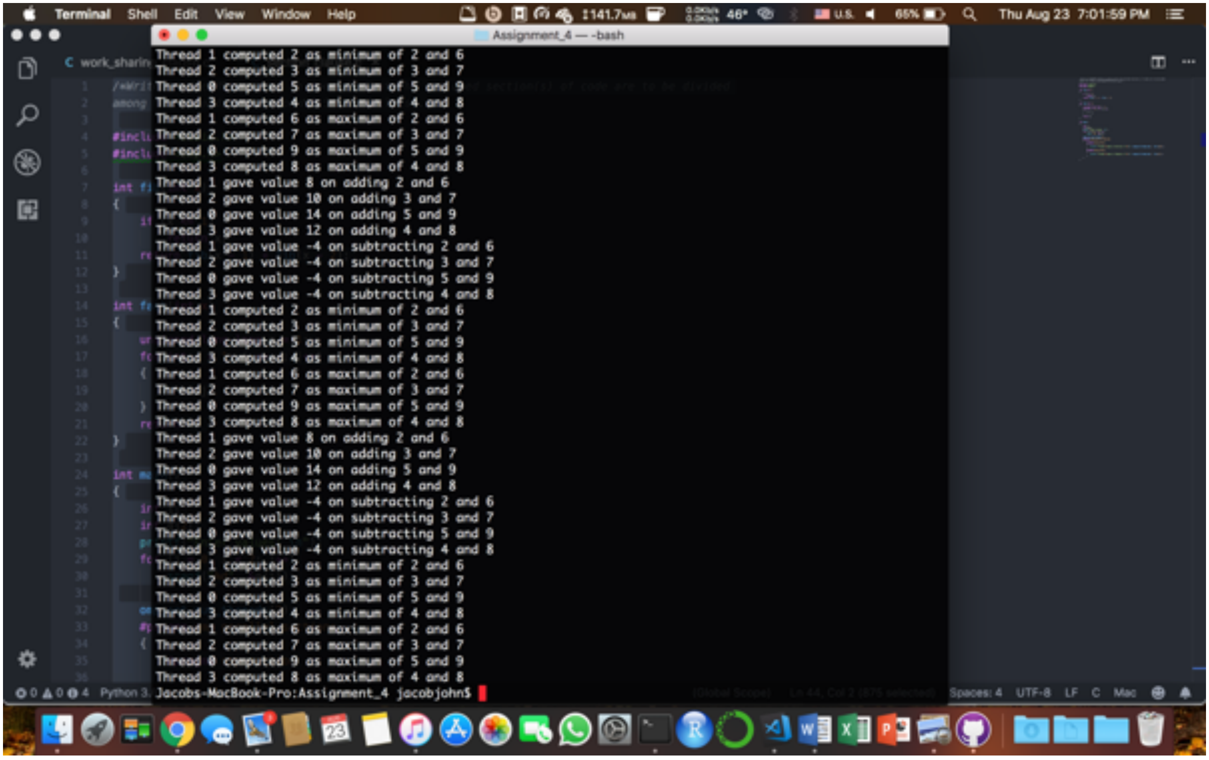
Code:

```
/* ‘Work Sharing’ clause to assign each thread an
independent set of iterations.*/

#include <stdio.h>
#include <omp.h>

int add(int x, int y)
{
    return (x + y);
}

int sub(int x, int y)
{
    return (x - y);
}

int min(int x, int y)
{
```



```
    if (x < y)
        return x;
    else
        return y;
}

int max(int x, int y)
{
    if (x > y)
        return x;
    else
        return y;
}

int main()
{
    int i;
    int a[10];
    printf("Enter values: \n");
    #pragma omp parallel for shared(a)
    for (i = 0; i < 9; i++)
        scanf("%d", &a[i]);

    omp_set_num_threads(4);
    int var;
    #pragma omp parallel for
    //The threads divide up the loop iterations among themselves
    for(i = 0; i < 8; i ++)
    {
        for(int ttid = 0; ttid < omp_get_num_threads();ttid++)
        {
            int tid = omp_get_thread_num();
            printf("Thread %d gave value %d on adding %d and %d\n", tid, add(a[i], a[i +
1]), a[i], a[i + 1]);
            printf("Thread %d gave value %d on subtracting %d and %d\n", tid, sub(a[i], a[i
+ 1]), a[i], a[i + 1]);
            printf("Thread %d computed %d as minimum of %d and %d\n", tid, min(a[i], a[i +
1]), a[i], a[i + 1]);
            printf("Thread %d computed %d as maximum of %d and %d\n", tid, max(a[i], a[i +
1]), a[i], a[i + 1]);
        }
    }
    return 0;
}
```

Output:



```
Thread 1 computed 2 as minimum of 2 and 6
Thread 2 computed 3 as minimum of 3 and 7
Thread 0 computed 5 as minimum of 5 and 9
Thread 3 computed 4 as minimum of 4 and 8
Thread 1 computed 6 as maximum of 2 and 6
Thread 2 computed 7 as maximum of 3 and 7
Thread 0 computed 9 as maximum of 5 and 9
Thread 3 computed 8 as maximum of 4 and 8
Thread 1 gave value 8 on adding 2 and 6
Thread 2 gave value 10 on adding 3 and 7
Thread 0 gave value 14 on adding 5 and 9
Thread 3 gave value 12 on adding 4 and 8
Thread 1 gave value -4 on subtracting 2 and 6
Thread 2 gave value -4 on subtracting 3 and 7
Thread 0 gave value -4 on subtracting 5 and 9
Thread 3 gave value -4 on subtracting 4 and 8
Thread 1 computed 2 as minimum of 2 and 6
Thread 2 computed 3 as minimum of 3 and 7
Thread 0 computed 5 as minimum of 5 and 9
Thread 3 computed 4 as minimum of 4 and 8
Thread 1 computed 6 as maximum of 2 and 6
Thread 2 computed 7 as maximum of 3 and 7
Thread 0 computed 9 as maximum of 5 and 9
Thread 3 computed 8 as maximum of 4 and 8
Thread 1 gave value 8 on adding 2 and 6
Thread 2 gave value 10 on adding 3 and 7
Thread 0 gave value 14 on adding 5 and 9
Thread 3 gave value 12 on adding 4 and 8
Thread 1 gave value -4 on subtracting 2 and 6
Thread 2 gave value -4 on subtracting 3 and 7
Thread 0 gave value -4 on subtracting 5 and 9
Thread 3 gave value -4 on subtracting 4 and 8
Thread 1 computed 2 as minimum of 2 and 6
Thread 2 computed 3 as minimum of 3 and 7
Thread 0 computed 5 as minimum of 5 and 9
Thread 3 computed 4 as minimum of 4 and 8
Thread 1 computed 6 as maximum of 2 and 6
Thread 2 computed 7 as maximum of 3 and 7
Thread 0 computed 9 as maximum of 5 and 9
Thread 3 computed 8 as maximum of 4 and 8
Jacobs-MacBook-Pro:Assignment_4 jacobjohn$
```

Execution

```
Jacobs-MacBook-Pro:Assignment_4 jacobjohn$ gcc-8 -fopenmp work_sharing_clause.c
Jacobs-MacBook-Pro:Assignment_4 jacobjohn$ ./a.out
Enter values:
1
2
3
4
5
6
7
8
9
Thread 1 gave value 12 on adding 9 and 3
Thread 2 gave value 9 on adding 7 and 2
Thread 1 gave value 6 on subtracting 9 and 3
Thread 3 gave value 10 on adding 6 and 4
Thread 0 gave value 6 on adding 1 and 5
Thread 2 gave value 5 on subtracting 7 and 2
Thread 1 computed 3 as minimum of 9 and 3
Thread 3 gave value 2 on subtracting 6 and 4
Thread 0 gave value -4 on subtracting 1 and 5
Thread 2 computed 2 as minimum of 7 and 2
Thread 1 computed 9 as maximum of 9 and 3
Thread 3 computed 4 as minimum of 6 and 4
Thread 0 computed 1 as minimum of 1 and 5
Thread 2 computed 7 as maximum of 7 and 2
Thread 1 gave value 12 on adding 9 and 3
Thread 3 computed 6 as maximum of 6 and 4
Thread 0 computed 5 as maximum of 1 and 5
Thread 2 gave value 9 on adding 7 and 2
Thread 1 gave value 6 on subtracting 9 and 3
Thread 3 gave value 10 on adding 6 and 4
Thread 0 gave value 6 on adding 1 and 5
Thread 2 gave value 5 on subtracting 7 and 2
Thread 1 computed 3 as minimum of 9 and 3
Thread 3 gave value 2 on subtracting 6 and 4
Thread 0 gave value -4 on subtracting 1 and 5
Thread 2 computed 2 as minimum of 7 and 2
Thread 1 computed 9 as maximum of 9 and 3
Thread 3 computed 4 as minimum of 6 and 4
Thread 0 computed 1 as minimum of 1 and 5
Thread 2 computed 7 as maximum of 7 and 2
Thread 1 gave value 12 on adding 9 and 3
Thread 3 computed 6 as maximum of 6 and 4
Thread 0 computed 5 as maximum of 1 and 5
Thread 2 gave value 9 on adding 7 and 2
Thread 1 gave value 6 on subtracting 9 and 3
Thread 3 gave value 10 on adding 6 and 4
Thread 0 gave value 6 on adding 1 and 5
Thread 2 gave value 5 on subtracting 7 and 2
Thread 1 computed 3 as minimum of 9 and 3
Thread 3 gave value 2 on subtracting 6 and 4
Thread 0 gave value -4 on subtracting 1 and 5
Thread 2 computed 2 as minimum of 7 and 2
Thread 1 computed 9 as maximum of 9 and 3
Thread 3 computed 4 as minimum of 6 and 4
Thread 0 computed 1 as minimum of 1 and 5
Thread 2 computed 7 as maximum of 7 and 2
Thread 1 gave value 12 on adding 9 and 3
Thread 3 computed 6 as maximum of 6 and 4
Thread 0 computed 5 as maximum of 1 and 5
Thread 2 gave value 9 on adding 7 and 2
Thread 1 gave value 6 on subtracting 9 and 3
Thread 3 gave value 10 on adding 6 and 4
Thread 0 gave value 6 on adding 1 and 5
Thread 2 gave value 5 on subtracting 7 and 2
Thread 1 computed 3 as minimum of 9 and 3
Thread 3 gave value 2 on subtracting 6 and 4
Thread 0 gave value -4 on subtracting 1 and 5
Thread 2 computed 2 as minimum of 7 and 2
Thread 1 computed 9 as maximum of 9 and 3
Thread 3 computed 4 as minimum of 6 and 4
Thread 0 computed 1 as minimum of 1 and 5
Thread 2 computed 7 as maximum of 7 and 2
Thread 1 gave value 10 on adding 3 and 7
Thread 3 computed 6 as maximum of 6 and 4
Thread 0 computed 5 as maximum of 1 and 5
Thread 2 gave value 8 on adding 2 and 6
Thread 1 gave value -4 on subtracting 3 and 7
Thread 3 gave value 12 on adding 4 and 8
Thread 0 gave value 14 on adding 5 and 9
Thread 2 gave value -4 on subtracting 2 and 6
Thread 1 computed 3 as minimum of 3 and 7
Thread 3 gave value -4 on subtracting 4 and 8
Thread 0 gave value -4 on subtracting 5 and 9
Thread 2 computed 2 as minimum of 2 and 6
Thread 1 computed 7 as maximum of 3 and 7
Thread 3 computed 4 as minimum of 4 and 8
Thread 0 computed 5 as minimum of 5 and 9
Thread 2 computed 6 as maximum of 2 and 6
Thread 1 gave value 10 on adding 3 and 7
Thread 3 computed 8 as maximum of 4 and 8
Thread 0 computed 9 as maximum of 5 and 9
Thread 2 gave value 8 on adding 2 and 6
Thread 1 gave value -4 on subtracting 3 and 7
Thread 3 gave value 12 on adding 4 and 8
Thread 0 gave value 14 on adding 5 and 9
Thread 2 gave value -4 on subtracting 2 and 6
Thread 1 computed 3 as minimum of 3 and 7
Thread 3 gave value -4 on subtracting 4 and 8
Thread 0 gave value -4 on subtracting 5 and 9
Thread 2 computed 2 as minimum of 2 and 6
Thread 1 computed 7 as maximum of 3 and 7
Thread 3 computed 4 as minimum of 4 and 8
Thread 0 computed 5 as minimum of 5 and 9
Thread 2 computed 6 as maximum of 2 and 6
Thread 1 gave value 10 on adding 3 and 7
Thread 3 computed 8 as maximum of 4 and 8
Thread 0 computed 9 as maximum of 5 and 9
Thread 2 gave value 8 on adding 2 and 6
Thread 1 gave value -4 on subtracting 3 and 7
Thread 3 gave value 12 on adding 4 and 8
Thread 0 gave value 14 on adding 5 and 9
Thread 2 gave value -4 on subtracting 2 and 6
Thread 1 computed 3 as minimum of 3 and 7
Thread 3 gave value -4 on subtracting 4 and 8
Thread 0 gave value -4 on subtracting 5 and 9
Thread 2 computed 2 as minimum of 2 and 6
Thread 1 computed 7 as maximum of 3 and 7
Thread 3 computed 4 as minimum of 4 and 8
Thread 0 computed 5 as minimum of 5 and 9
Thread 2 computed 6 as maximum of 2 and 6
Thread 1 gave value 10 on adding 3 and 7
Thread 3 computed 8 as maximum of 4 and 8
Thread 0 computed 9 as maximum of 5 and 9
Thread 2 gave value 8 on adding 2 and 6
Thread 1 gave value -4 on subtracting 3 and 7
Thread 3 gave value 12 on adding 4 and 8
Thread 0 gave value 14 on adding 5 and 9
Thread 2 gave value -4 on subtracting 2 and 6
Thread 1 computed 3 as minimum of 3 and 7
Thread 3 gave value -4 on subtracting 4 and 8
Thread 0 gave value -4 on subtracting 5 and 9
Thread 2 computed 2 as minimum of 2 and 6
Thread 1 computed 7 as maximum of 3 and 7
Thread 3 computed 4 as minimum of 4 and 8
Thread 0 computed 5 as minimum of 5 and 9
Thread 2 computed 6 as maximum of 2 and 6
Thread 3 computed 8 as maximum of 4 and 8
Thread 0 computed 9 as maximum of 5 and 9
```

SCENARIO – 2

Write an OpenMP program to specify that the enclosed section(s) of code are to be divided among the threads using OpenMP SECTION clause.

Description

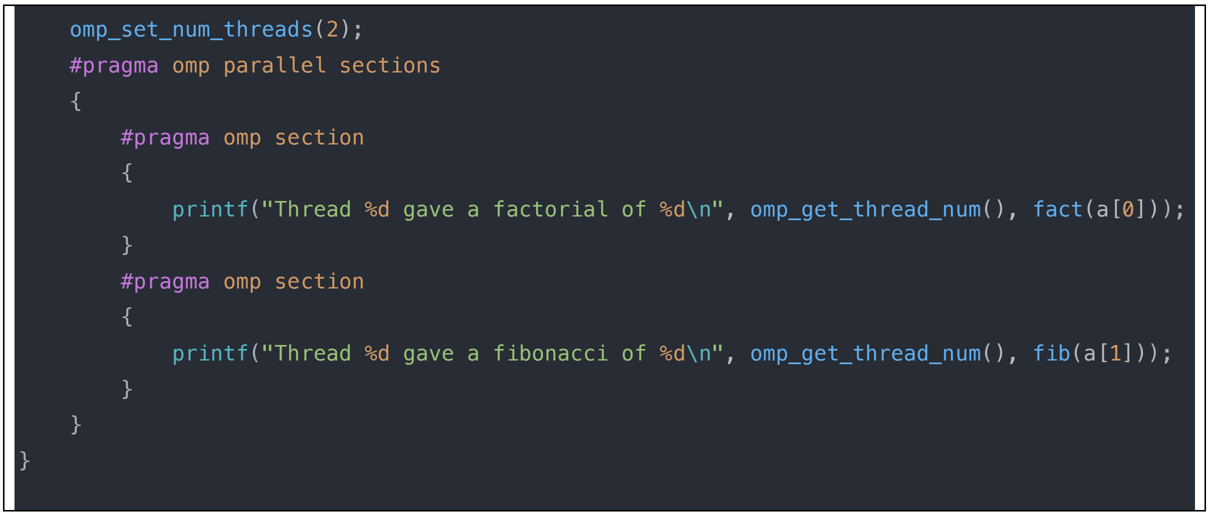
Independent SECTION directives are nested within a SECTIONS directive. Each SECTION is executed once by a thread in the team. Different sections may be executed by different threads. It is possible that for a thread to execute more than one section if it is quick enough and the implementation permits such.

Brief about your approach:

This code uses the section clause to excecute openmp threads. According to OpenMP standard 3.1, section 2.5.2, “The sections construct is a noniterative worksharing construct that contains a set of structured blocks that are to be distributed among and executed by the threads in a team. Each structured block is executed once by one of the threads in the team in the context of its implicit task.”

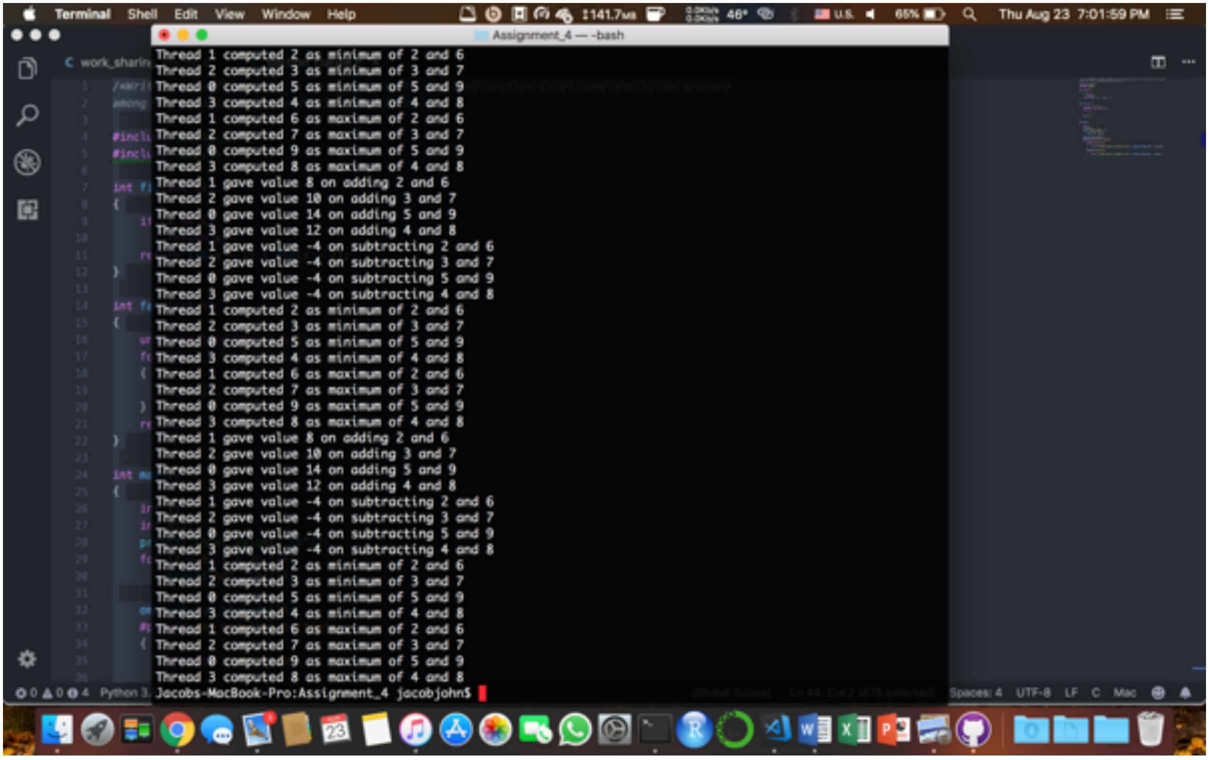
Code:

```
/*Write an OpenMP program to specify that the enclosed section(s) of code are to be divided
among the threads using OpenMP SECTION clause.*/

#include <stdio.h>
#include <omp.h>

int fib(int x)
{
    if (x <= 1)
        return x;
    return fib(x - 1) + fib(x - 2);
}

int fact(int x)
{
    unsigned long long n = 1;
    for(int j = 1; j <= x; j++)
    {
        n *= j;
    }
    return n;
}

int main()
{
    int i;
    int a[2];
    printf("Enter values: \n");
    for (i = 0; i < 2; i++)
        scanf("%d", &a[i]);
```



```c
    omp_set_num_threads(2);
    #pragma omp parallel sections
    {
        #pragma omp section
        {
            printf("Thread %d gave a factorial of %d\n", omp_get_thread_num(), fact(a[0]));
        }
        #pragma omp section
        {
            printf("Thread %d gave a fibonacci of %d\n", omp_get_thread_num(), fib(a[1]));
        }
    }
}
```

Output:



```
Thread 1 computed 2 as minimum of 2 and 6
Thread 2 computed 3 as minimum of 3 and 7
Thread 0 computed 5 as minimum of 5 and 9
Thread 3 computed 4 as minimum of 4 and 8
Thread 1 computed 6 as maximum of 2 and 6
Thread 2 computed 7 as maximum of 3 and 7
Thread 0 computed 9 as maximum of 5 and 9
Thread 3 computed 8 as maximum of 4 and 8
Thread 1 gave value 8 on adding 2 and 6
Thread 2 gave value 10 on adding 3 and 7
Thread 0 gave value 14 on adding 5 and 9
Thread 3 gave value 12 on adding 4 and 8
Thread 1 gave value -4 on subtracting 2 and 6
Thread 2 gave value -4 on subtracting 3 and 7
Thread 0 gave value -4 on subtracting 5 and 9
Thread 3 gave value -4 on subtracting 4 and 8
Thread 1 computed 2 as minimum of 2 and 6
Thread 2 computed 3 as minimum of 3 and 7
Thread 0 computed 5 as minimum of 5 and 9
Thread 3 computed 4 as minimum of 4 and 8
Thread 1 computed 6 as maximum of 2 and 6
Thread 2 computed 7 as maximum of 3 and 7
Thread 0 computed 9 as maximum of 5 and 9
Thread 3 computed 8 as maximum of 4 and 8
Thread 1 gave value 8 on adding 2 and 6
Thread 2 gave value 10 on adding 3 and 7
Thread 0 gave value 14 on adding 5 and 9
Thread 3 gave value 12 on adding 4 and 8
Thread 1 gave value -4 on subtracting 2 and 6
Thread 2 gave value -4 on subtracting 3 and 7
Thread 0 gave value -4 on subtracting 5 and 9
Thread 3 gave value -4 on subtracting 4 and 8
Thread 1 computed 2 as minimum of 2 and 6
Thread 2 computed 3 as minimum of 3 and 7
Thread 0 computed 5 as minimum of 5 and 9
Thread 3 computed 4 as minimum of 4 and 8
Thread 1 computed 6 as maximum of 2 and 6
Thread 2 computed 7 as maximum of 3 and 7
Thread 0 computed 9 as maximum of 5 and 9
Thread 3 computed 8 as maximum of 4 and 8
Jacobs-MacBook-Pro:Assignment_4 jacobjohn$
```

Execution

```
Jacobs-MacBook-Pro:Assignment_4 jacobjohn$ gcc-8 -fopenmp section_clause.c
Jacobs-MacBook-Pro:Assignment_4 jacobjohn$ ./a.out
Enter values:
10
1
Thread 0 gave a factorial of 3628800
Thread 1 gave a fibonacci of 1
```