

NAME: MOUNVI PODAPATI
REG. NO: 19BCE0396
SLOT: L15 + L16
FACULTY: PROF. DEEPAK B.D
DATED: 1ST SEPTEMBER 2021

LAB ASSESSMENT – 3

AIM: Write a simple OpenMP program to demonstrate Arithmetic Operation using Section Clause.

SOURCE CODE:

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

//Expression: (1+2)+(3-4)+(5*6)+(6/3)

int addition (int x,int y) {
    return x+y;
}
int subtraction (int x,int y) {
    return x-y;
}
int multiplication (int x,int y) {
    return x*y;
}
int division (int x,int y) {
    return x/y;
}

void main() {
    int sum = 0 , diff = 0 , prod = 0, div = 0 , total = 0;
    #pragma omp parallel sections
    {
        #pragma omp section
        sum = addition(1,2);

        #pragma omp section
        diff = subtraction(3,4);

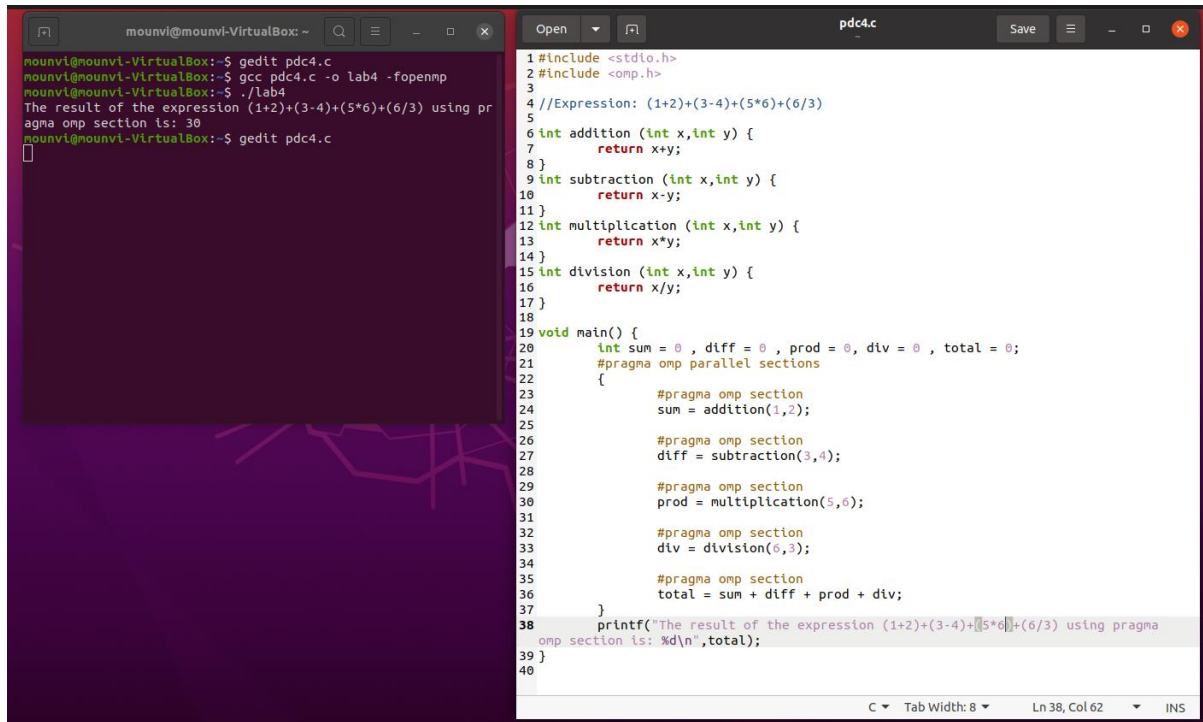
        #pragma omp section
        prod = multiplication(5,6);

        #pragma omp section
        div = division(6,3);

        #pragma omp section
        total = sum + diff + prod + div;
    }
}
```

```
printf("The result of the expression (1+2)+(3-4)+(5*6)+(6/3) using pragma omp section is:
%d\n",total);
}
```

EXECUTION:



The screenshot displays a terminal window on the left and a code editor on the right. The terminal shows the compilation and execution of a C program named `pd4.c`. The output of the program is "The result of the expression (1+2)+(3-4)+(5*6)+(6/3) using pragma omp section is: 30". The code editor shows the source code of `pd4.c`, which includes headers for `stdio.h` and `omp.h`. It defines four functions: `addition`, `subtraction`, `multiplication`, and `division`. The `main` function uses OpenMP sections to execute these operations in parallel. The final result is printed using `printf`.

```
1 #include <stdio.h>
2 #include <omp.h>
3
4 //Expression: (1+2)+(3-4)+(5*6)+(6/3)
5
6 int addition (int x,int y) {
7     return x+y;
8 }
9 int subtraction (int x,int y) {
10    return x-y;
11 }
12 int multiplication (int x,int y) {
13    return x*y;
14 }
15 int division (int x,int y) {
16    return x/y;
17 }
18
19 void main() {
20     int sum = 0 , diff = 0 , prod = 0, div = 0 , total = 0;
21     #pragma omp parallel sections
22     {
23         #pragma omp section
24         sum = addition(1,2);
25
26         #pragma omp section
27         diff = subtraction(3,4);
28
29         #pragma omp section
30         prod = multiplication(5,6);
31
32         #pragma omp section
33         div = division(6,3);
34
35         #pragma omp section
36         total = sum + diff + prod + div;
37     }
38     printf("The result of the expression (1+2)+(3-4)+(5*6)+(6/3) using pragma
39 omp section is: %d\n",total);
40 }
```

REMARKS:

Section clause has been exploited and using this I have verified arithmetic operations like addition, subtraction, multiplication and division. The results were calculated separately by using pragma section. The section clause is used to indicate that the particular parallel section will be having multiple subsections that are executed in parallel. The section clause identifies these subsections and the section clause can be very useful when different parts of the same logic can be conducted in parallel, enhancing time efficiency.