Computer Science & Engineering

CSE4001

Parallel and Distributed Computing

LAB ASSIGNMENT 1

Submitted to **Prof. DEEBAK B.D.**

TOPIC: INTRODUCTION TO OPENMP

NAME: PUNIT MIDDHA

REG.NO: 19BCE2060

SLOT: L55+L56

DATE: 17/08/2021

Aim:

Write a simple OpenMP program to demonstrate the parallel loop construct.

a. Use OMP_SET_THREAD_NUM( ) and OMP_GET_THREAD_NUM( ) to find the number of processing unit
b. Use function invoke to print 'Hello World'
c. To examine the above scenario, the functions such as omp_get_num_procs(), omp_set_num_threads(),omp_get_num_threads(),omp_in_parallel(), omp_get_dynamic() and omp_get_nested() are listed and the explanation is given below to explore the concept practically.
   omp_set_num_threads() - takes an integer argument and requests that the Operating System provide that number of threads in subsequent parallel regions.

   omp_get_num_threads() (integer function) - returns the actual number of threads in the current team of threads.

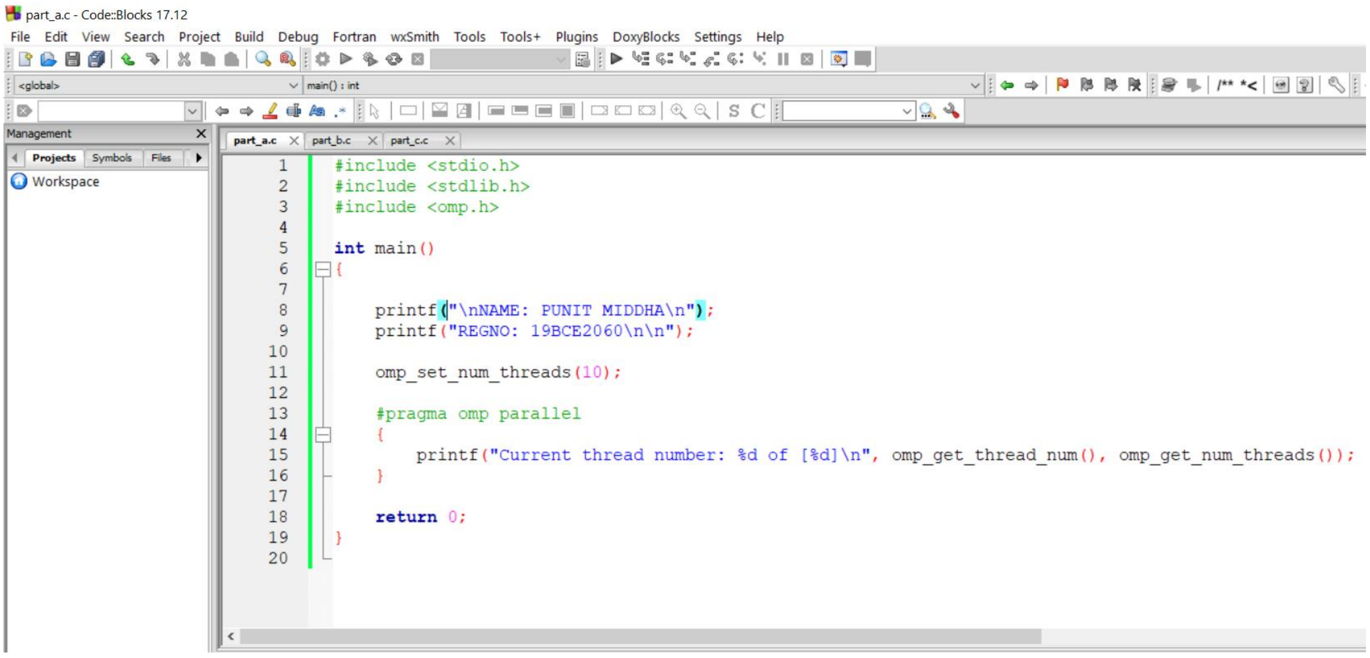
   omp_get_thread_num() (integer function) - returns the ID of a thread, where the ID ranges from 0 to the number of threads minus 1. The thread with the ID of 0 is the master thread. **omp_get_num_procs()** - returns the number of processors that are available when the function is called.

   omp_get_dynamic() - returns a value that indicates if the number of threads available in subsequent parallel region can be adjusted by the run time. o omp_get_nested( ) returns a value that indicates if nested parallelism is enabled.

PART-A:

SOURCE CODE:

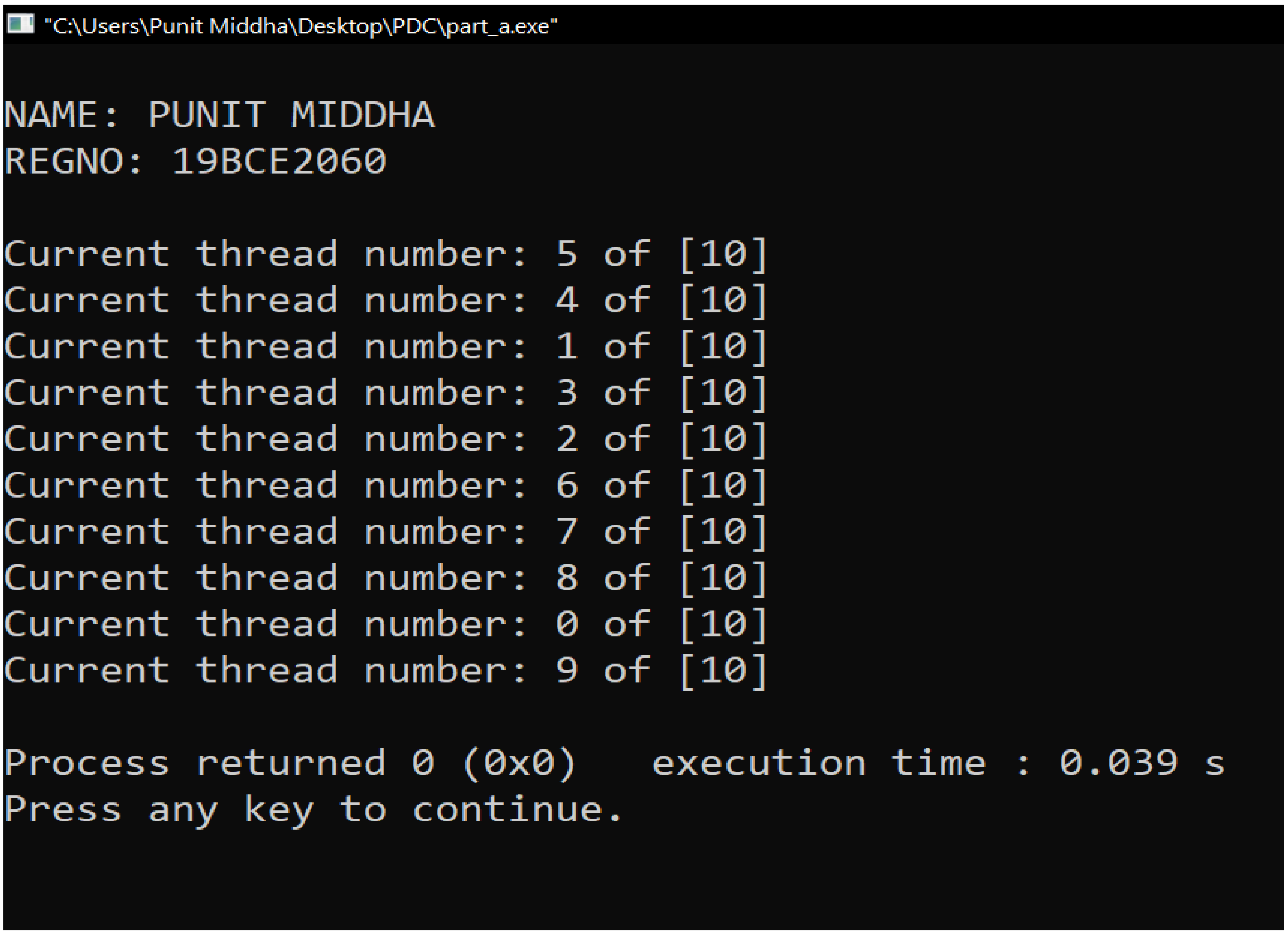
```
#include <stdio.h>

#include <stdlib.h>

#include <omp.h>

int main()

{

    printf("\nNAME: PUNIT MIDDHA\n");

    printf("REGNO: 19BCE2060\n\n");

    omp_set_num_threads(10);

    #pragma omp parallel

    {

        printf("Current thread number: %d of [%d]\n", omp_get_thread_num(), 
omp_get_num_threads());

    }
    return 0;

}
```



part_a.c - Code::Blocks 17.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Management
Projects Symbols Files
Workspace
part_a.c part_b.c part_c.c

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

int main()
{

    printf("\nNAME: PUNIT MIDDHA\n");
    printf("REGNO: 19BCE2060\n\n");

    omp_set_num_threads(10);

    #pragma omp parallel
    {
        printf("Current thread number: %d of [%d]\n", omp_get_thread_num(), omp_get_num_threads());
    }

    return 0;
}
```

EXECUTION:



```
"C:\Users\Punit Middha\Desktop\PDC\part_a.exe"

NAME: PUNIT MIDDHA
REGNO: 19BCE2060

Current thread number: 5 of [10]
Current thread number: 4 of [10]
Current thread number: 1 of [10]
Current thread number: 3 of [10]
Current thread number: 2 of [10]
Current thread number: 6 of [10]
Current thread number: 7 of [10]
Current thread number: 8 of [10]
Current thread number: 0 of [10]
Current thread number: 9 of [10]

Process returned 0 (0x0)   execution time : 0.039 s
Press any key to continue.
```

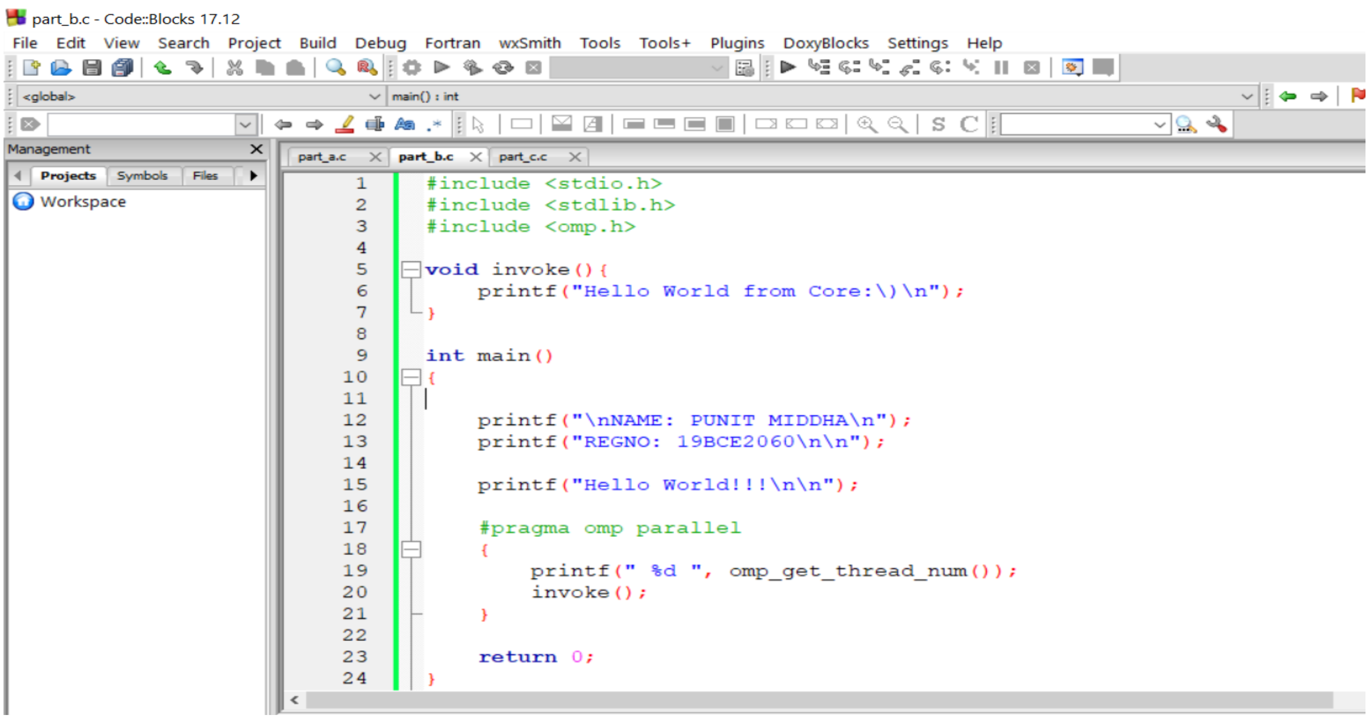
REMARKS:

- **omp_set_num_threads** function was used to set the total number of threads to 10.
- **omp_get_thread_num** function to get the id of the current thread i.e., from 0 to 9.
- **omp_get_num_threads** to print the total no of threads, 10 in this case.

PART-B:

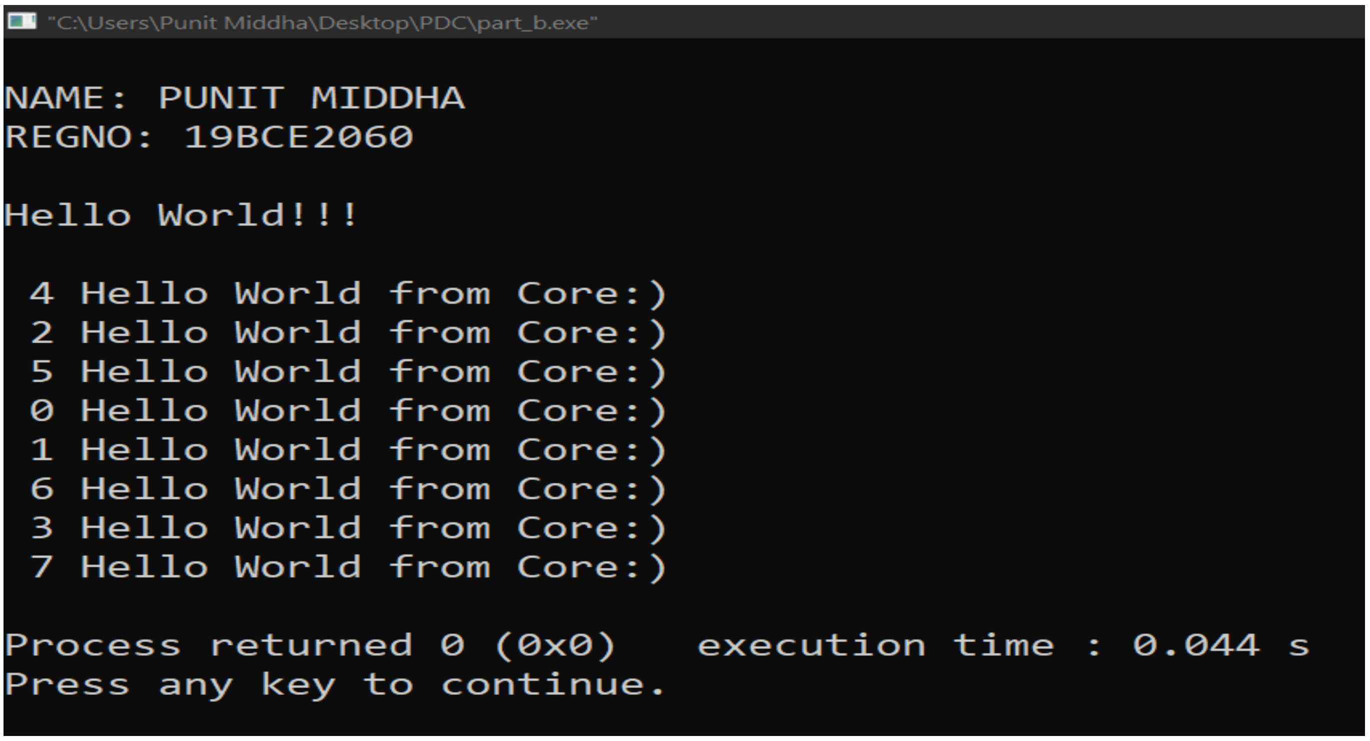
SOURCE CODE:

```
#include <stdio.h>

#include <stdlib.h>

#include <omp.h>

void invoke(){

    printf("Hello World from Core:\)\n");

}

int main()

{

    printf("\nNAME: PUNIT MIDDHA\n");

    printf("REGNO: 19BCE2060\n\n");

    printf("Hello World!!!\n\n");

    #pragma omp parallel

    {

        printf(" %d ", omp_get_thread_num());

        invoke();

    }

    return 0;

}
```


part_b.c - Code::Blocks 17.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
main() : int
Management
Projects Symbols Files
Workspace
part_a.c
part_b.c
part_c.c

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

void invoke(){
    printf("Hello World from Core:\)\n");
}

int main()
{

    printf("\nNAME: PUNIT MIDDHA\n");
    printf("REGNO: 19BCE2060\n\n");

    printf("Hello World!!!\n\n");

    #pragma omp parallel
    {
        printf(" %d ", omp_get_thread_num());
        invoke();
    }

    return 0;
}
```

EXECUTION:



```

NAME: PUNIT MIDDHA
REGNO: 19BCE2060

Hello World!!!

 4 Hello World from Core:)
 2 Hello World from Core:)
 5 Hello World from Core:)
 0 Hello World from Core:)
 1 Hello World from Core:)
 6 Hello World from Core:)
 3 Hello World from Core:)
 7 Hello World from Core:)

Process returned 0 (0x0)   execution time : 0.044 s
Press any key to continue.
```

REMARKS:

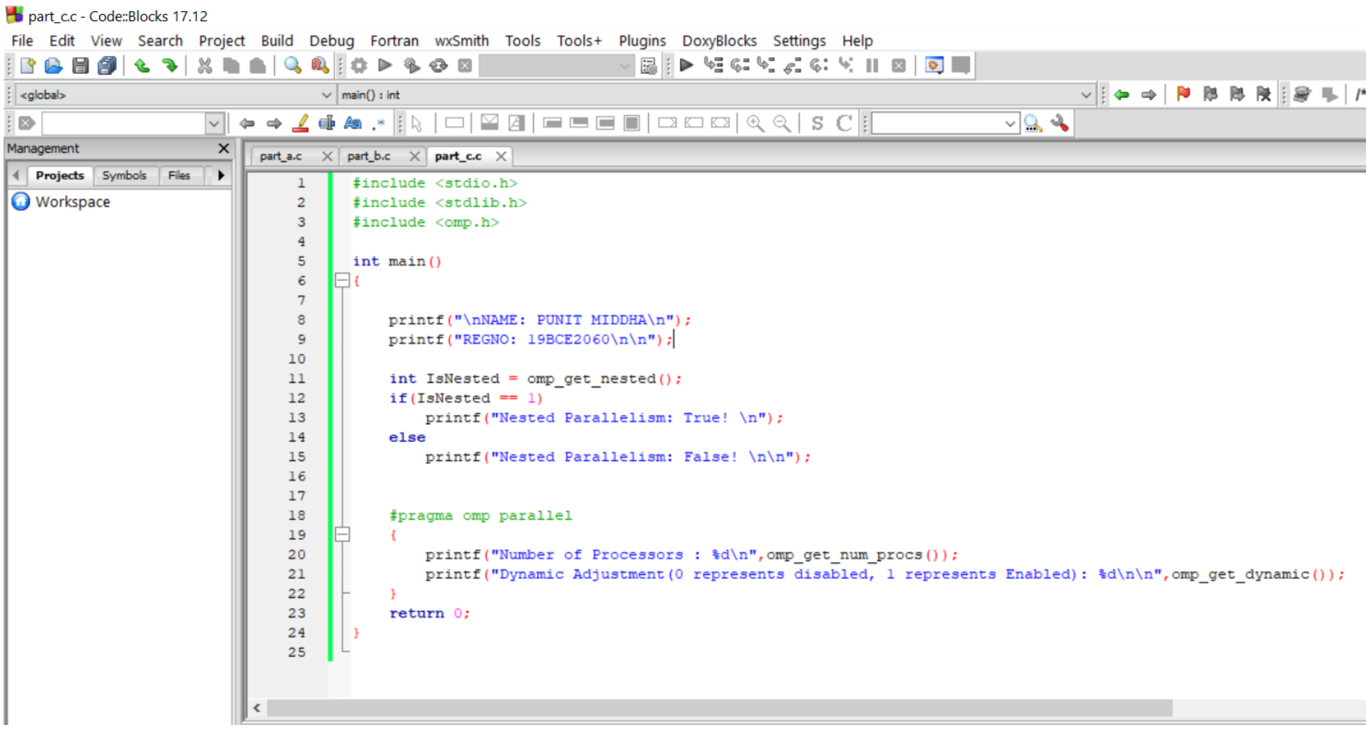
- Here, “Hello World!!!” was printed outside pragma and it printed only one time.
- **#pragma omp parallel** block is used to fork additional threads to carry out the work in parallel according to the no. of threads present.

- The function **invoke()** was called inside the pragma block and it worked parallelly for 8 times.

PART-C:

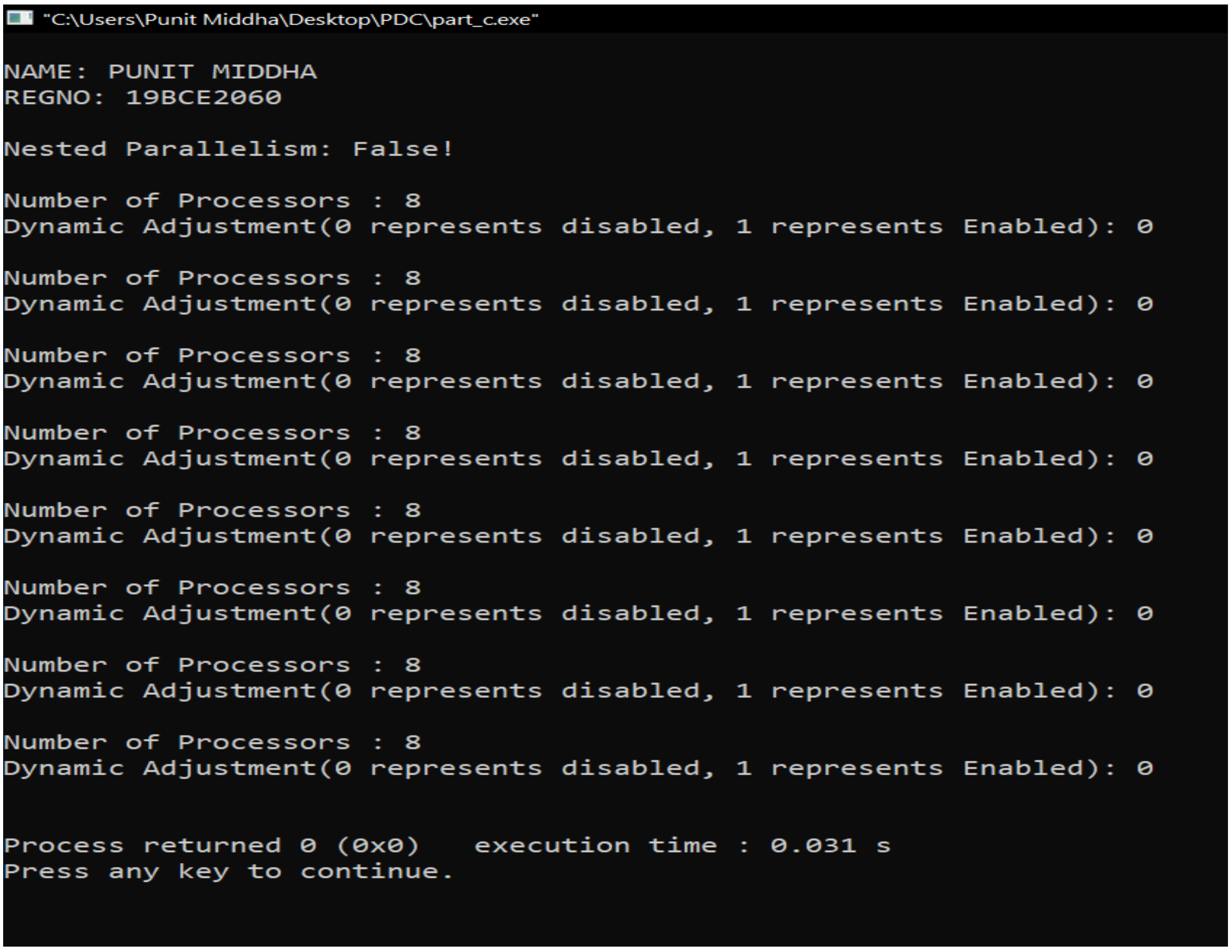
SOURCE CODE:

```c
#include <stdio.h>

#include <stdlib.h>

#include <omp.h>

int main()

{

    printf("\nNAME: PUNIT MIDDHA\n");

    printf("REGNO: 19BCE2060\n\n");

    int IsNested = omp_get_nested();

    if(IsNested == 1)

        printf("Nested Parallelism: True! \n");

    else

        printf("Nested Parallelism: False! \n\n");


    #pragma omp parallel

    {

        printf("Number of Processors : %d\n",omp_get_num_procs());

        printf("Dynamic Adjustment(0 represents disabled, 1 represents 
Enabled): %d\n\n",omp_get_dynamic());

    }

    return 0;

}
```


part_c.c - Code::Blocks 17.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
main() : int
Management
Projects Symbols Files
Workspace
part_a.c part_b.c part_c.c

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

int main()
{

    printf("\nNAME: PUNIT MIDDHA\n");
    printf("REGNO: 19BCE2060\n\n");

    int IsNested = omp_get_nested();
    if(IsNested == 1)
        printf("Nested Parallelism: True! \n");
    else
        printf("Nested Parallelism: False! \n\n");


    #pragma omp parallel
    {
        printf("Number of Processors : %d\n",omp_get_num_procs());
        printf("Dynamic Adjustment(0 represents disabled, 1 represents Enabled): %d\n\n",omp_get_dynamic());
    }
    return 0;
}
```

EXECUTION:



```
NAME: PUNIT MIDDHA
REGNO: 19BCE2060

Nested Parallelism: False!

Number of Processors : 8
Dynamic Adjustment(0 represents disabled, 1 represents Enabled): 0

Number of Processors : 8
Dynamic Adjustment(0 represents disabled, 1 represents Enabled): 0

Number of Processors : 8
Dynamic Adjustment(0 represents disabled, 1 represents Enabled): 0

Number of Processors : 8
Dynamic Adjustment(0 represents disabled, 1 represents Enabled): 0

Number of Processors : 8
Dynamic Adjustment(0 represents disabled, 1 represents Enabled): 0

Number of Processors : 8
Dynamic Adjustment(0 represents disabled, 1 represents Enabled): 0

Number of Processors : 8
Dynamic Adjustment(0 represents disabled, 1 represents Enabled): 0

Number of Processors : 8
Dynamic Adjustment(0 represents disabled, 1 represents Enabled): 0


Process returned 0 (0x0)   execution time : 0.031 s
Press any key to continue.
```

REMARKS:

- The omp_get_nested function was used to find out whether nested parallelism was present or not and as shown in the output it is not present.
- omp_get_num_procs function returns the number of processors available in the system i.e., 8 as shown in output.
- omp_get_dynamic returns 0 in each case that indicates that the number of threads available in subsequent parallel region cannot be adjusted by the run time.