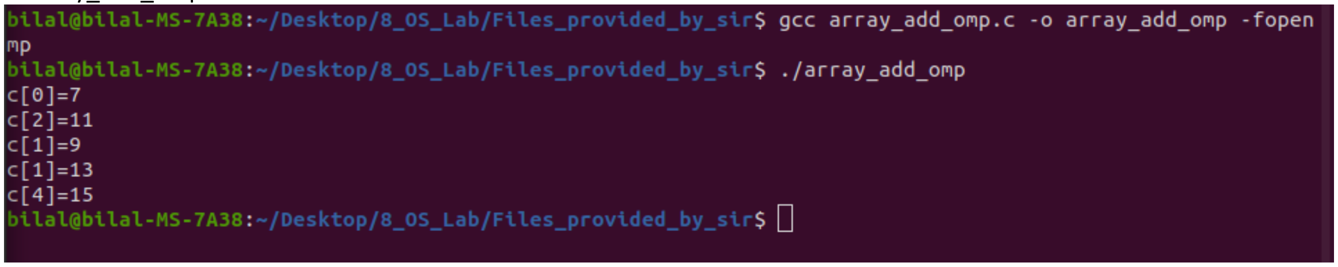
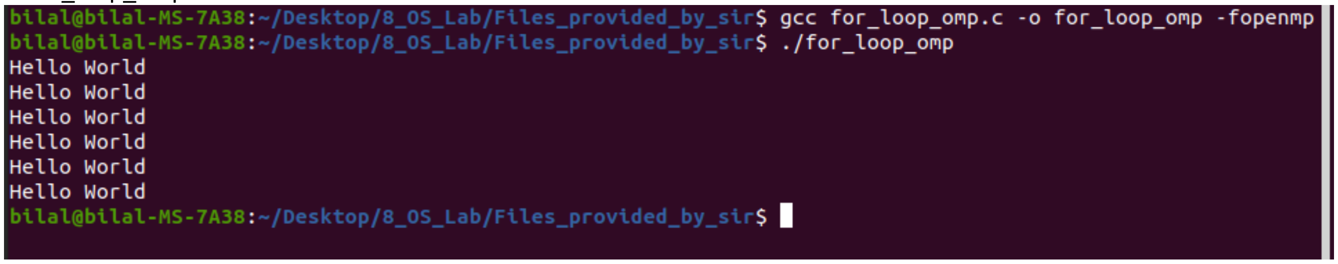
Task 01

1. array_add_omp



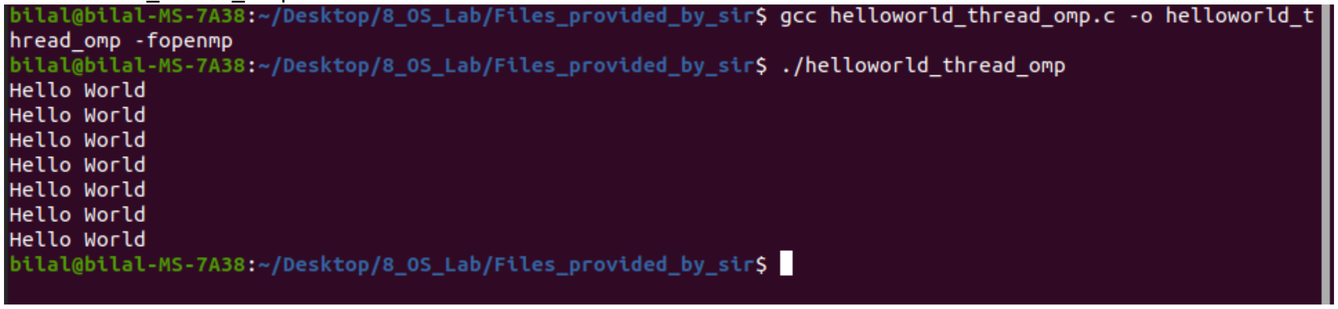
```
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ gcc array_add_omp.c -o array_add_omp -fopen
mp
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ ./array_add_omp
c[0]=7
c[2]=11
c[1]=9
c[1]=13
c[4]=15
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ 
```

2. for_loop_omp



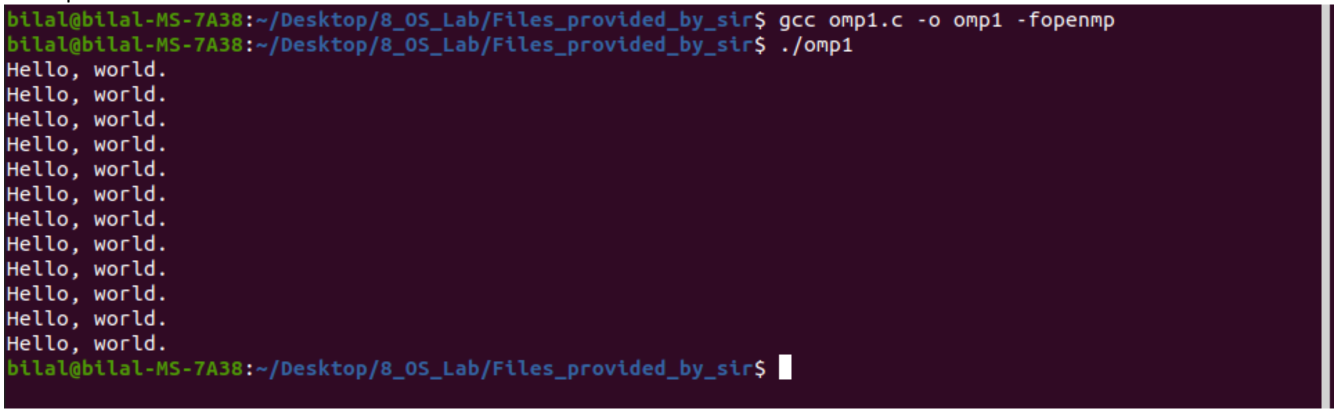
```
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ gcc for_loop_omp.c -o for_loop_omp -fopenmp
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ ./for_loop_omp
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ 
```

3. helloworld_thread_omp



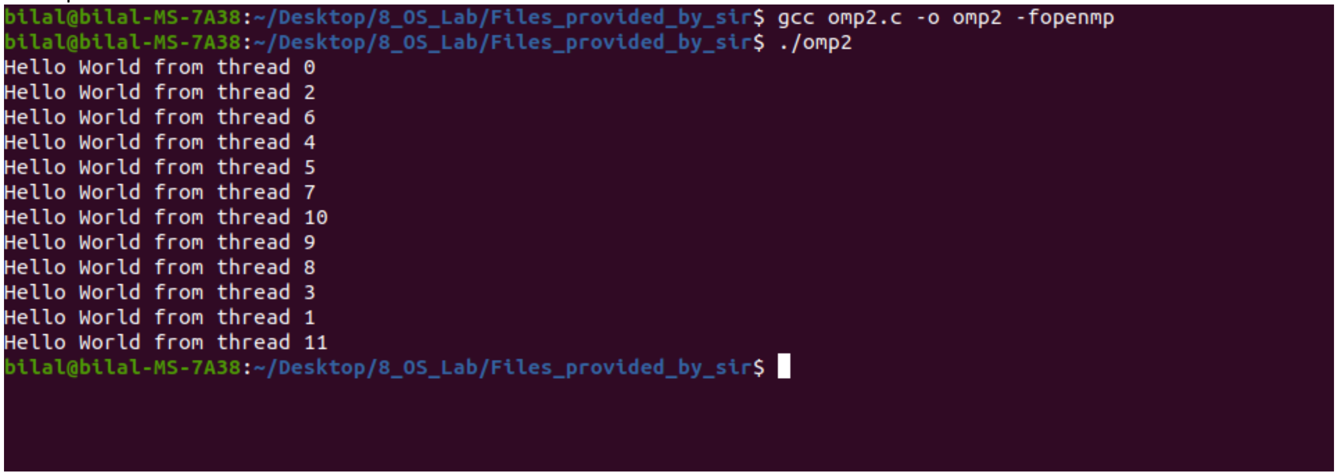
```
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ gcc helloworld_thread_omp.c -o helloworld_t
hread_omp -fopenmp
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ ./helloworld_thread_omp
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ 
```

4. omp1



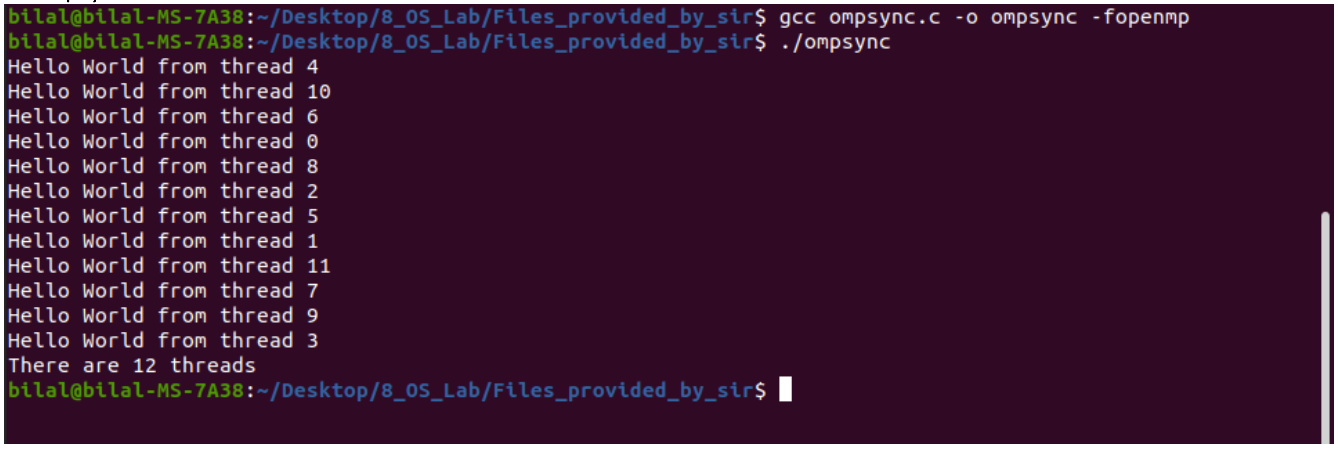
```
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ gcc omp1.c -o omp1 -fopenmp
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ ./omp1
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ 
```

5. omp2



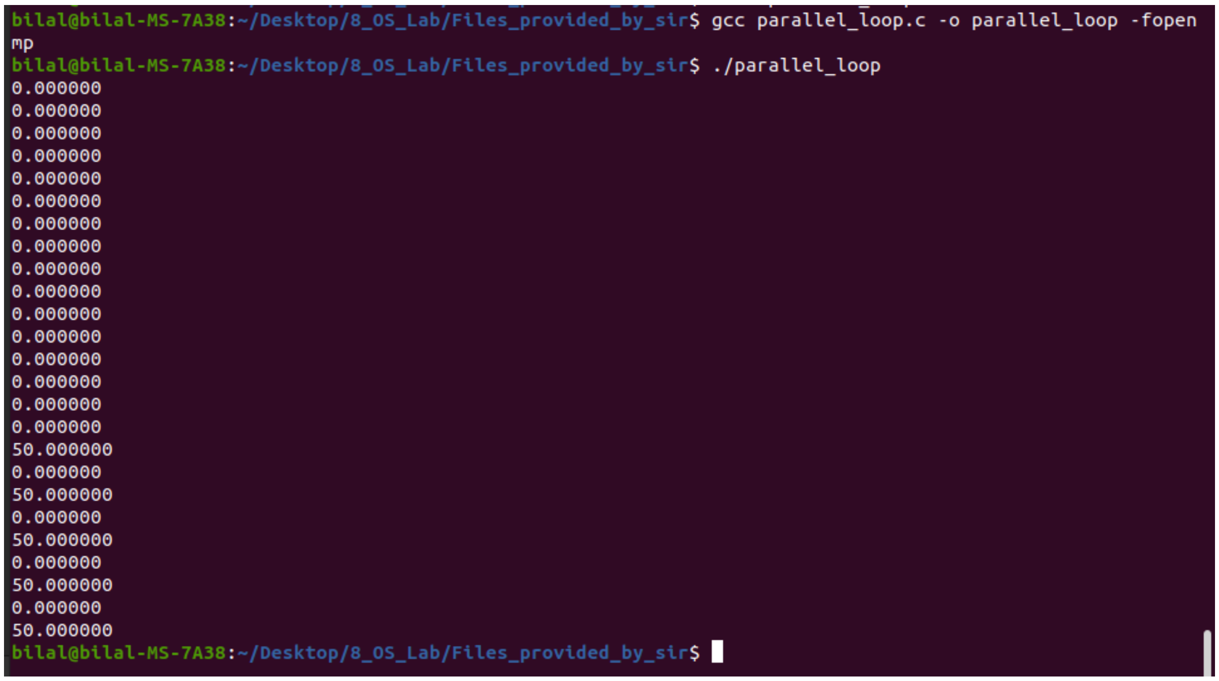
```
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ gcc omp2.c -o omp2 -fopenmp
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ ./omp2
Hello World from thread 0
Hello World from thread 2
Hello World from thread 6
Hello World from thread 4
Hello World from thread 5
Hello World from thread 7
Hello World from thread 10
Hello World from thread 9
Hello World from thread 8
Hello World from thread 3
Hello World from thread 1
Hello World from thread 11
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ 
```

6. ompsync



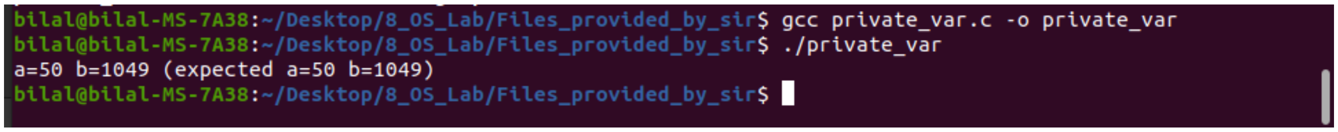
```
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ gcc ompsync.c -o ompsync -fopenmp
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ ./ompsync
Hello World from thread 4
Hello World from thread 10
Hello World from thread 6
Hello World from thread 0
Hello World from thread 8
Hello World from thread 2
Hello World from thread 5
Hello World from thread 1
Hello World from thread 11
Hello World from thread 7
Hello World from thread 9
Hello World from thread 3
There are 12 threads
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ 
```

7. parallel_loop

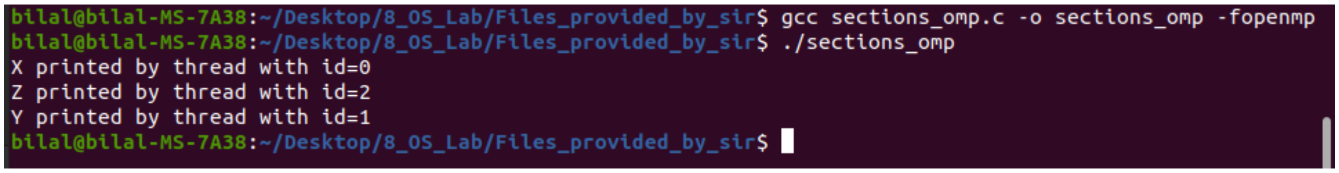


```
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ gcc parallel_loop.c -o parallel_loop -fopen
mp
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ ./parallel_loop
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
0.000000
50.000000
0.000000
50.000000
0.000000
50.000000
0.000000
50.000000
0.000000
50.000000
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ 
```

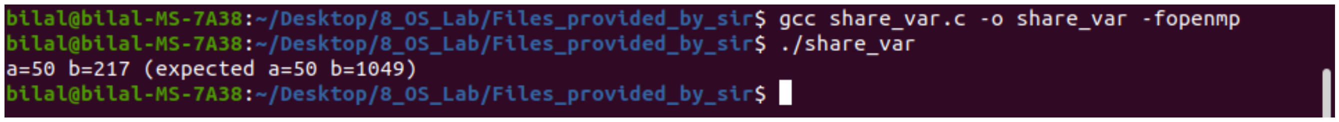
8. private_var


```
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ gcc private_var.c -o private_var
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ ./private_var
a=50 b=1049 (expected a=50 b=1049)
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ 
```

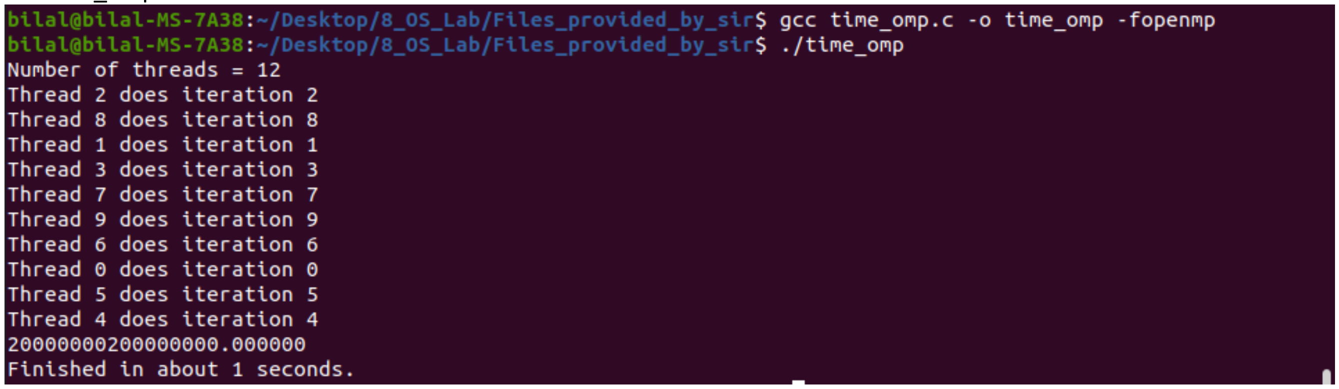
9. sections_omp


```
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ gcc sections_omp.c -o sections_omp -fopenmp
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ ./sections_omp
X printed by thread with id=0
Z printed by thread with id=2
Y printed by thread with id=1
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ 
```

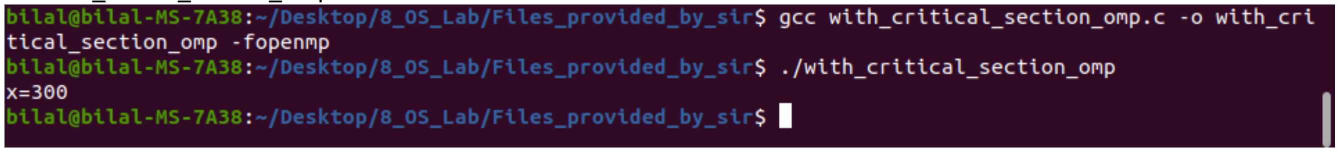
10. share_var


```
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ gcc share_var.c -o share_var -fopenmp
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ ./share_var
a=50 b=217 (expected a=50 b=1049)
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ 
```

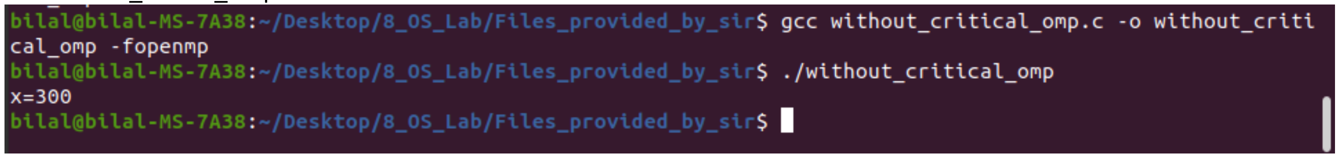
11. time_omp


```
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ gcc time_omp.c -o time_omp -fopenmp
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ ./time_omp
Number of threads = 12
Thread 2 does iteration 2
Thread 8 does iteration 8
Thread 1 does iteration 1
Thread 3 does iteration 3
Thread 7 does iteration 7
Thread 9 does iteration 9
Thread 6 does iteration 6
Thread 0 does iteration 0
Thread 5 does iteration 5
Thread 4 does iteration 4
20000000200000000.000000
Finished in about 1 seconds.
```

12. with_critical_section_omp


```
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ gcc with_critical_section_omp.c -o with_cri
tical_section_omp -fopenmp
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ ./with_critical_section_omp
x=300
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ 
```

13. without_critical_omp


```
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ gcc without_critical_omp.c -o without_criti
cal_omp -fopenmp
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ ./without_critical_omp
x=300
bilal@bilal-MS-7A38:~/Desktop/8_OS_Lab/Files_provided_by_sir$ 
```

Task 02: part(a)

```
#include<omp.h>
#include<stdio.h>
#include<time.h>
#include<stdlib.h>
#define N 2000000000
// 10 arab iterations
int main()
{
clock_t t;
time_t start, stop;
printf("This file will check the time period without openmp.\nWe are taking the value of
N=2000000000.\n");
float sum=0;
t=clock();
time(&start);
for (int i = 1; i < N; i++)
{
sum=1/(float)i+sum;
// printf("1/i= %f\nsum= %f.\n\n", (float)1/i,(float)sum);
}
printf("%.5f\n", sum);
time(&stop);
t=clock()-t;
double time_taken=((double)t)/CLOCKS_PER_SEC;
printf("It took %.0f seconds to perform this task without parallelism.\n", difftime(stop, start));
}
```

Task 02: part(b)

```
#include <omp.h>
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#define N 2000000000
// 10 arab iterations
int main()
{
clock_t t;
time_t start, stop;
long long int i;
printf("This file will check the time period without openmp.\nWe are taking the value of
N=2000000000.\n");
float sum = 0;
t = clock();
time(&start);
#pragma omp parallel for reduction(+:sum)
for ( i = 1; i < N; i++)
{
sum+=1/(float)i;
}
printf("%f\n",sum);
time(&stop);
t = clock() - t;
// double time_taken=((double)t)/CLOCKS_PER_SEC;
printf("It took %.0f seconds to perform this task with parallelism.\n", difftime(stop, start));
}
```

Task 03: part(a)

```
#include <stdio.h>
#include <omp.h>
#include <stdlib.h>
#include<time.h>
#define N 835
void printmatrix(int Matrix[][N])
{
for (int i = 0; i < N; i++)
{
for (int j = 0; j < N; j++)
{
printf("%d ", Matrix[i][j]);
}
printf("\n");
}
}
void Assign_values(int Matrix[][N])
{
for (int i = 0; i < N; i++)
{
for (int j = 0; j < N; j++)
{
Matrix[i][j] = rand() % N;
}
}
}
```

```
void Add_matrices(int Matrix01[][N], int Matrix02[][N], int Ans_Matrix[][N])
{
for (int i = 0; i < N; i++)
{
for (int j = 0; j < N; j++)
{
Ans_Matrix[i][j] = Matrix01[i][j] + Matrix02[i][j];
}
}
}

int main()
{
time_t start, end;
time(&start);
printf("Adding 2 NxN matrices without parallelism.\n");
int matrix_01[N][N] = {}, matrix_02[N][N] = {}, matrix_ans[N][N] = {};
// Assigning random values to both matrices
Assign_values(matrix_01);
Assign_values(matrix_02);

// displayig both matrices
// printf("This is matrix 01\n");
// printmatrix(matrix_01);
// printf("\nThis is matrix 02\n");
// printmatrix(matrix_02);

// now adding both matrices
Add_matrices(matrix_01,matrix_02,matrix_ans);
printf("\nAfter adding matrix 01 and matrix 02 we get:\n");
printmatrix(matrix_ans);
time(&end);
printf("It took %0.f seconds to add 2 matrices of 835x835 without parallelism.\n",difftime(end,start));

}
```

Task 03: part(b)

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

```
#include<time.h>
#define N 835
void printmatrix(int Matrix[][N])
{
for (int i = 0; i < N; i++)
{
for (int j = 0; j < N; j++)
{
printf("%d ", Matrix[i][j]);
}
printf("\n");
}
}
void Assign_values(int Matrix[][N])
{
for (int i = 0; i < N; i++)
{
for (int j = 0; j < N; j++)
{
Matrix[i][j] = rand() % N;
}
}
}

void Add_matrices(int Matrix01[][N], int Matrix02[][N], int Ans_Matrix[][N])
{
#pragma omp parallel for
for (int i = 0; i < N; i++)
{
for (int j = 0; j < N; j++)
{
Ans_Matrix[i][j] = Matrix01[i][j] + Matrix02[i][j];
}
}
}

int main()
{
time_t start, end;
time(&start);
printf("Adding 2 NxN matrices without parallelism.\n");
int matrix_01[N][N] = {}, matrix_02[N][N] = {}, matrix_ans[N][N] = {};
// Assigning random values to both matrices
Assign_values(matrix_01);
Assign_values(matrix_02);
// displayig both matrices
// printf("This is matrix 01\n");
// printmatrix(matrix_01);
// printf("\nThis is matrix 02\n");
// printmatrix(matrix_02);

// now adding both matrices
Add_matrices(matrix_01,matrix_02,matrix_ans);
printf("\nAfter adding matrix 01 and matrix 02 we get:\n");
printmatrix(matrix_ans);
time(&end);
printf("It took %0.f seconds to add 2 matrices of 835x835 with parallelism.\n",difftime(end,start));

}
```