


NAME : MUHAMMAD HAMZA

SEC : BS-SE-4A

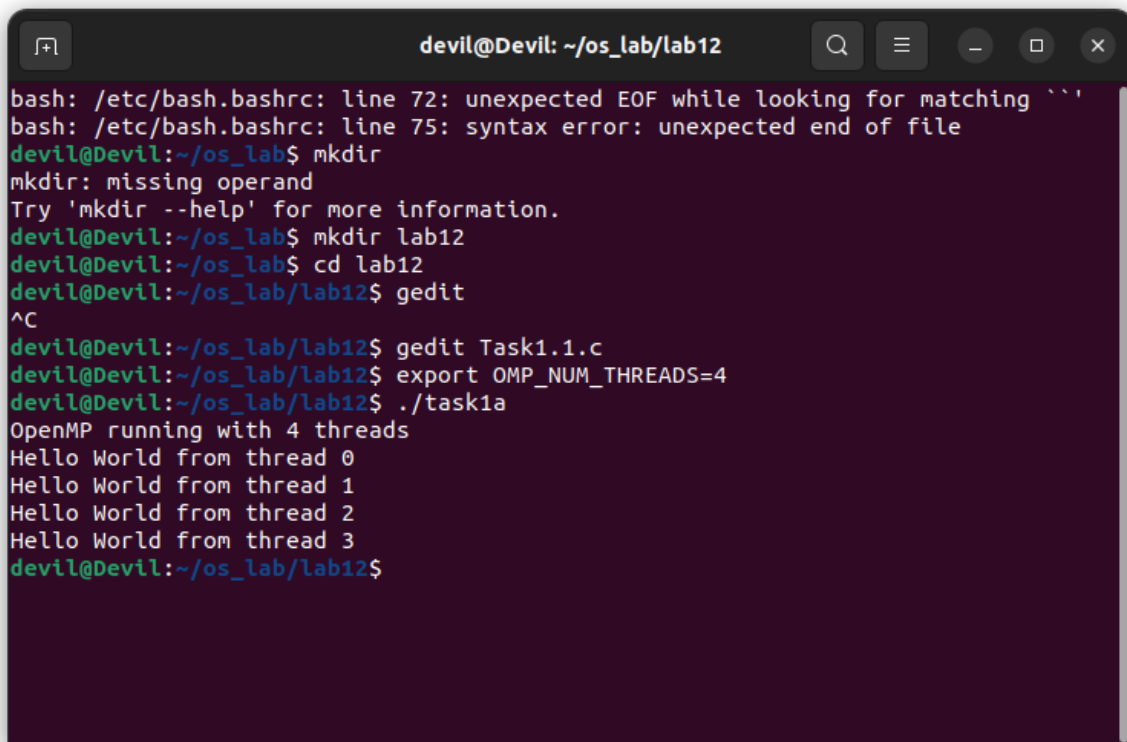
ROLL NO : 21K-3815

TASK 1

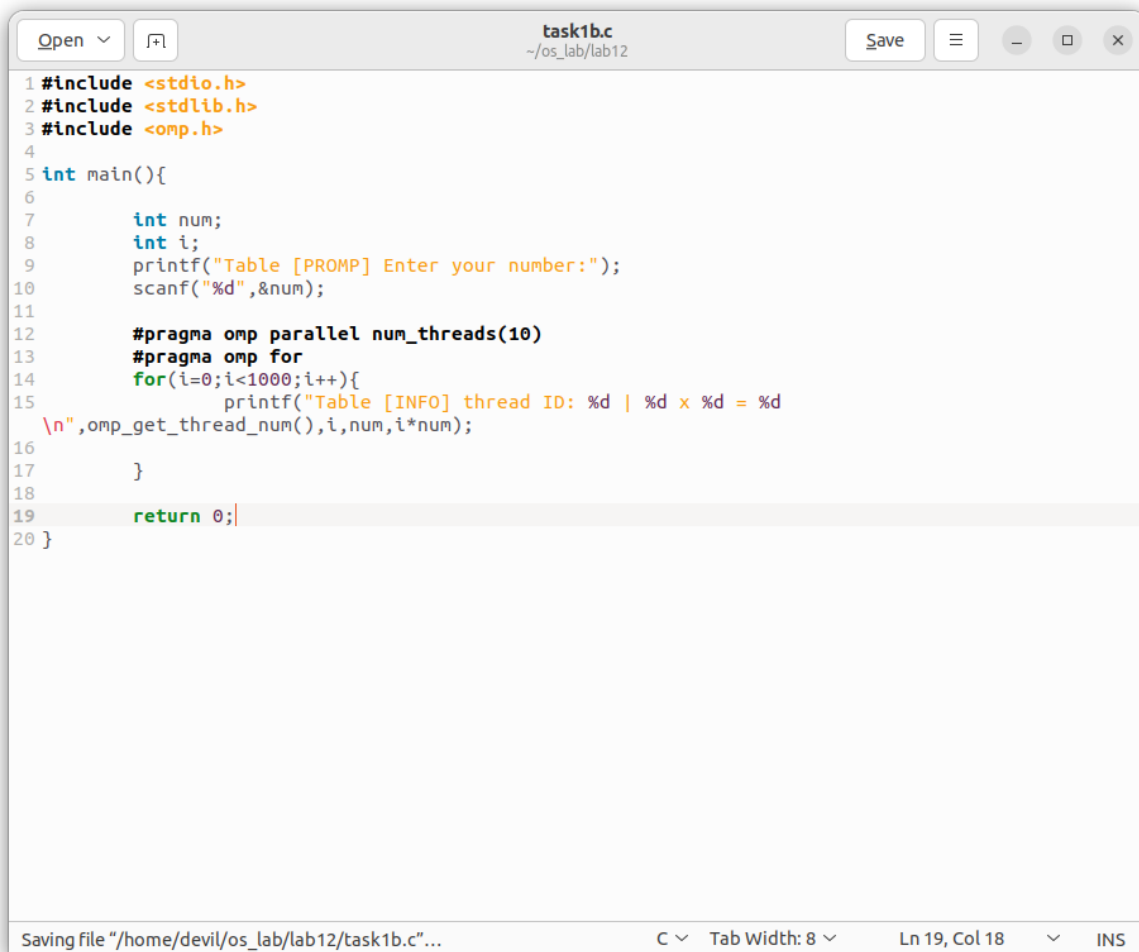


```
1 #include <stdio.h>
2 #include <omp.h>
3
4 int main(int argc, char *argv[]){
5     printf("OpenMP running with %d threads \n", omp_get_max_threads());
6
7     #pragma omp parallel
8     {
9         printf("Hello World from thread %d \n", omp_get_thread_num());
10    }
11 }
```

C Tab Width: 8 Ln 1, Col 1 INS



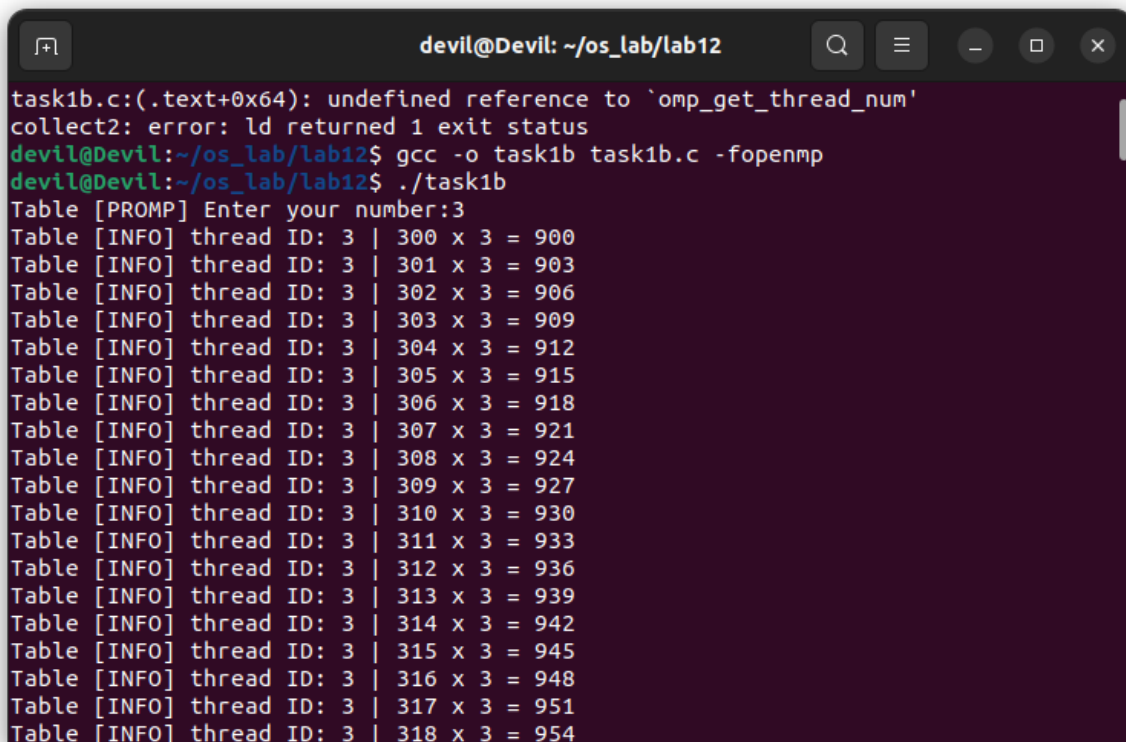
```
bash: /etc/bash.bashrc: line 72: unexpected EOF while looking for matching `''
bash: /etc/bash.bashrc: line 75: syntax error: unexpected end of file
devil@Devil:~/os_lab$ mkdir
mkdir: missing operand
Try 'mkdir --help' for more information.
devil@Devil:~/os_lab$ mkdir lab12
devil@Devil:~/os_lab$ cd lab12
devil@Devil:~/os_lab/lab12$ gedit
^C
devil@Devil:~/os_lab/lab12$ gedit Task1.1.c
devil@Devil:~/os_lab/lab12$ export OMP_NUM_THREADS=4
devil@Devil:~/os_lab/lab12$ ./task1a
OpenMP running with 4 threads
Hello World from thread 0
Hello World from thread 1
Hello World from thread 2
Hello World from thread 3
devil@Devil:~/os_lab/lab12$
```



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <omp.h>
4
5 int main(){
6
7     int num;
8     int i;
9     printf("Table [PROMP] Enter your number:");
10    scanf("%d",&num);
11
12    #pragma omp parallel num_threads(10)
13    #pragma omp for
14    for(i=0;i<1000;i++){
15        printf("Table [INFO] thread ID: %d | %d x %d = %d\n",omp_get_thread_num(),i,num,i*num);
16    }
17
18    return 0;
19 }
20 }
```

Saving file "/home/devil/os_lab/lab12/task1b.c"...

C Tab Width: 8 Ln 19, Col 18 INS



```
task1b.c:(.text+0x64): undefined reference to `omp_get_thread_num'
collect2: error: ld returned 1 exit status
devil@Devil:~/os_lab/lab12$ gcc -o task1b task1b.c -fopenmp
devil@Devil:~/os_lab/lab12$ ./task1b
Table [PROMP] Enter your number:3
Table [INFO] thread ID: 3 | 300 x 3 = 900
Table [INFO] thread ID: 3 | 301 x 3 = 903
Table [INFO] thread ID: 3 | 302 x 3 = 906
Table [INFO] thread ID: 3 | 303 x 3 = 909
Table [INFO] thread ID: 3 | 304 x 3 = 912
Table [INFO] thread ID: 3 | 305 x 3 = 915
Table [INFO] thread ID: 3 | 306 x 3 = 918
Table [INFO] thread ID: 3 | 307 x 3 = 921
Table [INFO] thread ID: 3 | 308 x 3 = 924
Table [INFO] thread ID: 3 | 309 x 3 = 927
Table [INFO] thread ID: 3 | 310 x 3 = 930
Table [INFO] thread ID: 3 | 311 x 3 = 933
Table [INFO] thread ID: 3 | 312 x 3 = 936
Table [INFO] thread ID: 3 | 313 x 3 = 939
Table [INFO] thread ID: 3 | 314 x 3 = 942
Table [INFO] thread ID: 3 | 315 x 3 = 945
Table [INFO] thread ID: 3 | 316 x 3 = 948
Table [INFO] thread ID: 3 | 317 x 3 = 951
Table [INFO] thread ID: 3 | 318 x 3 = 954
```

```

1 #include <stdio.h>
2 #include <omp.h>
3 int main(){
4
5     int i;
6     const int N = 1000;
7     int a=50;
8     int b=0;
9
10    #pragma omp parallel for default(shared)
11
12    for(i=0;i<N;i++){
13        b=a+i;
14    }
15
16    printf("a=%d b=%d (expected a=50 b=1049)\n",a,b);
17 }

```

Saving file "/home/devil/os_lab/lab12/task1c.c"...

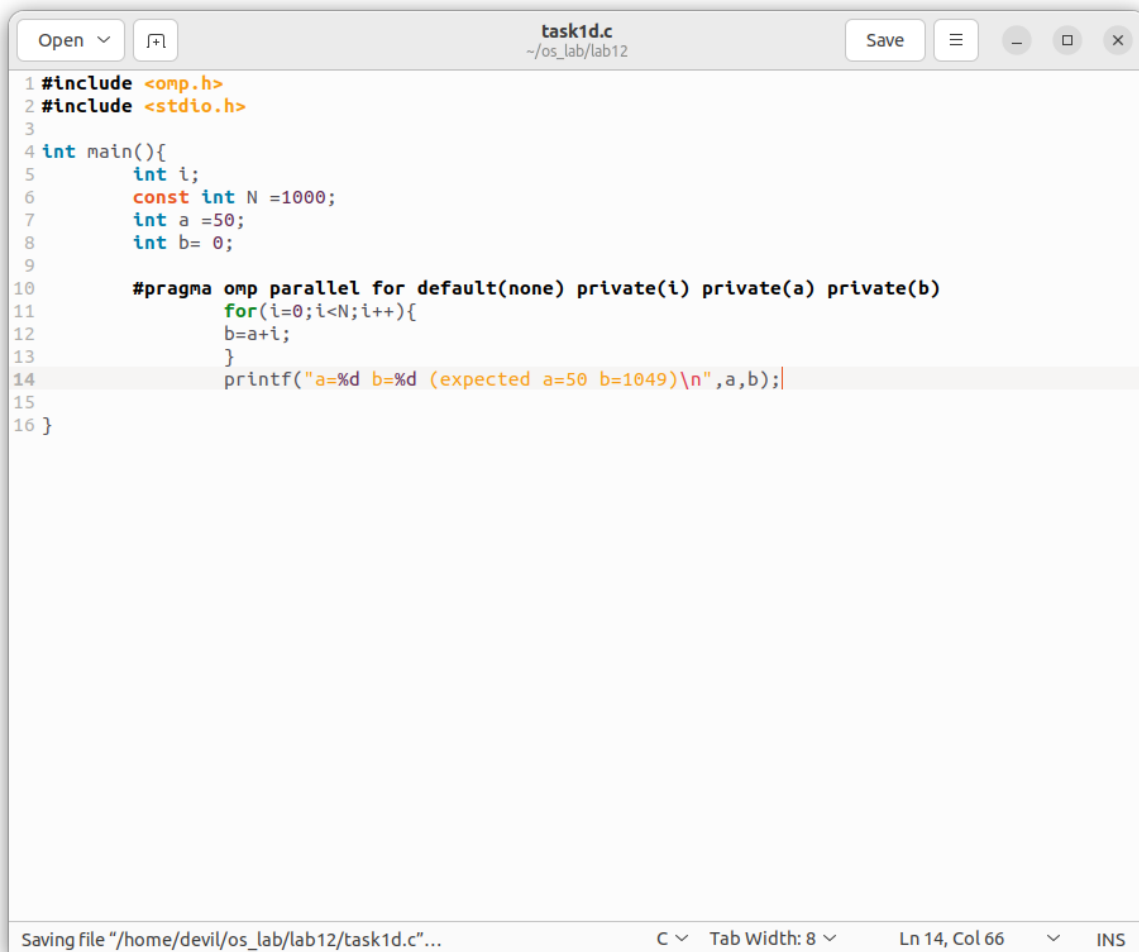
C Tab Width: 8 Ln 15, Col 9 INS

```

16 |         printf("a=%d b=%d (expected a=50 b=1049)\n",a,b);
task1c.c:16:53: error: unknown type name 'a'
16 |         printf("a=%d b=%d (expected a=50 b=1049)\n",a,b);
task1c.c:16:55: error: unknown type name 'b'
16 |         printf("a=%d b=%d (expected a=50 b=1049)\n",a,b);
task1c.c:17:2: error: expected identifier or '(' before '}' token
17 |     }
    |     ^

devil@Devil:~/os_lab/lab12$ gedit task1c.c
devil@Devil:~/os_lab/lab12$ gcc -o task1c task1c.c -fopenmp
devil@Devil:~/os_lab/lab12$ expor OMP_NUM_THREADS=1
Command 'expor' not found, did you mean:
  command 'expr' from deb coreutils (8.32-4.1ubuntu1)
Try: sudo apt install <deb name>
devil@Devil:~/os_lab/lab12$ export OMP_NUM_THREADS=1
devil@Devil:~/os_lab/lab12$ ./task1c
a=50 b=1049 (expected a=50 b=1049)
devil@Devil:~/os_lab/lab12$ export OMP_NUM_THREADS=4
devil@Devil:~/os_lab/lab12$ ./task1c
a=50 b=299 (expected a=50 b=1049)
devil@Devil:~/os_lab/lab12$

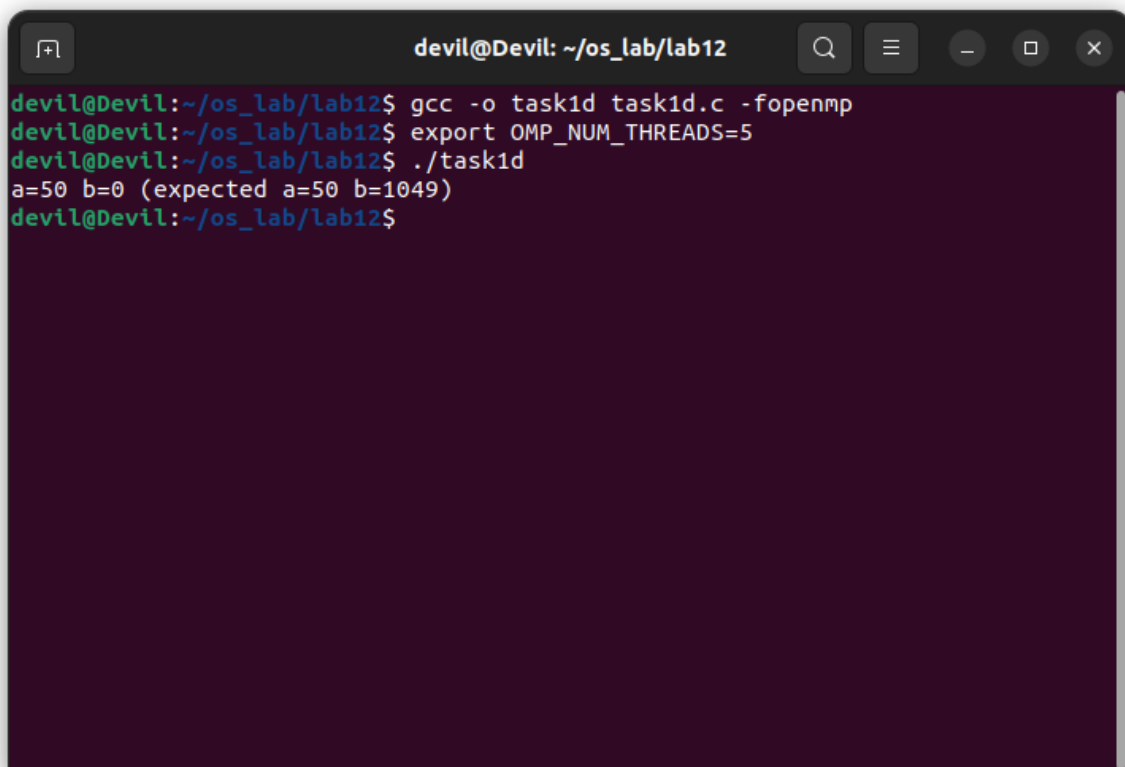
```



```
1 #include <omp.h>
2 #include <stdio.h>
3
4 int main(){
5     int i;
6     const int N =1000;
7     int a =50;
8     int b= 0;
9
10    #pragma omp parallel for default(none) private(i) private(a) private(b)
11        for(i=0;i<N;i++){
12            b=a+i;
13        }
14    printf("a=%d b=%d (expected a=50 b=1049)\n",a,b);
15
16 }
```

Saving file "/home/devil/os_lab/lab12/task1d.c"...

C Tab Width: 8 Ln 14, Col 66 INS



```
devil@Devil: ~/os_lab/lab12
devil@Devil:~/os_lab/lab12$ gcc -o task1d task1d.c -fopenmp
devil@Devil:~/os_lab/lab12$ export OMP_NUM_THREADS=5
devil@Devil:~/os_lab/lab12$ ./task1d
a=50 b=0 (expected a=50 b=1049)
devil@Devil:~/os_lab/lab12$
```

```

1 #include <omp.h>
2 #include <stdio.h>
3
4 int main(){
5
6     int i;
7     const int N =1000;
8     int sum=0;
9
10    #pragma omp parallel for private(i) reduction(+:sum)
11    for(i=0;i<N;i++){
12        sum+=i;
13    }
14    printf("Reduction sum=%d (expected %d)\n",sum,((N-1)*N)/2);
15 }

```

Saving file "/home/devil/os_lab/lab12/task1e.c"...

C Tab Width: 8 Ln 14, Col 60 INS

```

devil@Devil:~/os_lab/lab12$ gcc -o task1d task1d.c -fopenmp
devil@Devil:~/os_lab/lab12$ export OMP_NUM_THREADS=5
devil@Devil:~/os_lab/lab12$ ./task1d
a=50 b=0 (expected a=50 b=1049)
devil@Devil:~/os_lab/lab12$ gedit task1e.c
devil@Devil:~/os_lab/lab12$ gcc -o task1e task1e.c -fopenmp
devil@Devil:~/os_lab/lab12$ export OMP_NUM_THREADS=4
devil@Devil:~/os_lab/lab12$ ./task1e
Reduction sum=499500 (expected 499500)
devil@Devil:~/os_lab/lab12$

```

TASK 2

```
Open task2.c Save
1 #include <stdio.h>
2 #include <time.h>
3
4 double seriesSumSerial(int n) {
5     double sum = 0.0;
6     for (int i = 1; i <= n; i++) {
7         sum += 1.0 / i;
8     }
9     return sum;
10 }
11
12 double seriesSumParallel(int n) {
13     double sum = 0.0;
14     #pragma omp parallel for reduction(+:sum)
15     for (int i = 1; i <= n; i++) {
16         sum += 1.0 / i;
17     }
18     return sum;
19 }
20
21 int main() {
22     int n;
23     double sum;
24     clock_t start, end;
25     double cpu_time_used;
26
27     printf("Enter the value for n:");
28     scanf("%d", &n);
29
30     start = clock();
31     sum = seriesSumSerial(n);
32     end = clock();
33     cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
34     printf("Sum without OpenMP: %f\n", sum);
35     printf("Time without OpenMP: %f seconds\n", cpu_time_used);
36
37     start = clock();
38     sum = seriesSumParallel(n);
39     end = clock();
40     cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
41     printf("Sum with OpenMP: %f\n", sum);
42     printf("Time with OpenMP: %f seconds\n", cpu_time_used);
43 }
44
45 #define N 100000000
46
```

```
devil@Devil: ~/os_lab/lab12
devil@Devil:~/os_lab/lab12$ gcc -o task1d task1d.c -fopenmp
devil@Devil:~/os_lab/lab12$ export OMP_NUM_THREADS=5
devil@Devil:~/os_lab/lab12$ ./task1d
a=50 b=0 (expected a=50 b=1049)
devil@Devil:~/os_lab/lab12$ gedit task1e.c
devil@Devil:~/os_lab/lab12$ gcc -o task1e task1e.c -fopenmp
devil@Devil:~/os_lab/lab12$ export OMP_NUM_THREADS=4
devil@Devil:~/os_lab/lab12$ ./task1e
Reduction sum=499500 (expected 499500)
devil@Devil:~/os_lab/lab12$ gedit task2.c
devil@Devil:~/os_lab/lab12$ gcc -o task2 task2.c -fopenmp
devil@Devil:~/os_lab/lab12$ export OMP_NUM_THREADS=4
devil@Devil:~/os_lab/lab12$ ./task2
Enter the value for n: 20
Sum without OpenMP: 3.597740
Time without OpenMP: 0.000002 seconds

Sum with OpenMP: 3.597740
Time with OpenMP: 0.000920 seconds
devil@Devil:~/os_lab/lab12$
```

TASK 3

```
task3.c
~/os_lab/lab12

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4
5 void matrixAddition(int matrixA[][10], int matrixB[][10], int result[][10], int rows, int cols) {
6     for (int i = 0; i < rows; i++) {
7         for (int j = 0; j < cols; j++) {
8             result[i][j] = matrixA[i][j] + matrixB[i][j];
9         }
10    }
11}
12
13 int main() {
14     int rows = 10;
15     int cols = 10;
16
17     int matrixA[10][10];
18     int matrixB[10][10];
19     int result[10][10];
20
21
22     for (int i = 0; i < rows; i++) {
23         for (int j = 0; j < cols; j++) {
24             matrixA[i][j] = rand() % 100;
25             matrixB[i][j] = rand() % 100;
26         }
27     }
28
29     clock_t start, end;
30     double cpu_time_used;
31     start = clock();
32     matrixAddition(matrixA, matrixB, result, rows, cols);
33     end = clock();
34     cpu_time_used = ((double)(end - start)) / CLOCKS_PER_SEC;
35     printf("Matrix addition without OpenMP\n");
36     for (int i = 0; i < rows; i++) {
37         for (int j = 0; j < cols; j++) {
38             printf("%d", result[i][j]);
39         }
40         printf("\n");
41     }
42     printf("Time: %f seconds\n", cpu_time_used);
43
44
45     start = clock();
46     #pragma omp parallel for
47     for (int i = 0; i < rows; i++) {
48         for (int j = 0; j < cols; j++) {
49             result[i][j] = matrixA[i][j] + matrixB[i][j];
50         }
51     }
52     end = clock();
53     cpu_time_used = ((double)(end - start)) / CLOCKS_PER_SEC;
54     printf("Matrix addition with OpenMP\n");
55     for (int i = 0; i < rows; i++) {
56         for (int j = 0; j < cols; j++) {
57             printf("%d", result[i][j]);
58         }
59         printf("\n");
60     }
61     printf("Time: %f seconds\n", cpu_time_used);
62
63     return 0;
64 }
65
```

```
task3.c
~/os_lab/lab12

17 int matrixA[10][10];
18 int matrixB[10][10];
19 int result[10][10];
20
21
22 for (int i = 0; i < rows; i++) {
23     for (int j = 0; j < cols; j++) {
24         matrixA[i][j] = rand() % 100;
25         matrixB[i][j] = rand() % 100;
26     }
27 }
28
29 clock_t start, end;
30 double cpu_time_used;
31 start = clock();
32 matrixAddition(matrixA, matrixB, result, rows, cols);
33 end = clock();
34 cpu_time_used = ((double)(end - start)) / CLOCKS_PER_SEC;
35 printf("Matrix addition without OpenMP\n");
36 for (int i = 0; i < rows; i++) {
37     for (int j = 0; j < cols; j++) {
38         printf("%d", result[i][j]);
39     }
40     printf("\n");
41 }
42 printf("Time: %f seconds\n", cpu_time_used);
43
44
45 start = clock();
46 #pragma omp parallel for
47 for (int i = 0; i < rows; i++) {
48     for (int j = 0; j < cols; j++) {
49         result[i][j] = matrixA[i][j] + matrixB[i][j];
50     }
51 }
52 end = clock();
53 cpu_time_used = ((double)(end - start)) / CLOCKS_PER_SEC;
54 printf("Matrix addition with OpenMP\n");
55 for (int i = 0; i < rows; i++) {
56     for (int j = 0; j < cols; j++) {
57         printf("%d", result[i][j]);
58     }
59     printf("\n");
60 }
61 printf("Time: %f seconds\n", cpu_time_used);
62
63 return 0;
64 }
65
```



```
devll@Devll: ~/os_lab/lab12
devll@Devll:~/os_lab/lab12$ export OMP_NUM_THREADS=4
devll@Devll:~/os_lab/lab12$ ./task3
Matrix addition without OpenMP
169 92 128 178 70 89 149 89 66 108
79 96 112 85 102 31 80 136 149 53
102 40 121 122 85 39 171 129 132 177
30 111 41 75 70 119 127 81 98 93
95 154 172 54 153 92 144 51 112 133
165 112 68 99 169 108 81 127 154 63
69 48 168 156 36 123 150 151 89 43
99 61 22 138 23 89 119 63 97 76
167 92 91 93 98 38 143 134 92 95
83 65 82 95 153 66 128 46 54 57
Time: 0.000002 seconds

Matrix addition with OpenMP
169 92 128 178 70 89 149 89 66 108
79 96 112 85 102 31 80 136 149 53
102 40 121 122 85 39 171 129 132 177
30 111 41 75 70 119 127 81 98 93
95 154 172 54 153 92 144 51 112 133
165 112 68 99 169 108 81 127 154 63
69 48 168 156 36 123 150 151 89 43
99 61 22 138 23 89 119 63 97 76
167 92 91 93 98 38 143 134 92 95
83 65 82 95 153 66 128 46 54 57
Time: 0.030931 seconds
devll@Devll:~/os_lab/lab12$
```