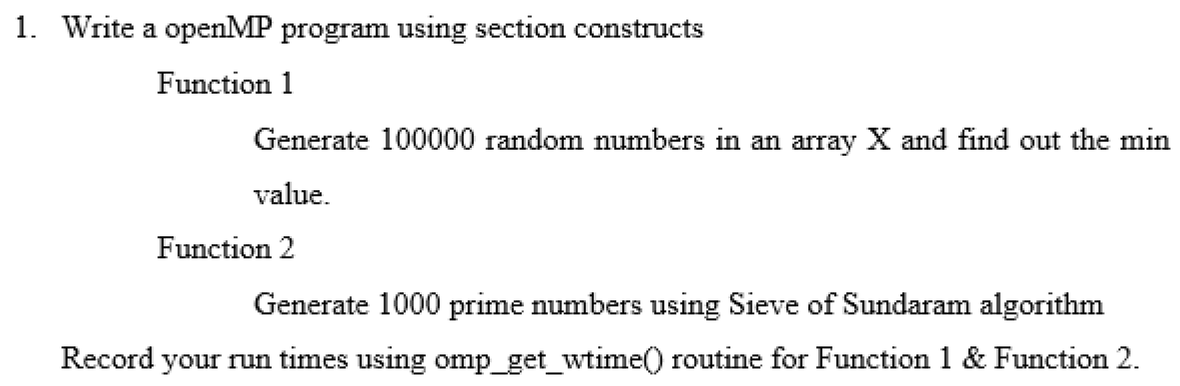
NAME – ANSH GOEL

REGISTER NO.- 20BCE1798

PARALLEL AND DISTRIBUTED COMPUTING

LAB – 4



1. Write a openMP program using section constructs

   Function 1

   Generate 100000 random numbers in an array X and find out the min value.

   Function 2

   Generate 1000 prime numbers using Sieve of Sundaram algorithm

   Record your run times using omp_get_wtime() routine for Function 1 & Function 2.

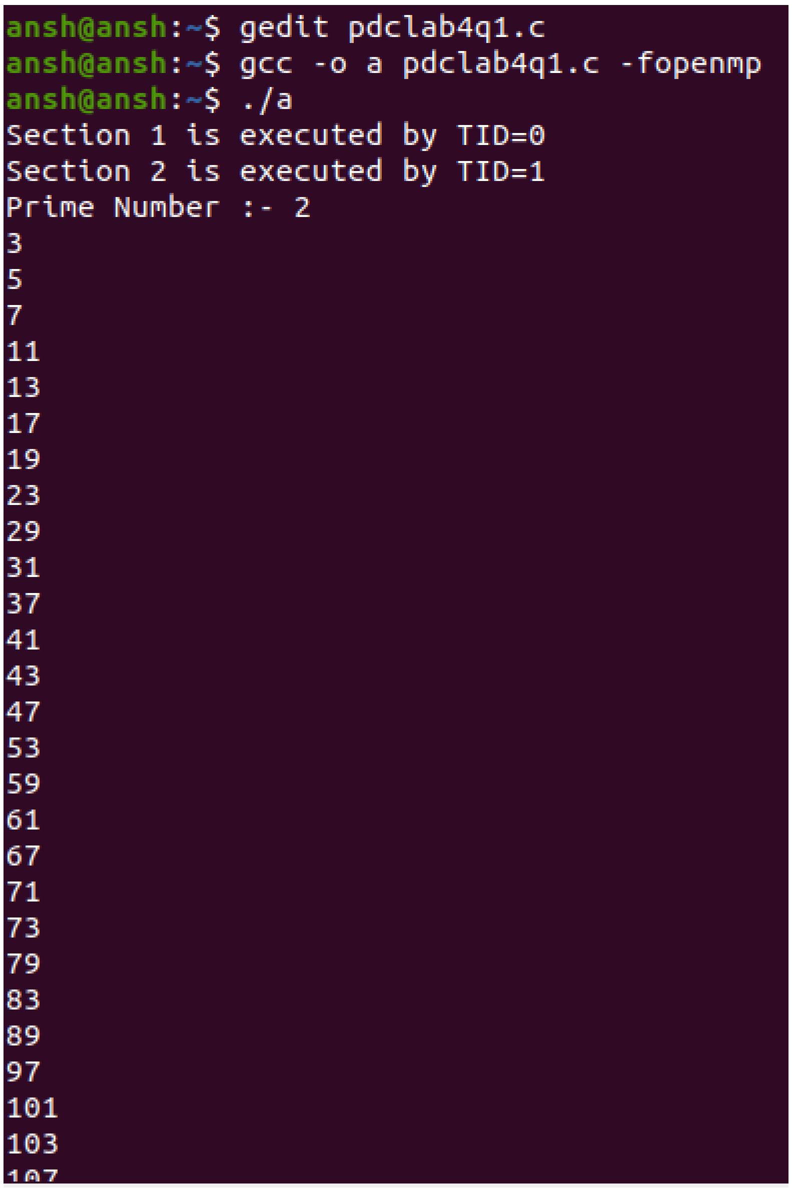
CODE:

```
#include<stdio.h>
#include<stdlib.h>
#include<omp.h>
void function1()
{
        int X[100000];
        for (int i = 0; i < 100000; i++){
                X[i] = rand();
        }
        int min = X[0];
        for (int i = 0; i < 100000; i++)
        {
                if (X[i] < min)
                        min = X[i];
        }
```

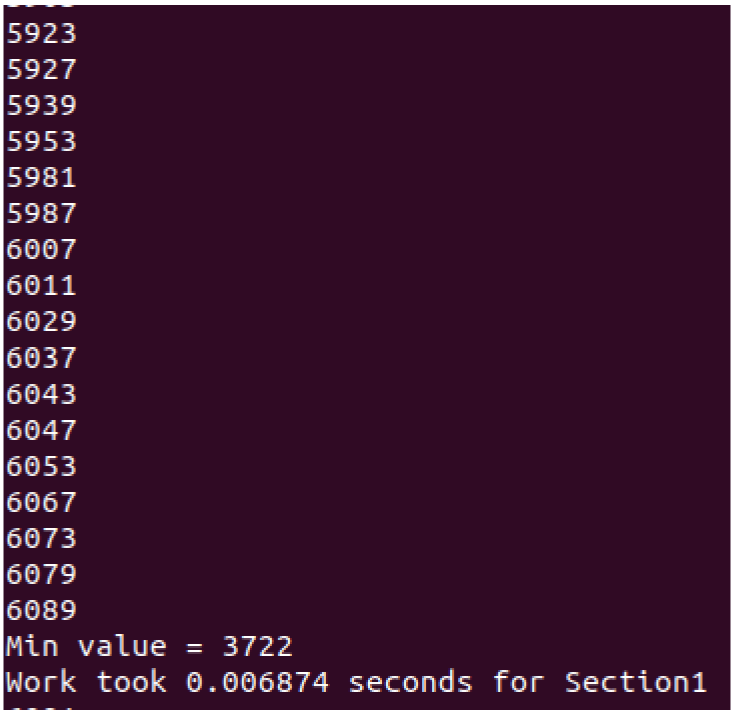
```
	printf("Min value = %d\n", min);
}
void function2()
{
	printf("Prime Number :- ");
	int number,i,j;
	number = 10000;
	int primes[number+1];
	//populating array with naturals numbers
	for(i = 2; i<=number; i++)
	primes[i] = i;
	i = 2;
	while ((i*i) <= number)
	{
	if (primes[i] != 0)
	{
		for(j=2; j<number; j++)
		{
			if (primes[i]*j > number)
				break;
			else
				// Instead of deleteing , making elemnets 0
				primes[primes[i]*j]=0;
		}
	}
	i++;
	}
	for(i = 2; i<=number; i++)
	{
	//If number is not 0 then it is prime
		if (primes[i]!=0)
		printf("%d\n",primes[i]);
	}
}
```

```
int main(int args, char *argv[])
{
        //omp_set_num_threads(3);
        #pragma omp parallel sections num_threads(2)
        {
                #pragma omp section
                {
                        double start;
                        double end;
                        start = omp_get_wtime();
                        printf("Section 1 is executed by TID=%d\n", omp_get_thread_num());
                        function1();
                        end = omp_get_wtime();
                        printf("Work took %f seconds for Section1\n", end - start);
                }
                #pragma omp section

                {
                        double start;
                        double end;
                        start = omp_get_wtime();
                        printf("Section 2 is executed by TID=%d\n", omp_get_thread_num());
                        function2();
                        end = omp_get_wtime();
                        printf("Work took %f seconds for Section2\n", end - start);
                }

        }
}
```

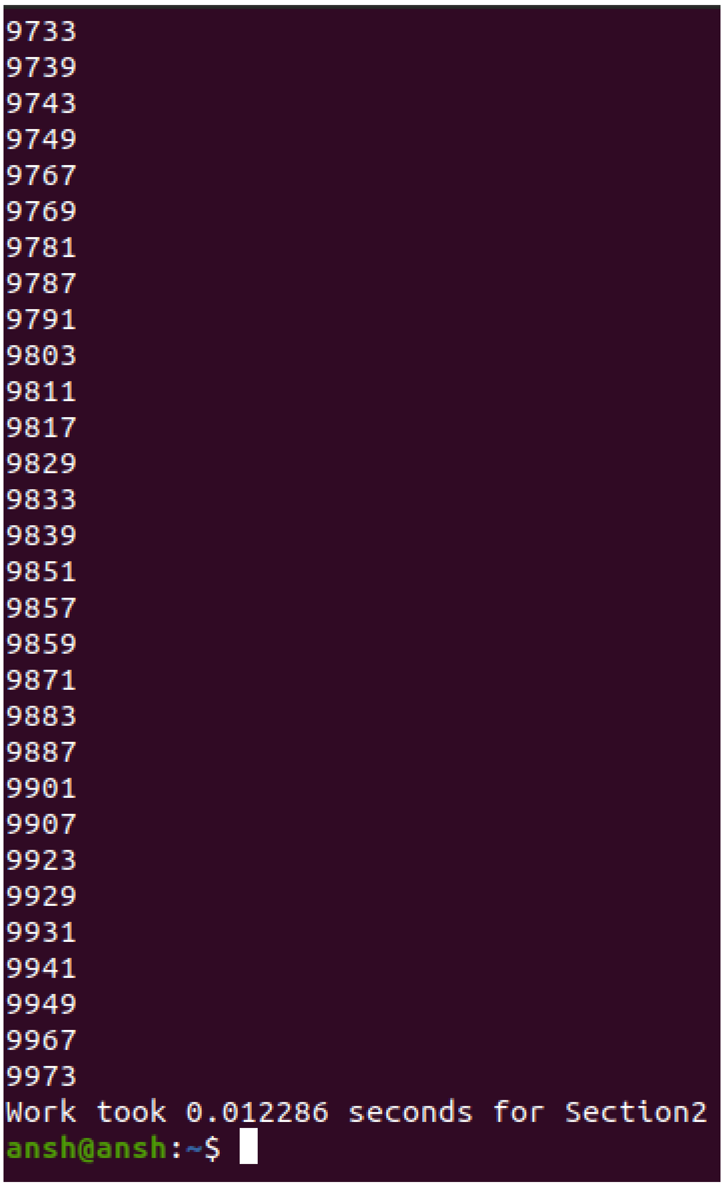
OUTPUT:



```
ansh@ansh:~$ gedit pdclab4q1.c
ansh@ansh:~$ gcc -o a pdclab4q1.c -fopenmp
ansh@ansh:~$ ./a
Section 1 is executed by TID=0
Section 2 is executed by TID=1
Prime Number :- 2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97
101
103
```



```
5923
5927
5939
5953
5981
5987
6007
6011
6029
6037
6043
6047
6053
6067
6073
6079
6089
Min value = 3722
Work took 0.006874 seconds for Section1
```



```
9733
9739
9743
9749
9767
9769
9781
9787
9791
9803
9811
9817
9829
9833
9839
9851
9857
9859
9871
9883
9887
9901
9907
9923
9929
9931
9941
9949
9967
9973
Work took 0.012286 seconds for Section2
ansh@ansh:~$
```

2. Write a multithreaded program using OpenMP for computing a matrix-matrix product for a large dimension. Use the OMP_NUM_THREADS environment variable to control the number of threads and plot the performance with varying numbers of threads (4,8 and 16). Consider four cases in which

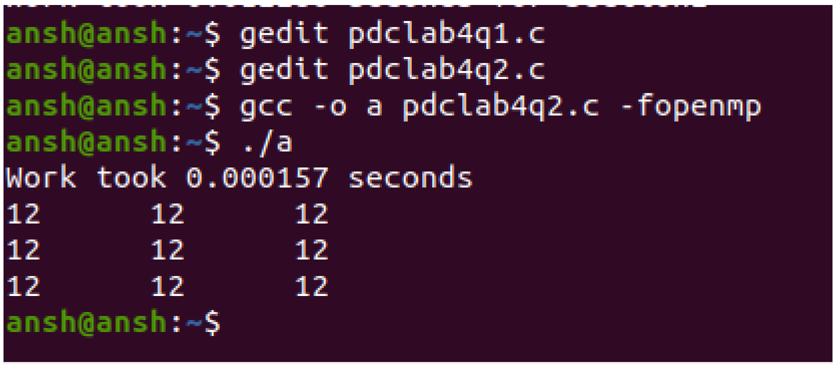
   i) Only the outermost loop is parallelized
   ii) The outer two loops are parallelized
   iii) All three loops are parallelized
   iv) Use collapse clause
   What is the observed result from these four cases?

   Record your run times using omp_get_wtime() routine.

CODE:

i)
```
#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <omp.h>

#define N 3

int A[N][N];

int B[N][N];

int C[N][N];

int main()

{

        int i,j,k;

        for (i= 0; i< N; i++)

        for (j= 0; j< N; j++)

                {

                A[i][j] = 2;

                B[i][j] = 2;

                }

        double start;
        double end;

        start = omp_get_wtime();

        #pragma omp parallel for private(i,j,k) shared(A,B,C)

        for (i = 0; i < N; ++i)

        {

        for (j = 0; j < N; ++j)

        {

                for (k = 0; k < N; ++k)

                {

                        C[i][j] += A[i][k] * B[k][j];

                }

        }

        }

        end = omp_get_wtime();

        printf("Work took %f seconds\n", end - start);

        for (i= 0; i< N; i++)

        {

        for (j= 0; j< N; j++)

        {

                printf("%d\t",C[i][j]);

        }

        printf("\n");

        }

        return 0;

}
```

OUTPUT:



```
ansh@ansh:~$ gedit pdclab4q1.c
ansh@ansh:~$ gedit pdclab4q2.c
ansh@ansh:~$ gcc -o a pdclab4q2.c -fopenmp
ansh@ansh:~$ ./a
Work took 0.000157 seconds
12      12      12
12      12      12
12      12      12
ansh@ansh:~$
```

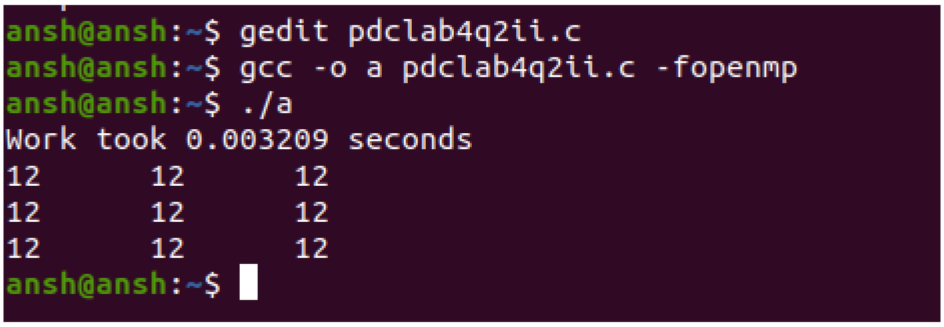
II)

CODE:

```
#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <omp.h>

#define N 3

int A[N][N];

int B[N][N];

int C[N][N];

int main()

{

        int i,j,k;

        for (i= 0; i< N; i++)

        for (j= 0; j< N; j++)

                {

                A[i][j] = 2;

                B[i][j] = 2;

                }
}

double start;

double end;

start = omp_get_wtime();

#pragma omp parallel for private(i,j,k) shared(A,B,C)

for (i = 0; i < N; ++i)

{

        #pragma omp parallel for private(j,k) shared(A,B,C)

for (j = 0; j < N; ++j)

{

        for (k = 0; k < N; ++k)

        {

                C[i][j] += A[i][k] * B[k][j];

        }

}

}

end = omp_get_wtime();

printf("Work took %f seconds\n", end - start);

for (i= 0; i< N; i++)

{

for (j= 0; j< N; j++)

{

        printf("%d\t",C[i][j]);

}

printf("\n");

}
```

```
            return 0;

}
```

OUTPUT:



```
ansh@ansh:~$ gedit pdclab4q2ii.c
ansh@ansh:~$ gcc -o a pdclab4q2ii.c -fopenmp
ansh@ansh:~$ ./a
Work took 0.003209 seconds
12      12      12
12      12      12
12      12      12
ansh@ansh:~$ 
```

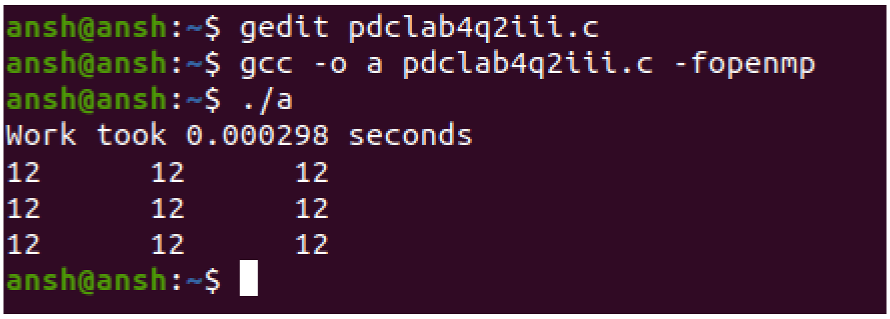
III)

CODE:

```
#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <omp.h>

#define N 3

int A[N][N];

int B[N][N];

int C[N][N];

int main()

{

            int i,j,k;

            for (i= 0; i< N; i++)

            for (j= 0; j< N; j++)

                        {

                        A[i][j] = 2;
```

```
        B[i][j] = 2;

        }

double start;

double end;

start = omp_get_wtime();

#pragma omp parallel for private(i,j,k) shared(A,B,C)

for (i = 0; i < N; ++i)

{

        #pragma omp parallel for private(j,k) shared(A,B,C)

for (j = 0; j < N; ++j)

{

        #pragma omp parallel for private(k) shared(A,B,C)

        for (k = 0; k < N; ++k)

        {

                C[i][j] += A[i][k] * B[k][j];

        }

}

}

end = omp_get_wtime();

printf("Work took %f seconds\n", end - start);

for (i= 0; i< N; i++)

{

for (j= 0; j< N; j++)

{

        printf("%d\t",C[i][j]);
        }

        printf("\n");

        }

        return 0;

}
```

OUTPUT:



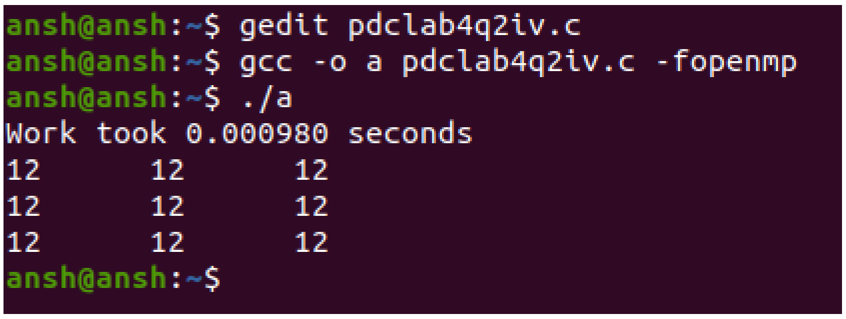
```
ansh@ansh:~$ gedit pdclab4q2iii.c
ansh@ansh:~$ gcc -o a pdclab4q2iii.c -fopenmp
ansh@ansh:~$ ./a
Work took 0.000298 seconds
12      12      12
12      12      12
12      12      12
ansh@ansh:~$
```

IV)

CODE:

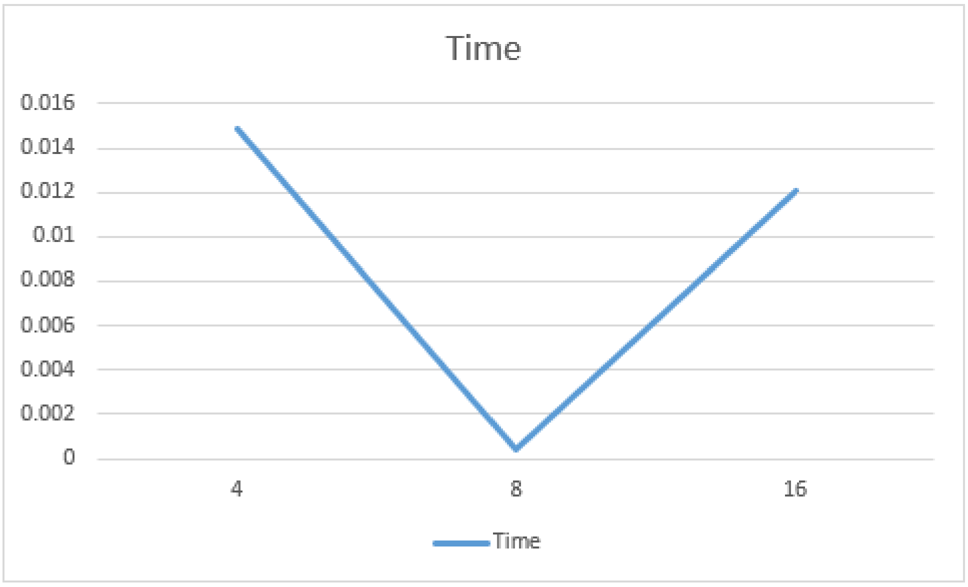
```
#include <stdio.h>

#include <stdlib.h>

#include <omp.h>

#define N 3

int A[N][N];

int B[N][N];

int C[N][N];

int main()

{

        int i,j,k;

        for (i= 0; i< N; i++)

        for (j= 0; j< N; j++)

                {

                A[i][j] = 2;

                B[i][j] = 2;

                }
        double start;
        double end;
        start = omp_get_wtime();
        #pragma omp parallel for private(i,j,k) shared(A,B,C) collapse(3)
        for (i = 0; i < N; ++i)
        {
        for (j = 0; j < N; ++j)
        {
                for (k = 0; k < N; ++k)
                {
                        C[i][j] += A[i][k] * B[k][j];
                }
        }
        }
        end = omp_get_wtime();
        printf("Work took %f seconds\n", end - start);
        for (i= 0; i< N; i++)
        {
        for (j= 0; j< N; j++)
        {
                printf("%d\t",C[i][j]);
        }
        printf("\n");
        }
        return 0;
}
```

OUTPUT:



```
ansh@ansh:~$ gedit pdclab4q2iv.c
ansh@ansh:~$ gcc -o a pdclab4q2iv.c -fopenmp
ansh@ansh:~$ ./a
Work took 0.000980 seconds
12      12      12
12      12      12
12      12      12
ansh@ansh:~$
```

Graph:


Time
Time