NAME – ANSH GOEL

REGISTER NO.- 20BCE1798

PARALLEL AND DISTRIBUTED COMPUTING

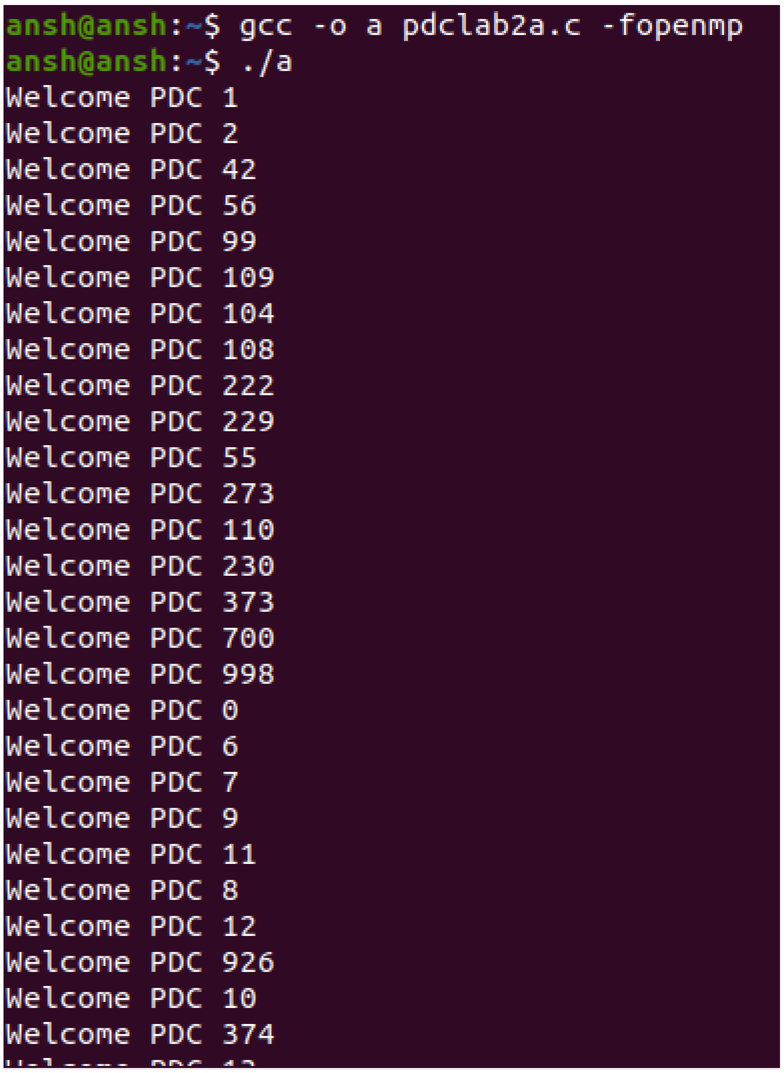
LAB – 2

1. **Create threads using runtime library routines.**

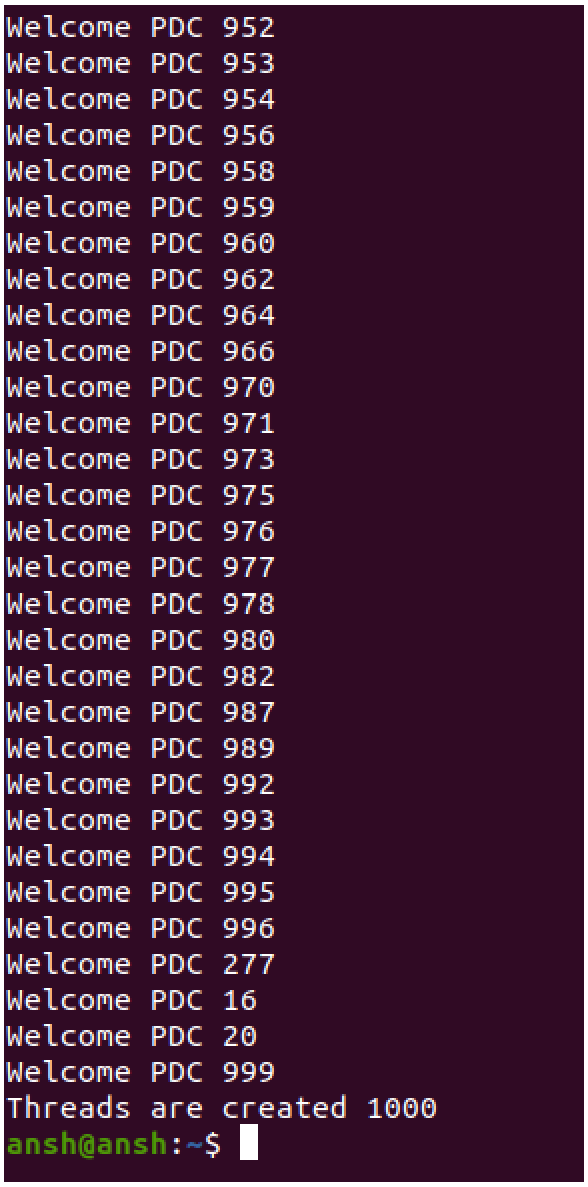
CODE:

```
#include<stdlib.h>

#include<stdio.h>

#include<omp.h>

int main(int argc, char *argv[])

{

        int tid,nthread;

        omp_set_num_threads(1000); #pragma

        omp parallel private(tid)

        {

                tid = omp_get_thread_num(); printf("Welcome PDC %d

                \n", tid);

        }

        if(tid == 0)

        {

                nthread = omp_get_max_threads(); printf("Threads are

                created %d\n", nthread);

        }

}
```

OUTPUT:



```
ansh@ansh:~$ gcc -o a pdclab2a.c -fopenmp
ansh@ansh:~$ ./a
Welcome PDC 1
Welcome PDC 2
Welcome PDC 42
Welcome PDC 56
Welcome PDC 99
Welcome PDC 109
Welcome PDC 104
Welcome PDC 108
Welcome PDC 222
Welcome PDC 229
Welcome PDC 55
Welcome PDC 273
Welcome PDC 110
Welcome PDC 230
Welcome PDC 373
Welcome PDC 700
Welcome PDC 998
Welcome PDC 0
Welcome PDC 6
Welcome PDC 7
Welcome PDC 9
Welcome PDC 11
Welcome PDC 8
Welcome PDC 12
Welcome PDC 926
Welcome PDC 10
Welcome PDC 374
```

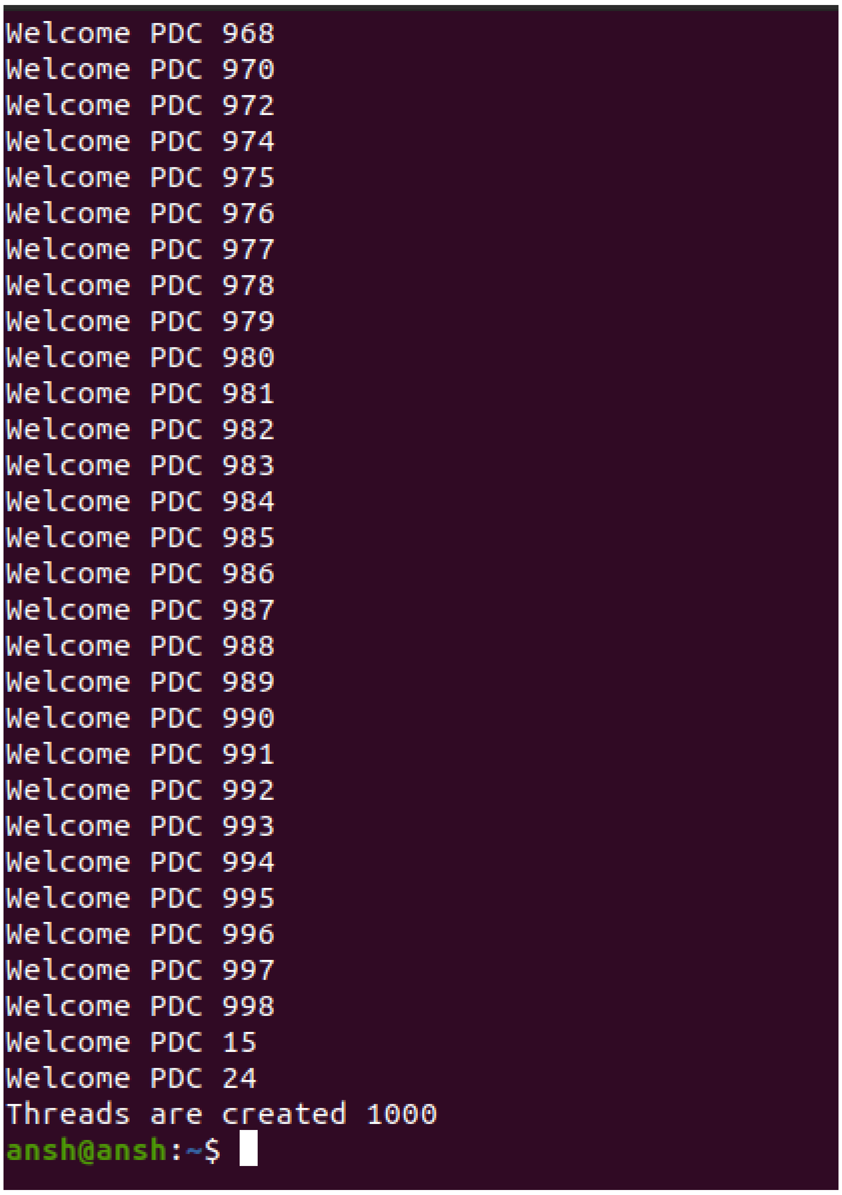


```
Welcome PDC 952
Welcome PDC 953
Welcome PDC 954
Welcome PDC 956
Welcome PDC 958
Welcome PDC 959
Welcome PDC 960
Welcome PDC 962
Welcome PDC 964
Welcome PDC 966
Welcome PDC 970
Welcome PDC 971
Welcome PDC 973
Welcome PDC 975
Welcome PDC 976
Welcome PDC 977
Welcome PDC 978
Welcome PDC 980
Welcome PDC 982
Welcome PDC 987
Welcome PDC 989
Welcome PDC 992
Welcome PDC 993
Welcome PDC 994
Welcome PDC 995
Welcome PDC 996
Welcome PDC 277
Welcome PDC 16
Welcome PDC 20
Welcome PDC 999
Threads are created 1000
ansh@ansh:~$
```

2. Create threads using compiler directives

CODE:

```
#include<stdlib.h>

#include<stdio.h>

#include<omp.h>

#define no_of_threads 1000

int main(int argc, char *argv[])

{


int tid,nthread;

omp_set_num_threads(no_of_threads);

 #pragma omp parallel private(tid)

{

tid = omp_get_thread_num();

printf("Welcome PDC %d \n", tid);

}


if(tid == 0)

{

nthread = omp_get_max_threads();

printf("Threads are created %d\n", nthread);

}

}
```

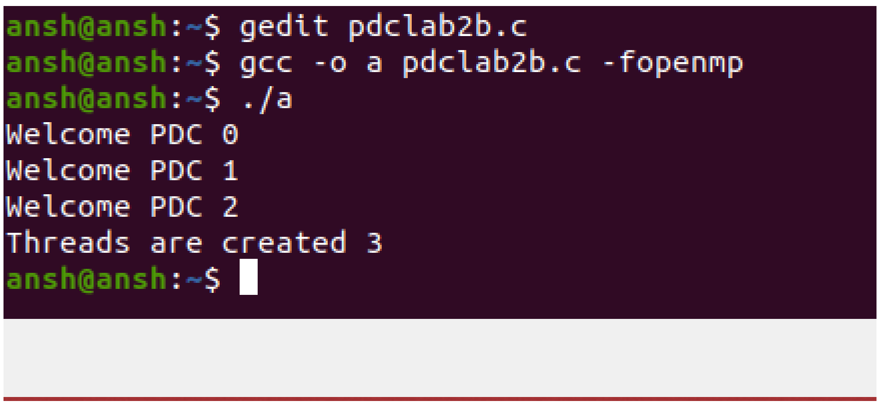


```
Welcome PDC 968
Welcome PDC 970
Welcome PDC 972
Welcome PDC 974
Welcome PDC 975
Welcome PDC 976
Welcome PDC 977
Welcome PDC 978
Welcome PDC 979
Welcome PDC 980
Welcome PDC 981
Welcome PDC 982
Welcome PDC 983
Welcome PDC 984
Welcome PDC 985
Welcome PDC 986
Welcome PDC 987
Welcome PDC 988
Welcome PDC 989
Welcome PDC 990
Welcome PDC 991
Welcome PDC 992
Welcome PDC 993
Welcome PDC 994
Welcome PDC 995
Welcome PDC 996
Welcome PDC 997
Welcome PDC 998
Welcome PDC 15
Welcome PDC 24
Threads are created 1000
ansh@ansh:~$
```

3. Create threads using environmental variables.

Code –

```
#include<stdlib.h>

#include<stdio.h>

#include<omp.h>

int main(int argc, char *argv[])

{

        int tid,nthread;

        #pragma omp parallel private(tid, nthread)


        {

                tid = omp_get_thread_num(); printf("Welcome PDC %d

                \n", tid);

        }

        if(tid == 0)

        {

                nthread = omp_get_max_threads(); printf("Threads are

                created %d\n", nthread);

        }
```

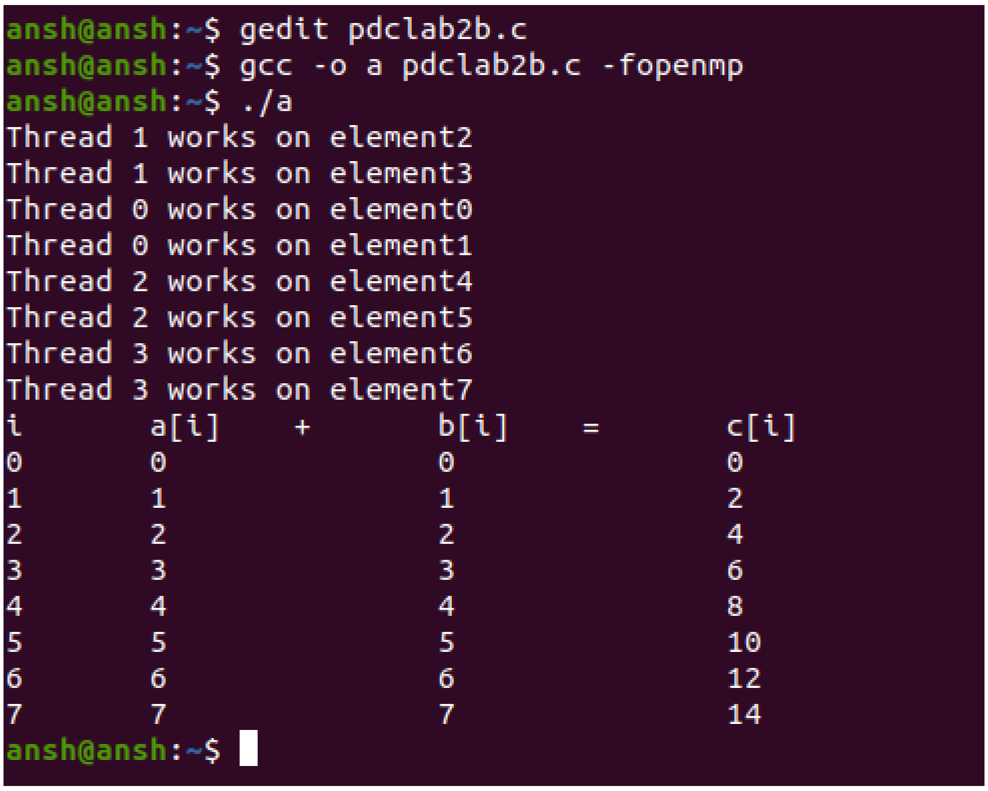


```
ansh@ansh:~$ gedit pdclab2b.c
ansh@ansh:~$ gcc -o a pdclab2b.c -fopenmp
ansh@ansh:~$ ./a
Welcome PDC 0
Welcome PDC 1
Welcome PDC 2
Threads are created 3
ansh@ansh:~$
```

1. **Vector addition**

Code –

```
#include <stdlib.h>

#include <stdio.h>

#include <omp.h> #define

ARRAY_SIZE 8

#define NUM_THREADS 4
int main(int argc, char *argv[])

{

    int   *a;

    int   *b;

    int *c;

    int n = ARRAY_SIZE;int

    n_per_thread;

    int total_threads = NUM_THREADS;int i;

    // allocate spce for the arrays

    a = (int *)malloc(sizeof(int) * n);b = (int

    *)malloc(sizeof(int)   *   n);  c   =   (int

    *)malloc(sizeof(int) * n);for (i = 0; i < n;

    i++)

    {

        a[i] = i;

    }

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

    {

        b[i] = i;

    }

    omp_set_num_threads(total_threads);n_per_thread = n /
    total_threads;
#pragma omp parallel for shared(a, b, c) private(i) schedule(static, n_per_thread)for (i = 0; i < n;
    i++)
    {
        c[i] = a[i] + b[i];
        printf("Thread %d works on element%d\n", omp_get_thread_num(), i);
    }
```



```
ansh@ansh:~$ gedit pdclab2b.c
ansh@ansh:~$ gcc -o a pdclab2b.c -fopenmp
ansh@ansh:~$ ./a
Thread 1 works on element2
Thread 1 works on element3
Thread 0 works on element0
Thread 0 works on element1
Thread 2 works on element4
Thread 2 works on element5
Thread 3 works on element6
Thread 3 works on element7
i       a[i]    +       b[i]    =       c[i]
0       0               0               0
1       1               1               2
2       2               2               4
3       3               3               6
4       4               4               8
5       5               5               10
6       6               6               12
7       7               7               14
ansh@ansh:~$ 
```