20BCE1550
Samridh Anand Paatni
CSE4001 Lab 02
Open MP Programming

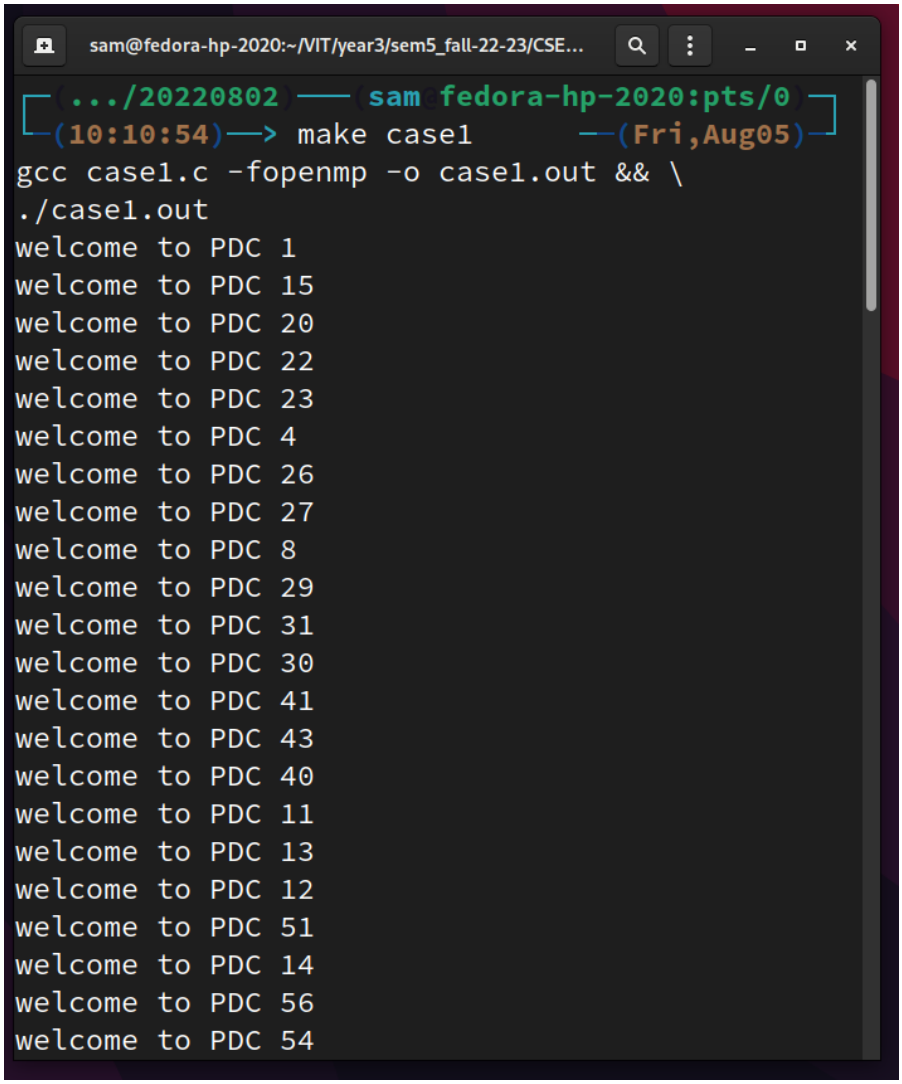
Q1. Create a hundred threads using:

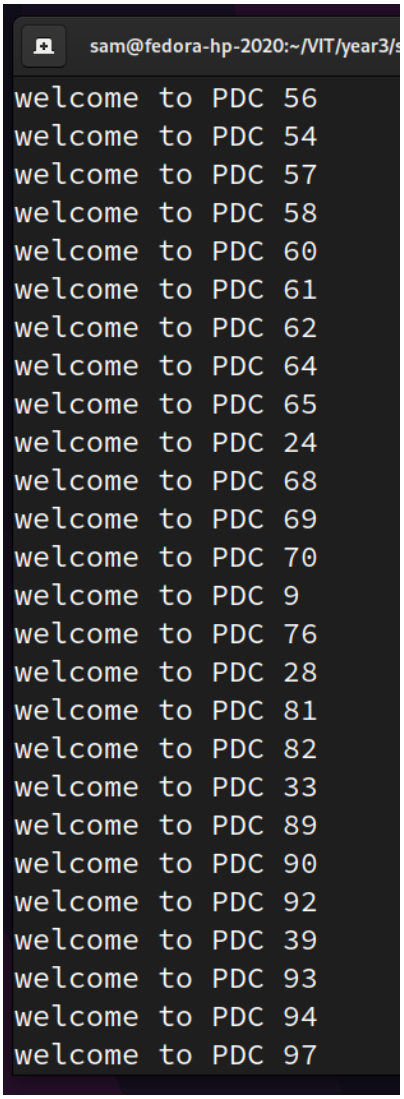
a) Runtime Library Routines

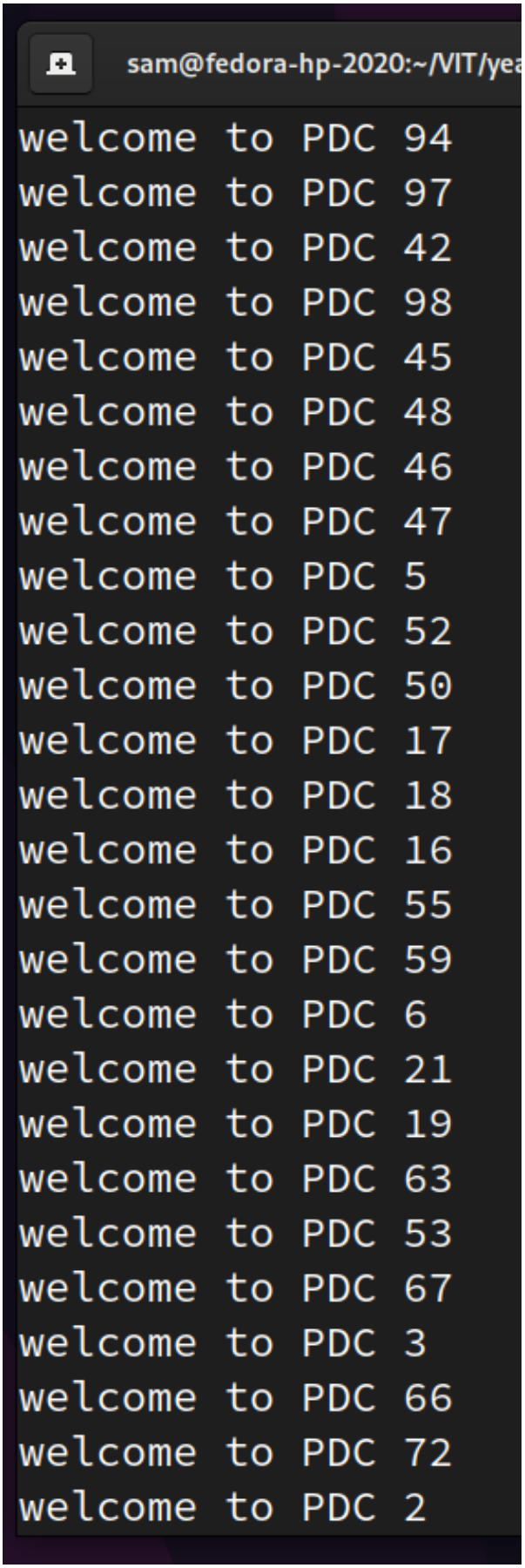
Code:

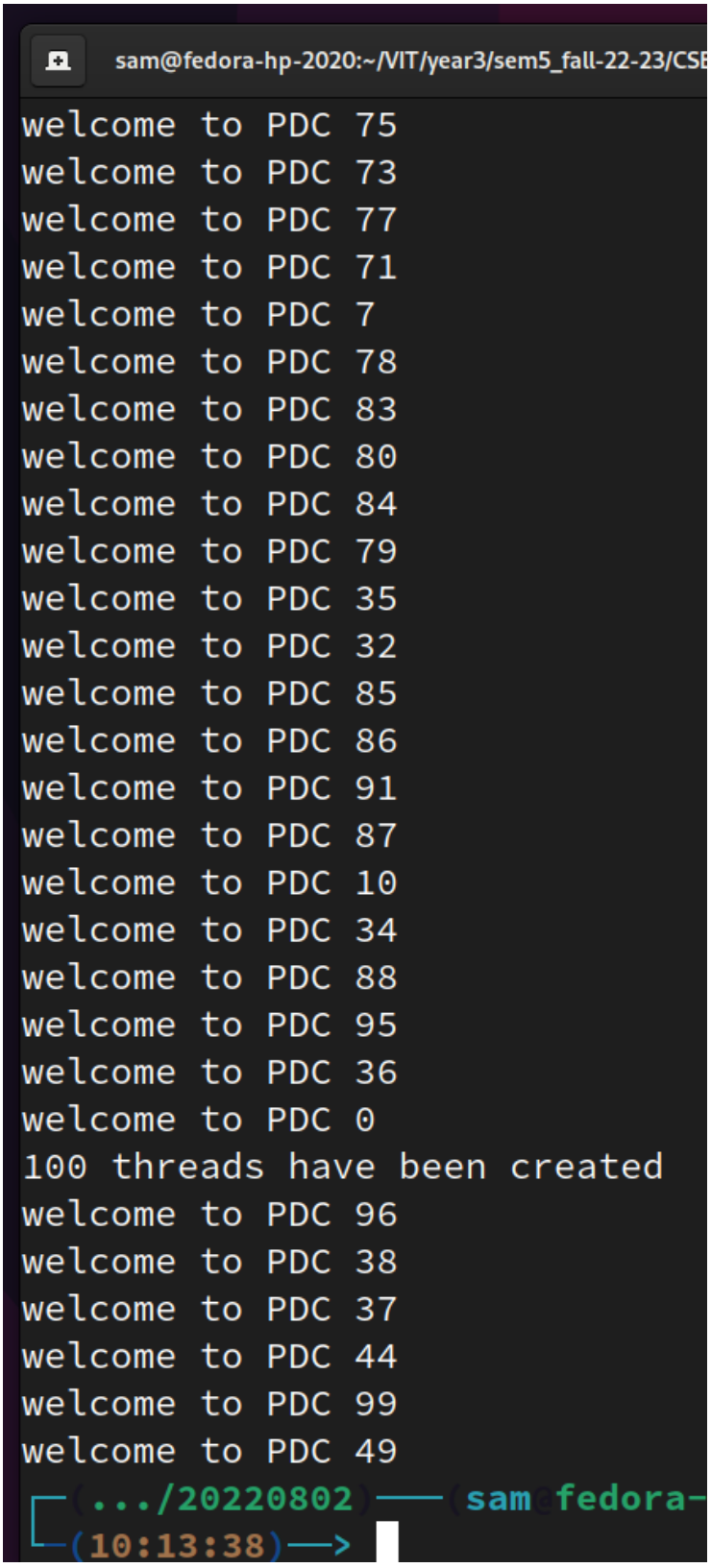
```
// using runtime library routines

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

// compile using: `gcc filename -fopenmp`

int main(int argc, char *argv[]) {
    int tid, numThreads;
    omp_set_num_threads(100);
    # pragma omp parallel private (tid, numThreads)
    {
        tid = omp_get_thread_num();
        printf("welcome to PDC %d\n", tid);

        if (tid == 0) {
            numThreads = omp_get_num_threads();
            printf("%d threads have been created\n", numThreads);
        }
    }
    return 0;
}
```

Output:


sam@fedora-hp-2020:~/VIT/year3/sem5_fall-22-23/CSE...
.../20220802 sam fedora-hp-2020:pts/0
10:10:54 make case1 Fri,Aug05
gcc case1.c -fopenmp -o case1.out && \
./case1.out
welcome to PDC 1
welcome to PDC 15
welcome to PDC 20
welcome to PDC 22
welcome to PDC 23
welcome to PDC 4
welcome to PDC 26
welcome to PDC 27
welcome to PDC 8
welcome to PDC 29
welcome to PDC 31
welcome to PDC 30
welcome to PDC 41
welcome to PDC 43
welcome to PDC 40
welcome to PDC 11
welcome to PDC 13
welcome to PDC 12
welcome to PDC 51
welcome to PDC 14
welcome to PDC 56
welcome to PDC 54


sam@fedora-hp-2020:~/VIT/year3/
welcome to PDC 56
welcome to PDC 54
welcome to PDC 57
welcome to PDC 58
welcome to PDC 60
welcome to PDC 61
welcome to PDC 62
welcome to PDC 64
welcome to PDC 65
welcome to PDC 24
welcome to PDC 68
welcome to PDC 69
welcome to PDC 70
welcome to PDC 9
welcome to PDC 76
welcome to PDC 28
welcome to PDC 81
welcome to PDC 82
welcome to PDC 33
welcome to PDC 89
welcome to PDC 90
welcome to PDC 92
welcome to PDC 39
welcome to PDC 93
welcome to PDC 94
welcome to PDC 97


sam@fedora-hp-2020:~/VIT/yea
welcome to PDC 94
welcome to PDC 97
welcome to PDC 42
welcome to PDC 98
welcome to PDC 45
welcome to PDC 48
welcome to PDC 46
welcome to PDC 47
welcome to PDC 5
welcome to PDC 52
welcome to PDC 50
welcome to PDC 17
welcome to PDC 18
welcome to PDC 16
welcome to PDC 55
welcome to PDC 59
welcome to PDC 6
welcome to PDC 21
welcome to PDC 19
welcome to PDC 63
welcome to PDC 53
welcome to PDC 67
welcome to PDC 3
welcome to PDC 66
welcome to PDC 72
welcome to PDC 2

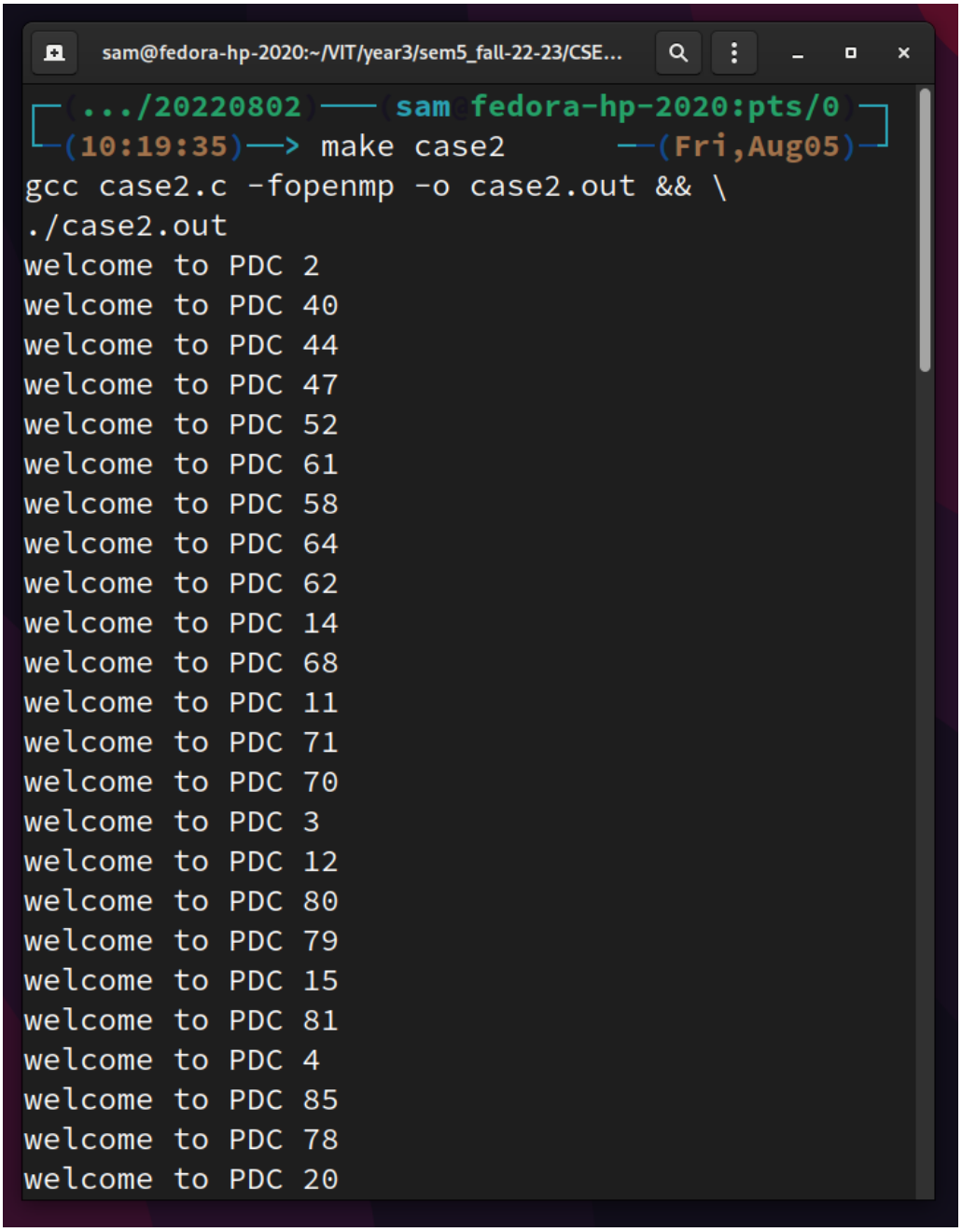

sam@fedora-hp-2020:~/VIT/year3/sem5_fall-22-23/CSE
welcome to PDC 75
welcome to PDC 73
welcome to PDC 77
welcome to PDC 71
welcome to PDC 7
welcome to PDC 78
welcome to PDC 83
welcome to PDC 80
welcome to PDC 84
welcome to PDC 79
welcome to PDC 35
welcome to PDC 32
welcome to PDC 85
welcome to PDC 86
welcome to PDC 91
welcome to PDC 87
welcome to PDC 10
welcome to PDC 34
welcome to PDC 88
welcome to PDC 95
welcome to PDC 36
welcome to PDC 0
100 threads have been created
welcome to PDC 96
welcome to PDC 38
welcome to PDC 37
welcome to PDC 44
welcome to PDC 99
welcome to PDC 49
(.../20220802)
(10:13:38)—>

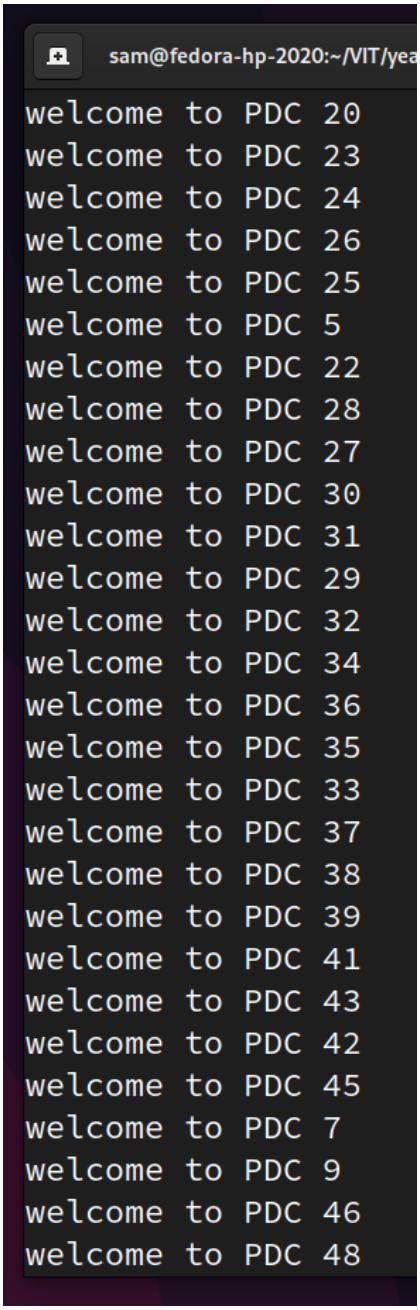
b)Compiler Directives

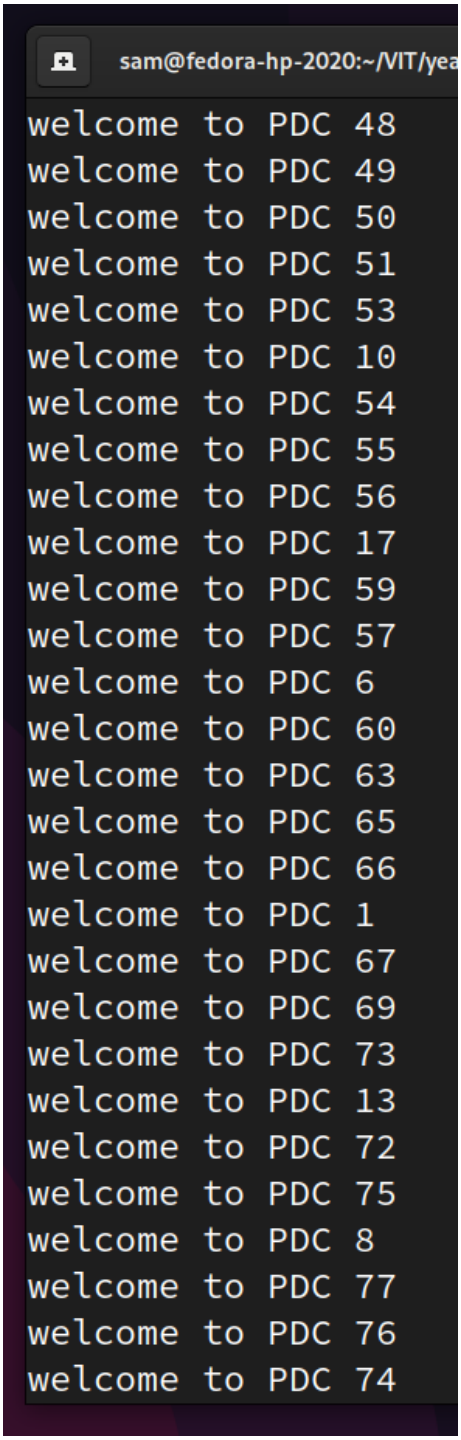
Code:

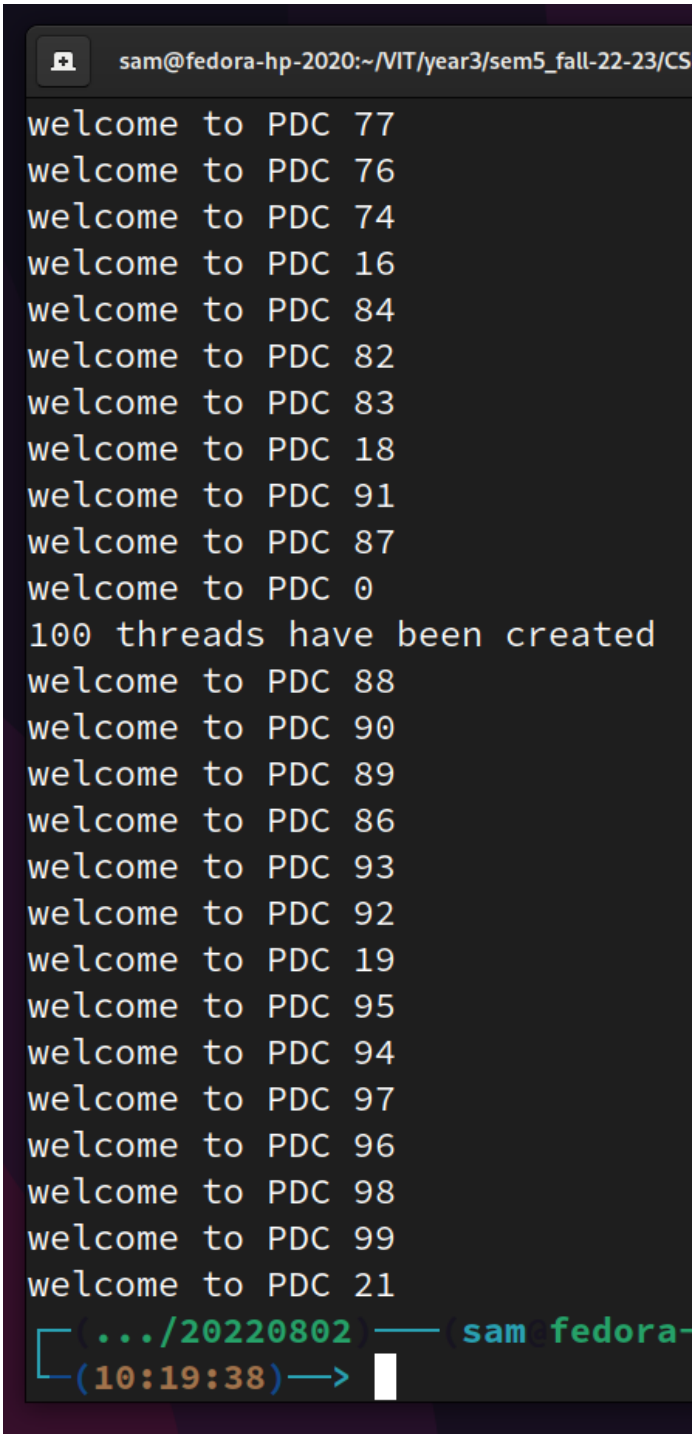
```
// using compiler directives

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

// compile using: `gcc filename -fopenmp`

int main(int argc, char *argv[]) {
    int tid, numThreads;

    # pragma omp parallel private (tid, numThreads) num_threads(100)
    {
        tid = omp_get_thread_num();
        printf("welcome to PDC %d\n", tid);

        if (tid == 0) {
            numThreads = omp_get_num_threads();
            printf("%d threads have been created\n", numThreads);
        }
    }
    return 0;
}
```

Output:


sam@fedora-hp-2020:~/VIT/year3/sem5_fall-22-23/CSE...
.../20220802 sam fedora-hp-2020:pts/0
(10:19:35)--> make case2 (Fri,Aug05)
gcc case2.c -fopenmp -o case2.out && \
./case2.out
welcome to PDC 2
welcome to PDC 40
welcome to PDC 44
welcome to PDC 47
welcome to PDC 52
welcome to PDC 61
welcome to PDC 58
welcome to PDC 64
welcome to PDC 62
welcome to PDC 14
welcome to PDC 68
welcome to PDC 11
welcome to PDC 71
welcome to PDC 70
welcome to PDC 3
welcome to PDC 12
welcome to PDC 80
welcome to PDC 79
welcome to PDC 15
welcome to PDC 81
welcome to PDC 4
welcome to PDC 85
welcome to PDC 78
welcome to PDC 20


sam@fedora-hp-2020:~/VIT/yea
welcome to PDC 20
welcome to PDC 23
welcome to PDC 24
welcome to PDC 26
welcome to PDC 25
welcome to PDC 5
welcome to PDC 22
welcome to PDC 28
welcome to PDC 27
welcome to PDC 30
welcome to PDC 31
welcome to PDC 29
welcome to PDC 32
welcome to PDC 34
welcome to PDC 36
welcome to PDC 35
welcome to PDC 33
welcome to PDC 37
welcome to PDC 38
welcome to PDC 39
welcome to PDC 41
welcome to PDC 43
welcome to PDC 42
welcome to PDC 45
welcome to PDC 7
welcome to PDC 9
welcome to PDC 46
welcome to PDC 48


sam@fedora-hp-2020:~/VIT/yea
welcome to PDC 48
welcome to PDC 49
welcome to PDC 50
welcome to PDC 51
welcome to PDC 53
welcome to PDC 10
welcome to PDC 54
welcome to PDC 55
welcome to PDC 56
welcome to PDC 17
welcome to PDC 59
welcome to PDC 57
welcome to PDC 6
welcome to PDC 60
welcome to PDC 63
welcome to PDC 65
welcome to PDC 66
welcome to PDC 1
welcome to PDC 67
welcome to PDC 69
welcome to PDC 73
welcome to PDC 13
welcome to PDC 72
welcome to PDC 75
welcome to PDC 8
welcome to PDC 77
welcome to PDC 76
welcome to PDC 74

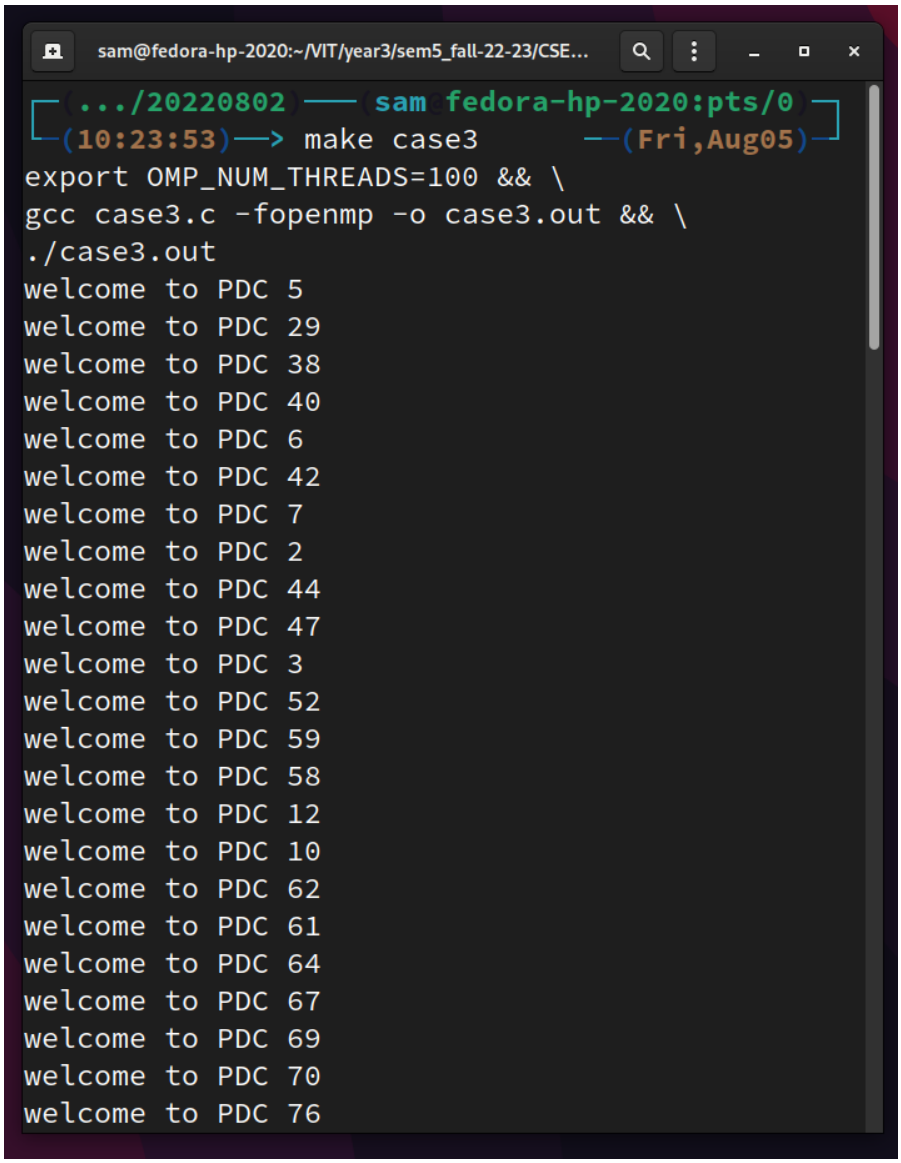

sam@fedora-hp-2020:~/VIT/year3/sem5_fall-22-23/CS
welcome to PDC 77
welcome to PDC 76
welcome to PDC 74
welcome to PDC 16
welcome to PDC 84
welcome to PDC 82
welcome to PDC 83
welcome to PDC 18
welcome to PDC 91
welcome to PDC 87
welcome to PDC 0
100 threads have been created
welcome to PDC 88
welcome to PDC 90
welcome to PDC 89
welcome to PDC 86
welcome to PDC 93
welcome to PDC 92
welcome to PDC 19
welcome to PDC 95
welcome to PDC 94
welcome to PDC 97
welcome to PDC 96
welcome to PDC 98
welcome to PDC 99
welcome to PDC 21
.../20220802
fedora-
10:19:38

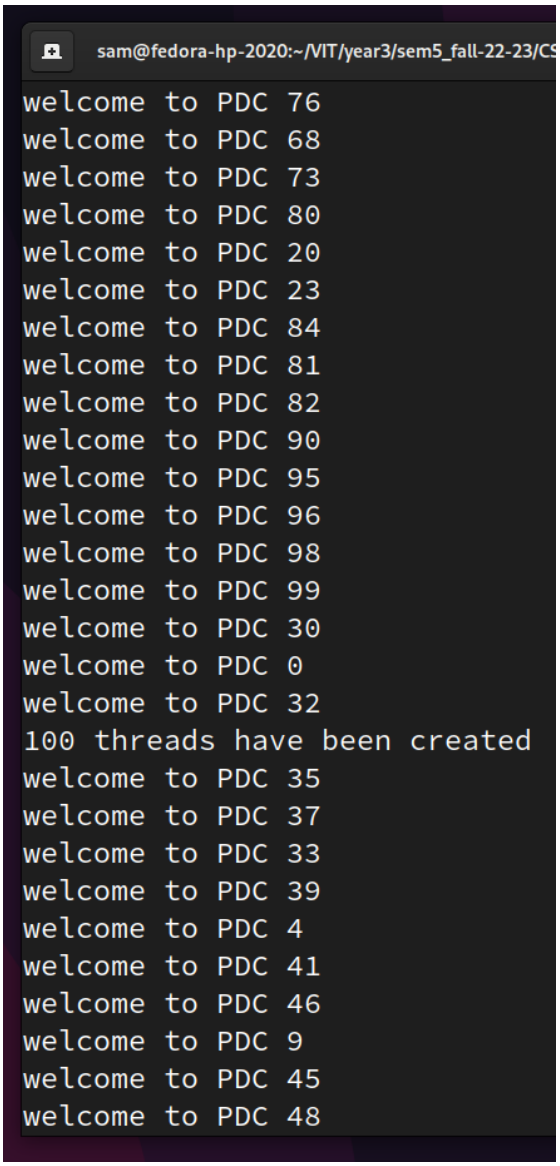
c) Environment Variables

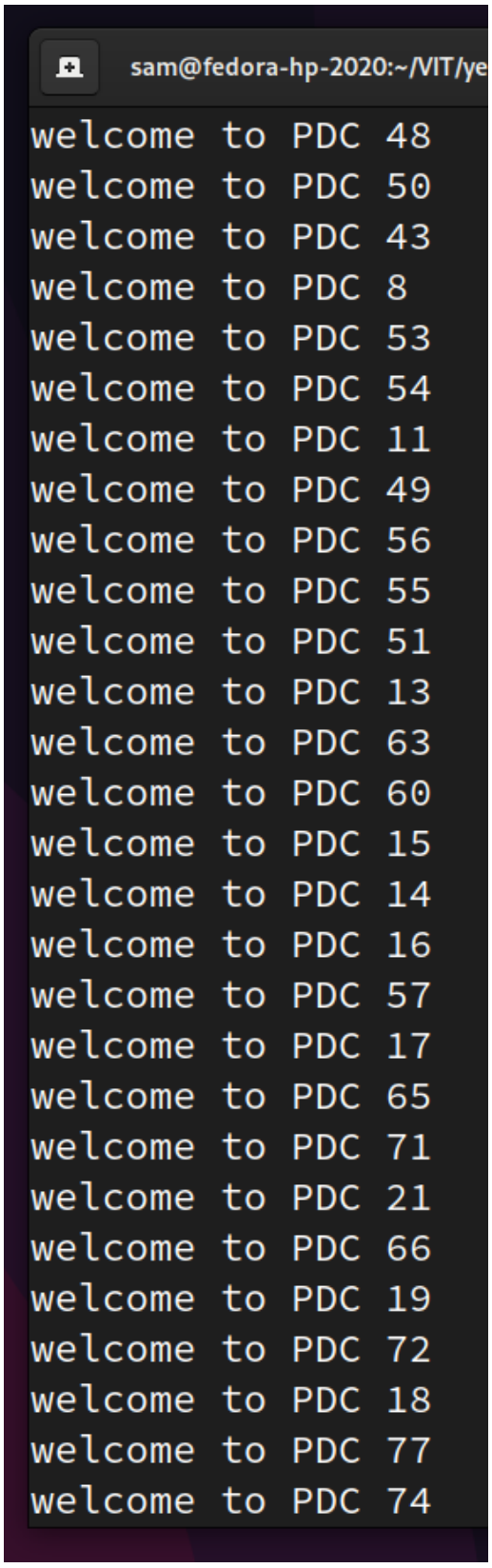
Code:

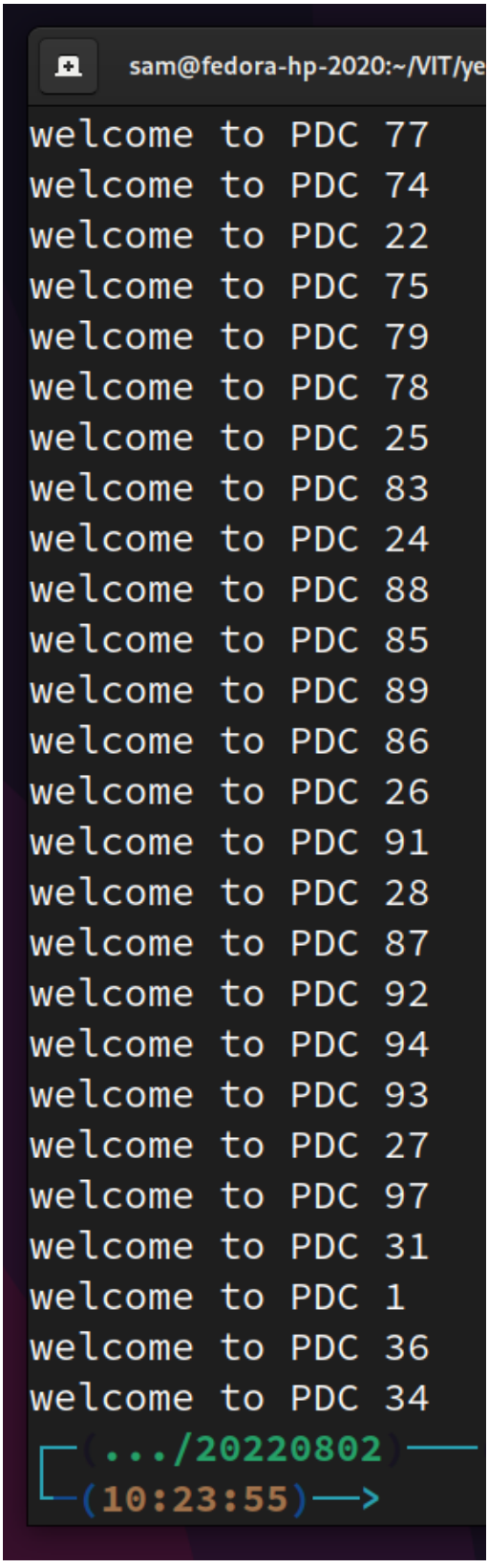
```c
// using environment variables

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

// compile using: `gcc filename -fopenmp`
// before running, give the command: `export OMP_NUM_THREADS=100` in bash

int main(int argc, char *argv[]) {
    int tid, numThreads;

    # pragma omp parallel private (tid, numThreads)
    {
        tid = omp_get_thread_num();
        printf("welcome to PDC %d\n", tid);

        if (tid == 0) {
            numThreads = omp_get_num_threads();
            printf("%d threads have been created\n", numThreads);
        }
    }
    return 0;
}
```

Output:


sam@fedora-hp-2020:~/VIT/year3/sem5_fall-22-23/CSE...
.../20220802 sam fedora-hp-2020:pts/0
(10:23:53)—> make case3 —(Fri,Aug05)
export OMP_NUM_THREADS=100 && \
gcc case3.c -fopenmp -o case3.out && \
./case3.out
welcome to PDC 5
welcome to PDC 29
welcome to PDC 38
welcome to PDC 40
welcome to PDC 6
welcome to PDC 42
welcome to PDC 7
welcome to PDC 2
welcome to PDC 44
welcome to PDC 47
welcome to PDC 3
welcome to PDC 52
welcome to PDC 59
welcome to PDC 58
welcome to PDC 12
welcome to PDC 10
welcome to PDC 62
welcome to PDC 61
welcome to PDC 64
welcome to PDC 67
welcome to PDC 69
welcome to PDC 70
welcome to PDC 76


sam@fedora-hp-2020:~/VIT/year3/sem5_fall-22-23/CS
welcome to PDC 76
welcome to PDC 68
welcome to PDC 73
welcome to PDC 80
welcome to PDC 20
welcome to PDC 23
welcome to PDC 84
welcome to PDC 81
welcome to PDC 82
welcome to PDC 90
welcome to PDC 95
welcome to PDC 96
welcome to PDC 98
welcome to PDC 99
welcome to PDC 30
welcome to PDC 0
welcome to PDC 32
100 threads have been created
welcome to PDC 35
welcome to PDC 37
welcome to PDC 33
welcome to PDC 39
welcome to PDC 4
welcome to PDC 41
welcome to PDC 46
welcome to PDC 9
welcome to PDC 45
welcome to PDC 48


sam@fedora-hp-2020:~/VIT/ye
welcome to PDC 48
welcome to PDC 50
welcome to PDC 43
welcome to PDC 8
welcome to PDC 53
welcome to PDC 54
welcome to PDC 11
welcome to PDC 49
welcome to PDC 56
welcome to PDC 55
welcome to PDC 51
welcome to PDC 13
welcome to PDC 63
welcome to PDC 60
welcome to PDC 15
welcome to PDC 14
welcome to PDC 16
welcome to PDC 57
welcome to PDC 17
welcome to PDC 65
welcome to PDC 71
welcome to PDC 21
welcome to PDC 66
welcome to PDC 19
welcome to PDC 72
welcome to PDC 18
welcome to PDC 77
welcome to PDC 74


sam@fedora-hp-2020:~/VIT/ye
welcome to PDC 77
welcome to PDC 74
welcome to PDC 22
welcome to PDC 75
welcome to PDC 79
welcome to PDC 78
welcome to PDC 25
welcome to PDC 83
welcome to PDC 24
welcome to PDC 88
welcome to PDC 85
welcome to PDC 89
welcome to PDC 86
welcome to PDC 26
welcome to PDC 91
welcome to PDC 28
welcome to PDC 87
welcome to PDC 92
welcome to PDC 94
welcome to PDC 93
welcome to PDC 27
welcome to PDC 97
welcome to PDC 31
welcome to PDC 1
welcome to PDC 36
welcome to PDC 34
.../20220802
10:23:55

Q2. Implement vector addition in serial and parallel and compare the results. Do the parallel computation using a 1000 threads.

Code:

Serial addition:

```c
#include <stdio.h>
#include <time.h>

#define VECTOR_SIZE 100000

int main() {
    // make the vectors
    int a[VECTOR_SIZE], b[VECTOR_SIZE], c[VECTOR_SIZE];
    for (int i = 0; i < VECTOR_SIZE; i++) {
        a[i] = VECTOR_SIZE - i;
        b[i] = i;
    }

    // serially add the vectors
    clock_t tSerial = clock();
    for (int i = 0; i < VECTOR_SIZE; i++) {
        c[i] = a[i] + b[i];
    }
    tSerial = clock() - tSerial;

    // show the output
    printf(
        "Serial addition took %f seconds\n",
        ((double) tSerial)/CLOCKS_PER_SEC
    );

    return 0;
}
```

Parallel addition:

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

#define VECTOR_SIZE 100000

int main(int argc, char * argv[]) {
    // make the vectors
    int a[VECTOR_SIZE], b[VECTOR_SIZE], c[VECTOR_SIZE];
```

```
    for (int i = 0; i < VECTOR_SIZE; i++) {
        a[i] = VECTOR_SIZE - i;
        b[i] = i;
    }

    int nThreads = atoi(argv[1]);

    // paralelly add the vectors:

    clock_t tPar = clock();

    // the part of the vector one thread will access
    int slice_size = VECTOR_SIZE / nThreads;

    int slice_start, slice_end;
    int tid;

    // make threads
    omp_set_num_threads(nThreads);
    #pragma omp parallel private (tid, slice_start, slice_end)
    {
        // allot a slice to the particular thread
        tid = omp_get_thread_num();
        slice_start = tid * slice_size;
        slice_end = slice_start + slice_size;

        // perform addition for the elements in the allotted slice_size
        for (int i = slice_start; i < slice_end; i++) {
            c[i] = a[i] + b[i];
        }
    }

    tPar = clock() - tPar;

    // show the output
    printf(
        "Parallel addition took %f seconds with %d threads\n",
        ((double) tPar)/CLOCKS_PER_SEC,
        nThreads
    );

    return 0;
}
```

Output:

Running the computation with 4 threads is the fastest (because I have a quad-core laptop). The parallel computation with a 100 more threads is slower than the serial execution.