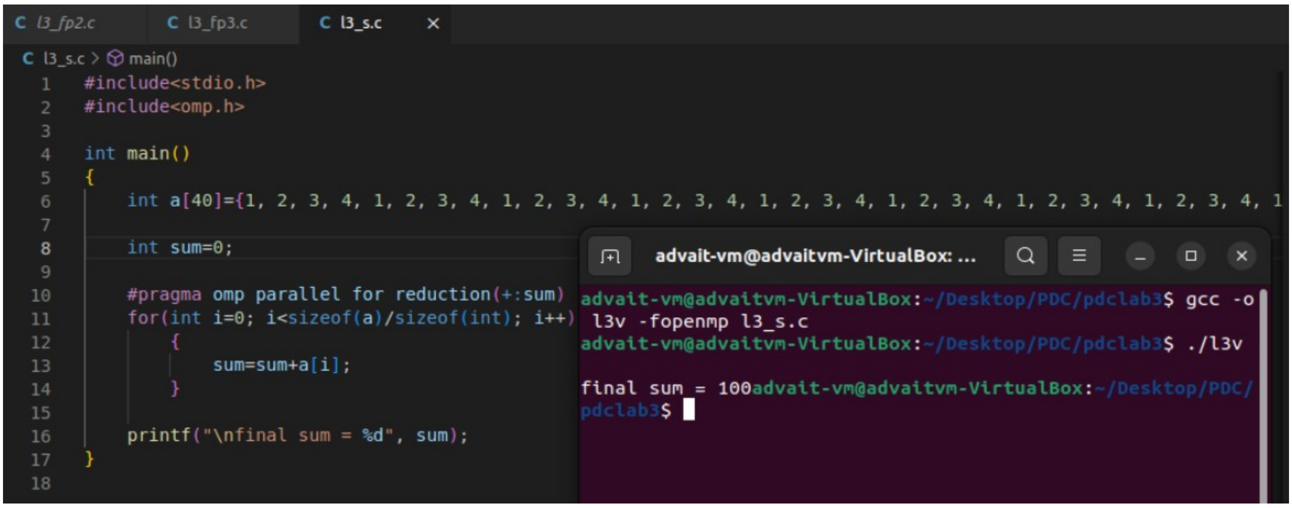
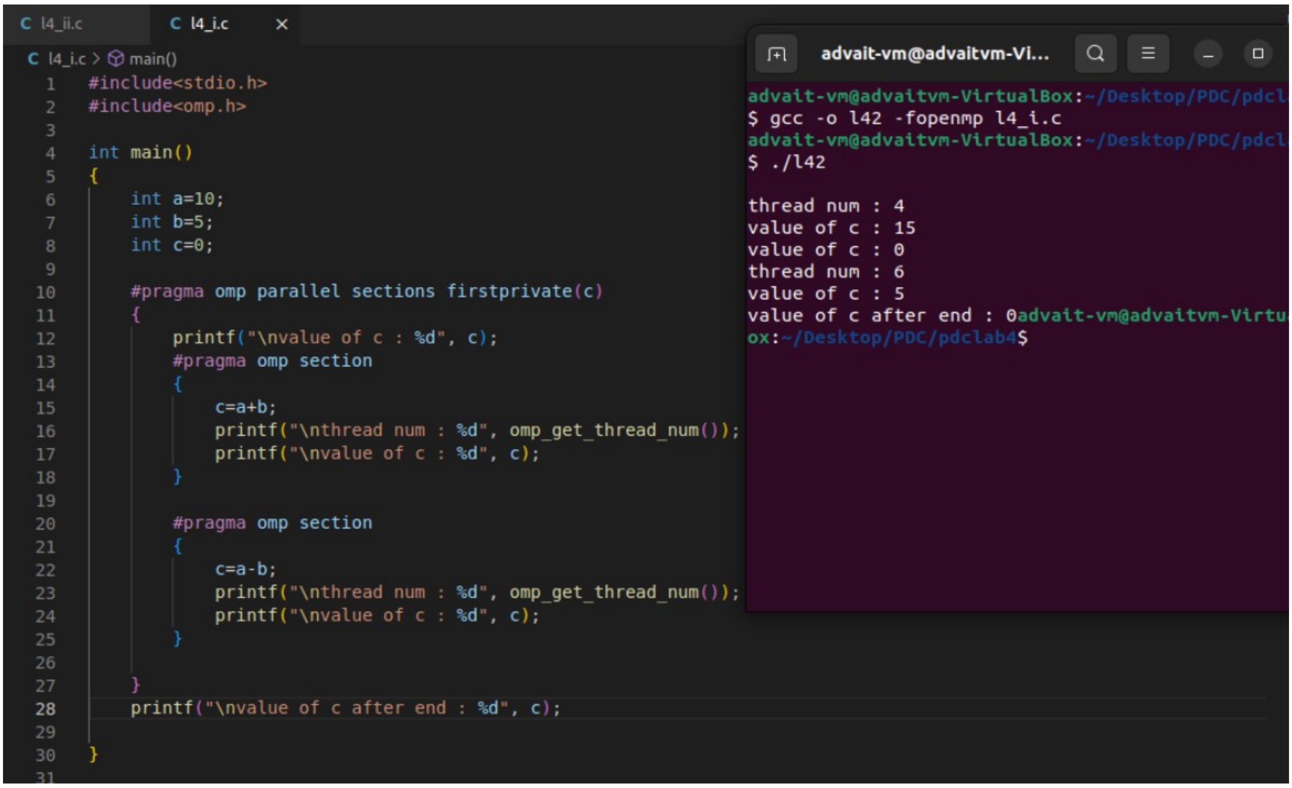
PDC Lab 4

Advait Deochakke
20BCE1143

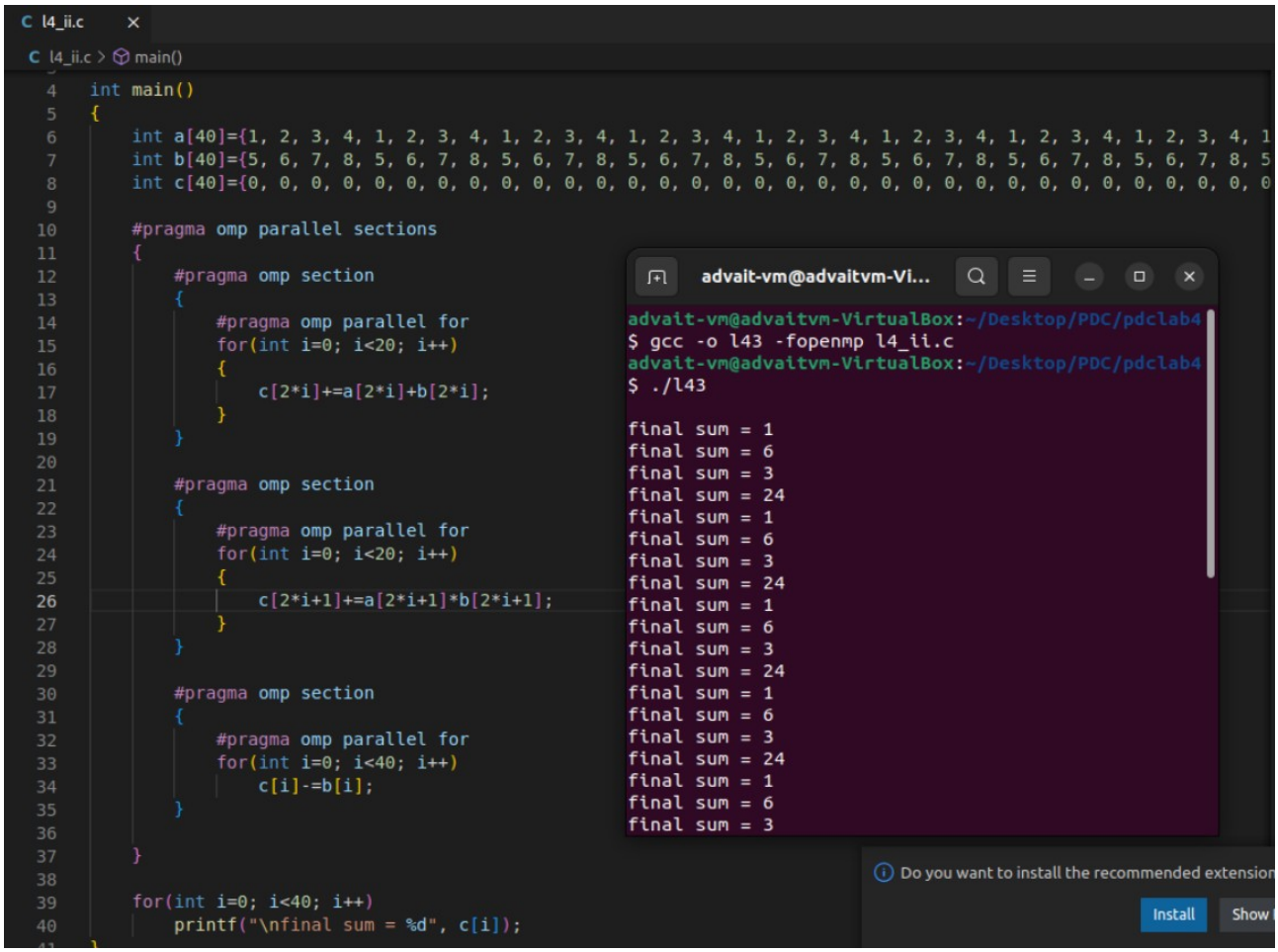
1) Reduction



2) c=a+b; c=a-b



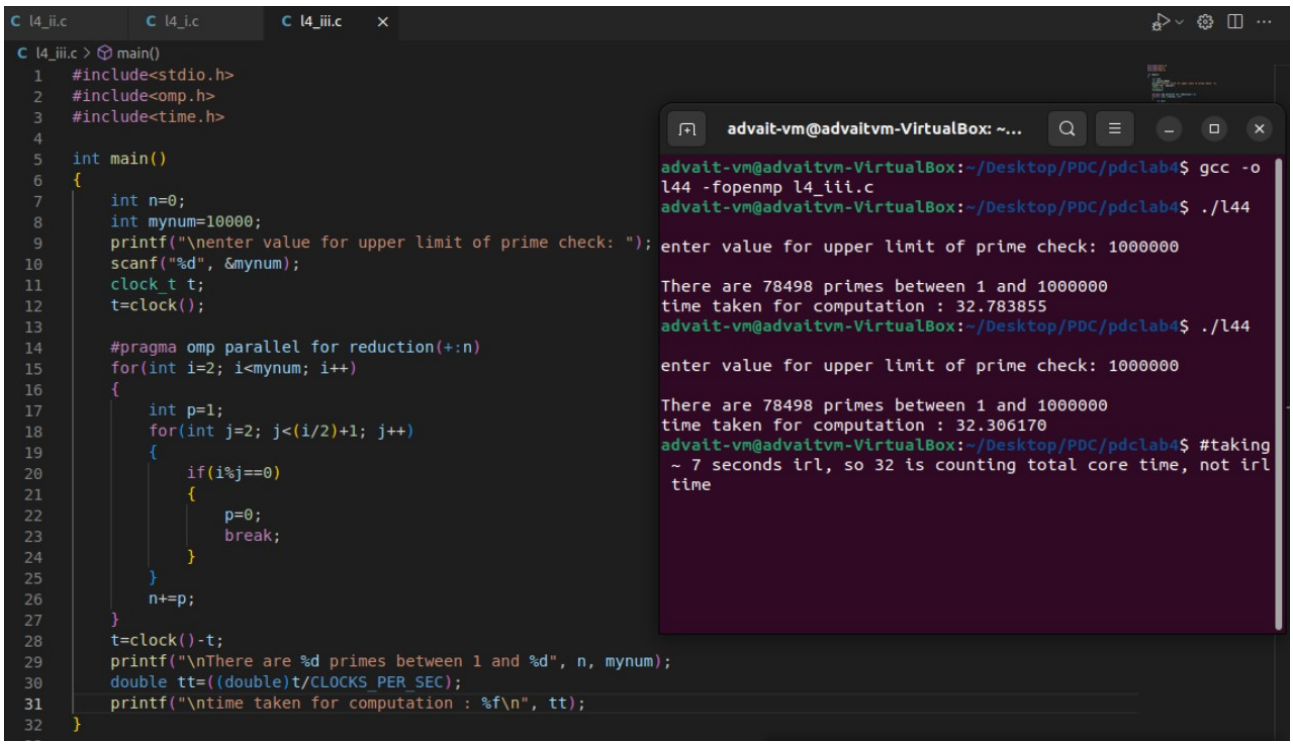
3) c[i]=a[i]+b[i]; c[i]=a[i]*b[i]; c[i]-b[i]



```c
int main()
{
    int a[40]={1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1
    int b[40]={5, 6, 7, 8, 5, 6, 7, 8, 5, 6, 7, 8, 5, 6, 7, 8, 5, 6, 7, 8, 5, 6, 7, 8, 5, 6, 7, 8, 5
    int c[40]={0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

    #pragma omp parallel sections
    {
        #pragma omp section
        {
            #pragma omp parallel for
            for(int i=0; i<20; i++)
            {
                c[2*i]+=a[2*i]+b[2*i];
            }
        }

        #pragma omp section
        {
            #pragma omp parallel for
            for(int i=0; i<20; i++)
            {
                c[2*i+1]+=a[2*i+1]*b[2*i+1];
            }
        }

        #pragma omp section
        {
            #pragma omp parallel for
            for(int i=0; i<40; i++)
                c[i]-=b[i];
        }

    }

    for(int i=0; i<40; i++)
        printf("\nfinal sum = %d", c[i]);
}
```

```
advait-vm@advaitvm-VirtualBox:~/Desktop/PDC/pdclab4
$ gcc -o l43 -fopenmp l4_ii.c
advait-vm@advaitvm-VirtualBox:~/Desktop/PDC/pdclab4
$ ./l43

final sum = 1
final sum = 6
final sum = 3
final sum = 24
final sum = 1
final sum = 6
final sum = 3
final sum = 24
final sum = 1
final sum = 6
final sum = 3
final sum = 24
final sum = 1
final sum = 6
final sum = 3
final sum = 24
final sum = 1
final sum = 6
final sum = 3
```

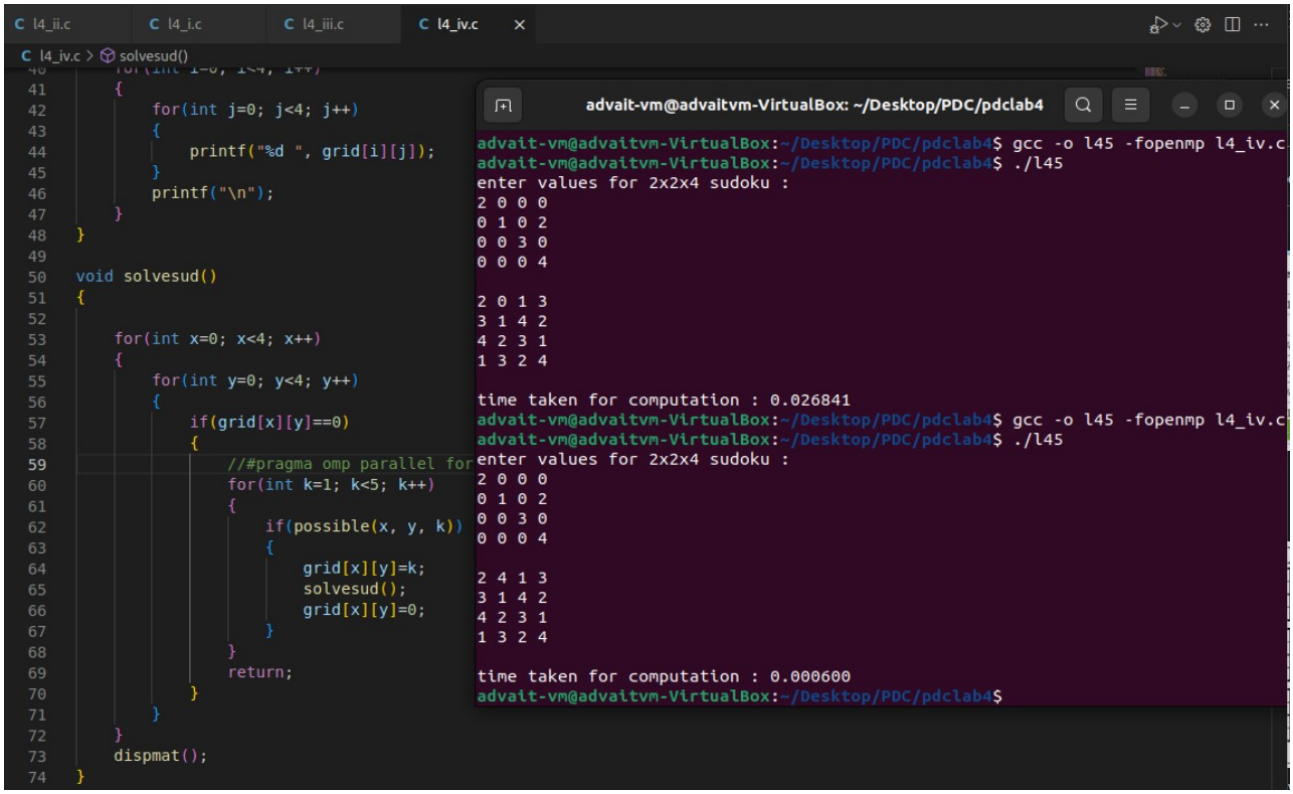
4) listing of prime nos <n



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

int main()
{
    int n=0;
    int mynum=10000;
    printf("\nenter value for upper limit of prime check: ");
    scanf("%d", &mynum);
    clock_t t;
    t=clock();

    #pragma omp parallel for reduction(+:n)
    for(int i=2; i<mynum; i++)
    {
        int p=1;
        for(int j=2; j<(i/2)+1; j++)
        {
            if(i%j==0)
            {
                p=0;
                break;
            }
        }
        n+=p;
    }
    t=clock()-t;
    printf("\nThere are %d primes between 1 and %d", n, mynum);
    double tt=((double)t/CLOCKS_PER_SEC);
    printf("\ntime taken for computation : %f\n", tt);
}
```

```
advait-vm@advaitvm-VirtualBox:~/Desktop/PDC/pdclab4$ gcc -o
l44 -fopenmp l4_iii.c
advait-vm@advaitvm-VirtualBox:~/Desktop/PDC/pdclab4$ ./l44

enter value for upper limit of prime check: 1000000

There are 78498 primes between 1 and 1000000
time taken for computation : 32.783855
advait-vm@advaitvm-VirtualBox:~/Desktop/PDC/pdclab4$ ./l44

enter value for upper limit of prime check: 1000000

There are 78498 primes between 1 and 1000000
time taken for computation : 32.306170
advait-vm@advaitvm-VirtualBox:~/Desktop/PDC/pdclab4$ #taking
 ~ 7 seconds irl, so 32 is counting total core time, not irl
 time
```

5) Sudoku solver (2x2x4)



```c
    {
        for(int j=0; j<4; j++)
        {
            printf("%d ", grid[i][j]);
        }
        printf("\n");
    }
}

void solvesud()
{

    for(int x=0; x<4; x++)
    {
        for(int y=0; y<4; y++)
        {
            if(grid[x][y]==0)
            {
                //#pragma omp parallel for
                for(int k=1; k<5; k++)
                {
                    if(possible(x, y, k))
                    {
                        grid[x][y]=k;
                        solvesud();
                        grid[x][y]=0;
                    }
                }
                return;
            }
        }
    }
    dispmat();
}
```

```
advait-vm@advaitvm-VirtualBox:~/Desktop/PDC/pdclab4$ gcc -o l45 -fopenmp l4_iv.c
advait-vm@advaitvm-VirtualBox:~/Desktop/PDC/pdclab4$ ./l45
enter values for 2x2x4 sudoku :
2 0 0 0
0 1 0 2
0 0 3 0
0 0 0 4

2 0 1 3
3 1 4 2
4 2 3 1
1 3 2 4

time taken for computation : 0.026841
advait-vm@advaitvm-VirtualBox:~/Desktop/PDC/pdclab4$ gcc -o l45 -fopenmp l4_iv.c
advait-vm@advaitvm-VirtualBox:~/Desktop/PDC/pdclab4$ ./l45
enter values for 2x2x4 sudoku :
2 0 0 0
0 1 0 2
0 0 3 0
0 0 0 4

2 4 1 3
3 1 4 2
4 2 3 1
1 3 2 4

time taken for computation : 0.000600
advait-vm@advaitvm-VirtualBox:~/Desktop/PDC/pdclab4$
```

Code for Sudoku:

```
#include<stdio.h>
#include<omp.h>
#include<time.h>
#include<stdbool.h>

int grid[4][4]={{0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}};

bool possible(int x, int y, int n)
{

  for(int i=0; i<4; i++)
  {
    if(grid[x][i]==n)
      return false;
  }

  for(int i=0; i<4; i++)
  {
    if(grid[i][y]==n)
      return false;
  }
  int x0=(x/2)*2;
  int y0=(y/2)*2;

  for(int i=0; i<2; i++)
  {
    for(int j=0; j<2; j++)
    {
      if(grid[x0+i][y0+i]==n)
        return false;
    }
  }

  return true;
}

void dispmat()
{
  printf("\n");
  for(int i=0; i<4; i++)
  {
    for(int j=0; j<4; j++)
```

```
    {
      printf("%d ", grid[i][j]);
    }
    printf("\n");
  }
}

void solvesud()
{

  for(int x=0; x<4; x++)
  {
    for(int y=0; y<4; y++)
    {
      if(grid[x][y]==0)
      {
        //#pragma omp parallel for (enable in parallel)
        for(int k=1; k<5; k++)
        {
          if(possible(x, y, k))
          {
            grid[x][y]=k;
            solvesud();
            grid[x][y]=0;
          }
        }
        return;
      }
    }
  }
  dispmat();
}

void takevals()
{
  for(int i=0; i<4; i++)
    scanf("%d %d %d %d", &grid[i][0], &grid[i][1], &grid[i]
[2], &grid[i][3]);
}

int main()
{
  clock_t t;
```

```
  t=clock();

  printf("enter values for 2x2x4 sudoku : \n");
  takevals();
  solvesud();
  t=clock()-t;
  double tt=((double)t/CLOCKS_PER_SEC);
  printf("\ntime taken for computation : %f\n", tt);
}
```