

# PDC Lab 4

Advait Deochakke  
20BCE1143

## 1) Reduction

```
C l3_fp2.c C l3_fp3.c X
```

```
C l3_s.c > main()
1 #include<stdio.h>
2 #include<omp.h>
3
4 int main()
5 {
6     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, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4};
7
8     int sum=0;
9
10    #pragma omp parallel for reduction(+:sum)
11    for(int i=0; i<sizeof(a)/sizeof(int); i++)
12        {
13            sum=sum+a[i];
14        }
15
16    printf("\nfinal sum = %d", sum);
17 }
18
```

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

```
C l4_i.c C l4_i.c x
C l4_i.c > main()
1 #include<stdio.h>
2 #include<omp.h>
3
4 int main()
5 {
6     int a=10;
7     int b=5;
8     int c=0;
9
10    #pragma omp parallel sections firstprivate(c)
11    {
12        printf("\nvalue of c : %d", c);
13        #pragma omp section
14        {
15            c=a+b;
16            printf("\nthread num : %d", omp_get_thread_num());
17            printf("\nvalue of c : %d", c);
18        }
19
20        #pragma omp section
21        {
22            c=a-b;
23            printf("\nthread num : %d", omp_get_thread_num());
24            printf("\nvalue of c : %d", c);
25        }
26    }
27
28    printf("\nvalue of c after end : %d", c);
29
30 }
```

```
advait-vm@advaitvm-VI...
advait-vm@advaitvm-VirtualBox:~/Desktop/PDC/pdclab4$ gcc -o l42 -fopenmp l4_i.c
advait-vm@advaitvm-VirtualBox:~/Desktop/PDC/pdclab4$ ./l42

thread num : 4
value of c : 15
value of c : 0
thread num : 6
value of c : 5
value of c after end : 0advait-vm@advaitvm-VirtualBox:~/Desktop/PDC/pdclab4$
```

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

```
C l4_ii.c x  
C l4_ii.c > main()  
4 int main()  
5 {  
6     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, 2, 3, 4, 1, 2, 3, 4, 1,  
7     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, 6, 7, 8, 5, 6, 7, 8, 5,  
8     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, 0, 0, 0, 0, 0, 0, 0, 0,  
9  
10    #pragma omp parallel sections  
11    {  
12        #pragma omp section  
13        {  
14            #pragma omp parallel for  
15            for(int i=0; i<20; i++)  
16            {  
17                c[2*i]+=a[2*i]+b[2*i];  
18            }  
19        }  
20  
21        #pragma omp section  
22        {  
23            #pragma omp parallel for  
24            for(int i=0; i<20; i++)  
25            {  
26                c[2*i+1]+=a[2*i+1]*b[2*i+1];  
27            }  
28        }  
29  
30        #pragma omp section  
31        {  
32            #pragma omp parallel for  
33            for(int i=0; i<40; i++)  
34            {  
35                c[i]-=b[i];  
36            }  
37        }  
38  
39    for(int i=0; i<40; i++)  
40        printf("\nfinal sum = %d", c[i]);
```

4) listing of prime nos  $< n$

```

C l4_iii.c > main()
1 #include<stdio.h>
2 #include<omp.h>
3 #include<time.h>
4
5 int main()
6 {
7     int n=0;
8     int mynum=100000;
9     printf("\nenter value for upper limit of prime check: ");
10    scanf("%d", &mynum);
11    clock_t t;
12    t=clock();
13
14    #pragma omp parallel for reduction(+:n)
15    for(int i=2; i<mynum; i++)
16    {
17        int p=1;
18        for(int j=2; j<(i/2)+1; j++)
19        {
20            if(i%j==0)
21            {
22                p=0;
23                break;
24            }
25        }
26        n+=p;
27    }
28    t=clock()-t;
29    printf("\nThere are %d primes between 1 and %d", n, mynum);
30    double tt=((double)t/CLOCKS_PER_SEC);
31    printf("\ntime taken for computation : %f\n", tt);
32 }

```

```

advaite-vm@advaite-vm-VirtualBox: ~/Desktop/PDC/pdclab4$ gcc -o l44 -fopenmp l4_iii.c
advaite-vm@advaite-vm-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
advaite-vm@advaite-vm-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
advaite-vm@advaite-vm-VirtualBox: ~/Desktop/PDC/pdclab4$ #taking ~ 7 seconds irl, so 32 is counting total core time, not irl time

```

## 5) Sudoku solver (2x2x4)

```

C l4_iv.c > solvesud()
41 {
42     for(int i=0; i<4; i++)
43     {
44         for(int j=0; j<4; j++)
45         {
46             printf("%d ", grid[i][j]);
47         }
48         printf("\n");
49     }
50
51 void solvesud()
52 {
53     for(int x=0; x<4; x++)
54     {
55         for(int y=0; y<4; y++)
56         {
57             if(grid[x][y]==0)
58             {
59                 // #pragma omp parallel for
60                 for(int k=1; k<5; k++)
61                 {
62                     if(possible(x, y, k))
63                     {
64                         grid[x][y]=k;
65                         solvesud();
66                         grid[x][y]=0;
67                     }
68                 }
69                 return;
70             }
71         }
72     }
73     dispmat();
74 }

```

```

advaite-vm@advaite-vm-VirtualBox: ~/Desktop/PDC/pdclab4$ gcc -o l45 -fopenmp l4_iv.c
advaite-vm@advaite-vm-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
advaite-vm@advaite-vm-VirtualBox: ~/Desktop/PDC/pdclab4$ gcc -o l45 -fopenmp l4_iv.c
advaite-vm@advaite-vm-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
advaite-vm@advaite-vm-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);
```

```
}
```