

CSE222P112

Submitted by:-

Manorath Bajaj

14BCE0442

Question 4.21

Write a Multithreaded program that calculates the average minimum and maximum value from an array in different threads.

I have used 90 81 78 95 79 72 85 as the sample input. #include<stdio.h>

The code for the program is given Below:

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

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

```
// declaring the variables globally
```

```
int avg=0,min=0,max=0,data[7]={90,81,78,95,79,72,85};
```

```
//creating 3 functions to run in 3 diffrent threads
```

```
void *Calc_avg();
```

```
void *Calc_max();
```

```
void *Calc_min();
```

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

```
{
```

```
    pthread_t tid1,tid2,tid3;
```

```
    pthread_create(&tid1,NULL,Calc_avg,NULL);
```

```
    printf("after first thread\n");
```

```
    pthread_create(&tid2,NULL,Calc_max,NULL);
```

```
    printf("after second thread\n");
```

```
    pthread_create(&tid3,NULL,Calc_min,NULL);
```

```
    printf("after third thread\n");
```

```
    pthread_join(tid1,NULL);
```

```
    pthread_join(tid2,NULL);
```

```
    pthread_join(tid3,NULL);
```

```
// Printing in the main thread
```

```
    printf("avg value is : %d\n",avg);
```

```
    printf("min value is: %d\n",min);
```

```
    printf("max value is: %d\n",max);
```

```

}

void *Calc_avg()
{
    int i;
    for(i=0;i<7;i++)
        avg=avg+data[i];
    avg=avg/7;

}

void *Calc_max()
{
    int i;
    max=data[0];
    for(i=0;i<7;i++)
    {
        if(data[i]>max)
        {
            max=data[i];
        }
    }

}

}

void *Calc_min()
{
    int i;
    min=data[0];
    for(i=0;i<7;i++)
    {
        if(data[i]<min)
        {
            min=data[i];
        }
    }

}

}

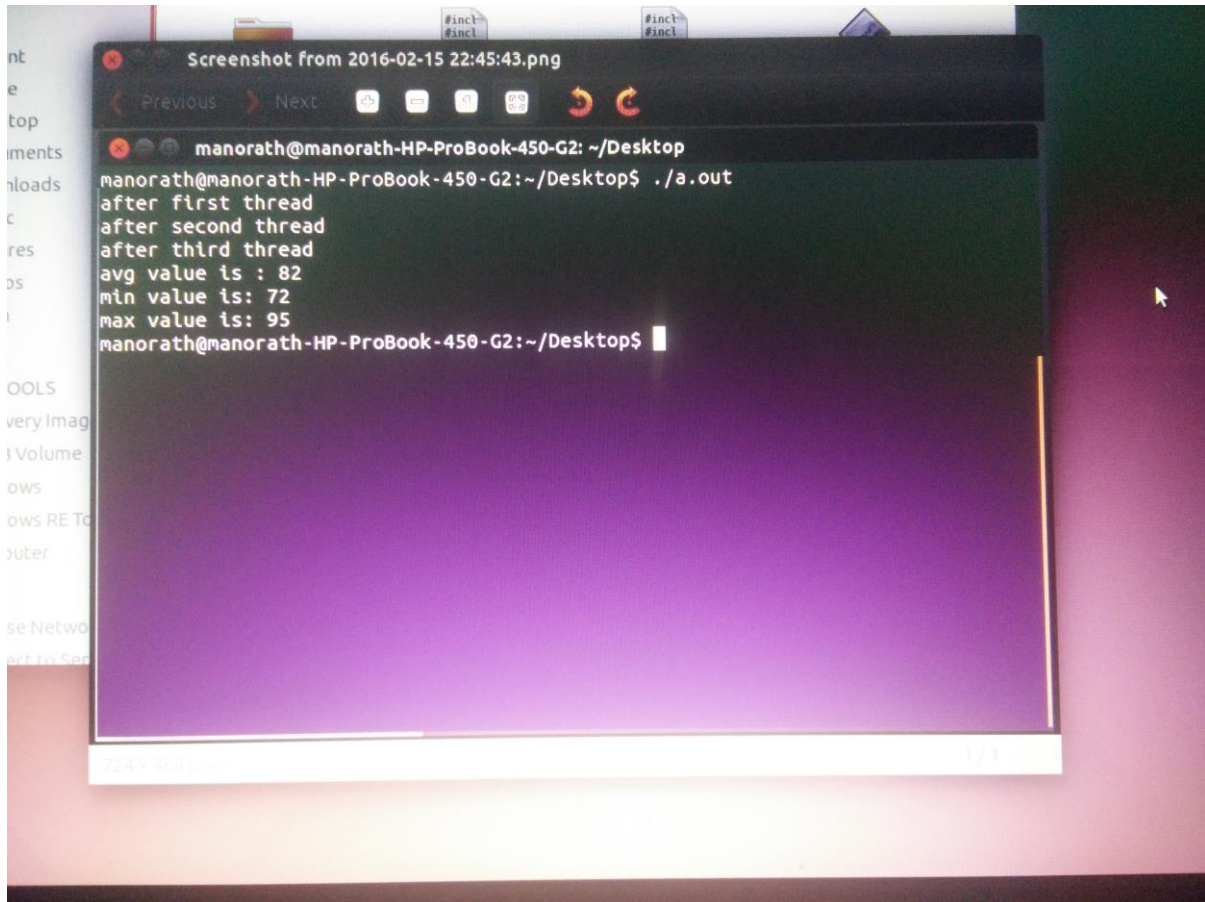
```

I made 3 different functions which were assigned to 3 different threads, and the result was printed from the main function.

OUTPUT:

A Screenshot of the output:

(using a mobile camera because I was having compatibility problems between OS's)



```
manorath@manorath-HP-ProBook-450-G2: ~/Desktop
manorath@manorath-HP-ProBook-450-G2:~/Desktop$ ./a.out
after first thread
after second thread
after third thread
avg value is : 82
min value is: 72
max value is: 95
manorath@manorath-HP-ProBook-450-G2:~/Desktop$
```

The Sudoku Validator

For the Sudoku Validator, I took a little different approach. My algorithm for checking and validating a Sudoku was based on the addition of numbers from 1 to 9 is 45. Rather than using loops, to decrease the time complexity, I added the summation of rows, columns and boxes(the last rule for Sudoku). Also I assigned a single row to a single thread. Hence The Validator uses 27 threads, 9 rows 9 columns and 9 boxes. There are only 3 functions in main as I have used main only as a driver function but 27 different functions are assigned to 27 different threads. This happens in the create () function. The join() function joins the threads and the final() function prints the final answer.The following Sudoku was used for input:

6	2	4	5	3	9	1	8	7
5	1	9	7	2	8	6	3	4
8	3	7	6	1	4	2	9	5
1	4	3	8	6	5	7	2	9
9	5	8	2	4	7	3	6	1
7	6	2	3	9	1	4	5	8
3	7	1	9	5	6	8	4	2
4	9	6	1	8	2	5	7	3
2	8	5	4	7	3	9	1	6

The Code:

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

```
#include<stdlib.h>
```

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

```
#include<string.h>
```

```
int a[9][9]={    {6,2,4,5,3,9,1,8,7},
                  {5,1,9,7,2,8,6,3,4},
                  {8,3,7,6,1,4,2,9,5},
                  {1,4,3,8,6,5,7,2,9},
                  {9,5,8,2,4,7,3,6,1},
                  {7,6,2,3,9,1,4,5,8},
                  {3,7,1,9,5,6,8,4,2},
                  {4,9,6,1,8,2,5,7,3},
                  {2,8,5,4,7,3,9,1,6}};
```

```
//init tid's globally in order to make a function to create threads
```

```
pthread_t
```

```
row0,row1,row2,row3,row4,row5,row6,row7,row8,col0,col1,col2,col3,col4,col5,col6,col7,col8;
```

```
pthread_t box0,box1,box2,box3,box4,box5,box6,box7,box8;
```

```
//flags to check
```

```
int frow[8],fcol[8],fbox[8];
```

```
//check functions
```

```
void *check_row0()
```

```
{  
    if(a[0][0]+a[0][1]+a[0][2]+a[0][3]+a[0][4]+a[0][5]+a[0][6]+a[0][7]+a[0][8]==45)  
        frow[0]=1;  
}
```

```
void *check_row1()
```

```
{  
    if(a[1][0]+a[1][1]+a[1][2]+a[1][3]+a[1][4]+a[1][5]+a[1][6]+a[1][7]+a[1][8]==45)  
        frow[1]=1;  
}
```

```
void *check_row2()
```

```
{  
    if(a[2][0]+a[2][1]+a[2][2]+a[2][3]+a[2][4]+a[2][5]+a[2][6]+a[2][7]+a[2][8]==45)  
        frow[2]=1;  
}
```

```
void *check_row3()
```

```
{  
    if(a[3][0]+a[3][1]+a[3][2]+a[3][3]+a[3][4]+a[3][5]+a[3][6]+a[3][7]+a[3][8]==45)  
        frow[3]=1;  
}
```

```
void *check_row4()
```

```
{  
    if(a[4][0]+a[4][1]+a[4][2]+a[4][3]+a[4][4]+a[4][5]+a[4][6]+a[4][7]+a[4][8]==45)  
        frow[4]=1;  
}
```

```
void *check_row5()
```

```

{
    if(a[5][0]+a[5][1]+a[5][2]+a[5][3]+a[5][4]+a[5][5]+a[5][6]+a[5][7]+a[5][8]==45)
        frow[5]=1;
}

void *check_row6()
{
    if(a[6][0]+a[6][1]+a[6][2]+a[6][3]+a[6][4]+a[6][5]+a[6][6]+a[6][7]+a[6][8]==45)
        frow[6]=1;
}

void *check_row7()
{
    if(a[7][0]+a[7][1]+a[7][2]+a[7][3]+a[7][4]+a[7][5]+a[7][6]+a[7][7]+a[7][8]==45)
        frow[7]=1;
}

void *check_row8()
{
    if(a[8][0]+a[8][1]+a[8][2]+a[8][3]+a[8][4]+a[8][5]+a[8][6]+a[8][7]+a[8][8]==45)
        frow[8]=1;
}

void *check_coloumn0()
{
    if(a[0][0]+a[1][0]+a[2][0]+a[3][0]+a[4][0]+a[5][0]+a[6][0]+a[7][0]+a[8][0]==45)
        fcol[0]=1;
}

void *check_coloumn1()
{
    if(a[0][1]+a[1][1]+a[2][1]+a[3][1]+a[4][1]+a[5][1]+a[6][1]+a[7][1]+a[8][1]==45)
        fcol[1]=1;
}

```

```

}

void *check_coloumn2()
{
    if(a[0][2]+a[1][2]+a[2][2]+a[3][2]+a[4][2]+a[5][2]+a[6][2]+a[7][2]+a[8][2]==45)
        fcol[2]=1;
}

void *check_coloumn3()
{
    if(a[0][3]+a[1][3]+a[2][3]+a[3][3]+a[4][3]+a[5][3]+a[6][3]+a[7][3]+a[8][3]==45)
        fcol[3]=1;
}

void *check_coloumn4()
{
    if(a[0][4]+a[1][4]+a[2][4]+a[3][4]+a[4][4]+a[5][4]+a[6][4]+a[7][4]+a[8][4]==45)
        fcol[4]=1;
}

void *check_coloumn5()
{
    if(a[0][5]+a[1][5]+a[2][5]+a[3][5]+a[4][5]+a[5][5]+a[6][5]+a[7][5]+a[8][5]==45)
        fcol[5]=1;
}

void *check_coloumn6()
{
    if(a[0][6]+a[1][6]+a[2][6]+a[3][6]+a[4][6]+a[5][6]+a[6][6]+a[7][6]+a[8][6]==45)
        fcol[6]=1;
}

void *check_coloumn7()
{

```

```

        if(a[0][7]+a[1][7]+a[2][7]+a[3][7]+a[4][7]+a[5][7]+a[6][7]+a[7][7]+a[8][7]==45)
            fcol[7]=1;
    }

void *check_coloumn8()
{
    if(a[0][8]+a[1][8]+a[2][8]+a[3][8]+a[4][8]+a[5][8]+a[6][8]+a[7][8]+a[8][8]==45)
        fcol[8]=1;
}

void *check_box0()
{
    if(a[0][0]+a[0][1]+a[0][2]+a[1][0]+a[1][1]+a[1][2]+a[2][0]+a[2][1]+a[2][2]==45)
        fbox[0]=1;
}

void *check_box1()
{
    if(a[0][3]+a[0][4]+a[0][5]+a[1][3]+a[1][4]+a[1][5]+a[2][3]+a[2][4]+a[2][5]==45)
        fbox[1]=1;
}

void *check_box2()
{
    if(a[0][6]+a[0][7]+a[0][8]+a[1][6]+a[1][7]+a[1][8]+a[2][6]+a[2][7]+a[2][8]==45)
        fbox[2]=1;
}

void *check_box3()
{
    if(a[3][0]+a[3][1]+a[3][2]+a[4][0]+a[4][1]+a[4][2]+a[5][0]+a[5][1]+a[5][2]==45)
        fbox[3]=1;
}

```



```

void *check_box4()
{
    if(a[3][3]+a[3][4]+a[3][5]+a[4][3]+a[4][4]+a[4][5]+a[5][3]+a[5][4]+a[5][5]==45)
        fbox[4]=1;
}

void *check_box5()
{
    if(a[3][6]+a[3][7]+a[3][8]+a[4][6]+a[4][7]+a[4][8]+a[5][6]+a[5][7]+a[5][8]==45)
        fbox[5]=1;
}

void *check_box6()
{
    if(a[6][0]+a[6][1]+a[6][2]+a[7][0]+a[7][1]+a[7][2]+a[8][0]+a[8][1]+a[8][2]==45)
        fbox[6]=1;
}

void *check_box7()
{
    if(a[6][3]+a[6][4]+a[6][5]+a[7][3]+a[7][4]+a[7][5]+a[8][3]+a[8][4]+a[8][5]==45)
        fbox[7]=1;
}

void *check_box8()
{
    if(a[6][6]+a[6][7]+a[6][8]+a[7][6]+a[7][7]+a[7][8]+a[8][6]+a[8][7]+a[8][8]==45)
        fbox[8]=1;
}

//create function
void create()
{

```

```
pthread_create(&row0,NULL,check_row0,NULL);
pthread_create(&row1,NULL,check_row1,NULL);
pthread_create(&row2,NULL,check_row2,NULL);
pthread_create(&row3,NULL,check_row3,NULL);
pthread_create(&row4,NULL,check_row4,NULL);
pthread_create(&row5,NULL,check_row5,NULL);
pthread_create(&row6,NULL,check_row6,NULL);
pthread_create(&row7,NULL,check_row7,NULL);
pthread_create(&row8,NULL,check_row8,NULL);
pthread_create(&col0,NULL,check_coloumn0,NULL);
pthread_create(&col1,NULL,check_coloumn1,NULL);
pthread_create(&col2,NULL,check_coloumn2,NULL);
pthread_create(&col3,NULL,check_coloumn3,NULL);
pthread_create(&col4,NULL,check_coloumn4,NULL);
pthread_create(&col5,NULL,check_coloumn5,NULL);
pthread_create(&col6,NULL,check_coloumn6,NULL);
pthread_create(&col7,NULL,check_coloumn7,NULL);
pthread_create(&col8,NULL,check_coloumn8,NULL);
pthread_create(&box0,NULL,check_box0,NULL);
pthread_create(&box1,NULL,check_box1,NULL);
pthread_create(&box2,NULL,check_box2,NULL);
pthread_create(&box3,NULL,check_box3,NULL);
pthread_create(&box4,NULL,check_box4,NULL);
pthread_create(&box5,NULL,check_box5,NULL);
pthread_create(&box6,NULL,check_box6,NULL);
pthread_create(&box7,NULL,check_box7,NULL);
pthread_create(&box8,NULL,check_box8,NULL);
```

```
}
```

```
// join function

void join()
{
    int i;

    pthread_join(row0,NULL);
    pthread_join(row2,NULL);
    pthread_join(row3,NULL);
    pthread_join(row4,NULL);
    pthread_join(row5,NULL);
    pthread_join(row6,NULL);
    pthread_join(row7,NULL);
    pthread_join(row8,NULL);
    pthread_join(col1,NULL);
    pthread_join(col2,NULL);
    pthread_join(col3,NULL);
    pthread_join(col4,NULL);
    pthread_join(col5,NULL);
    pthread_join(col6,NULL);
    pthread_join(col7,NULL);
    pthread_join(col8,NULL);
    pthread_join(box0,NULL);
    pthread_join(box1,NULL);
    pthread_join(box2,NULL);
    pthread_join(box3,NULL);
    pthread_join(box4,NULL);
    pthread_join(box5,NULL);
    pthread_join(box6,NULL);
```

```

        pthread_join(box7,NULL);
        pthread_join(box8,NULL);
    }

void final()
{
    int flag1=0,flag2=0,flag3=0,i;
    for(i=0;i<9;i++)
    {
        if(frow[i]!=1)
            {flag1=1;}
        if(fcol[i]!=1)
            {flag2=1;}
        if(fbox[i]!=1)
            {flag3=1;}
    }
    if(flag1==0 && flag2==0 && flag3==0)
        { printf("The Given Sudoku is correct\n");}
    else
        printf("The solution is invalid\n");
}

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

    create();

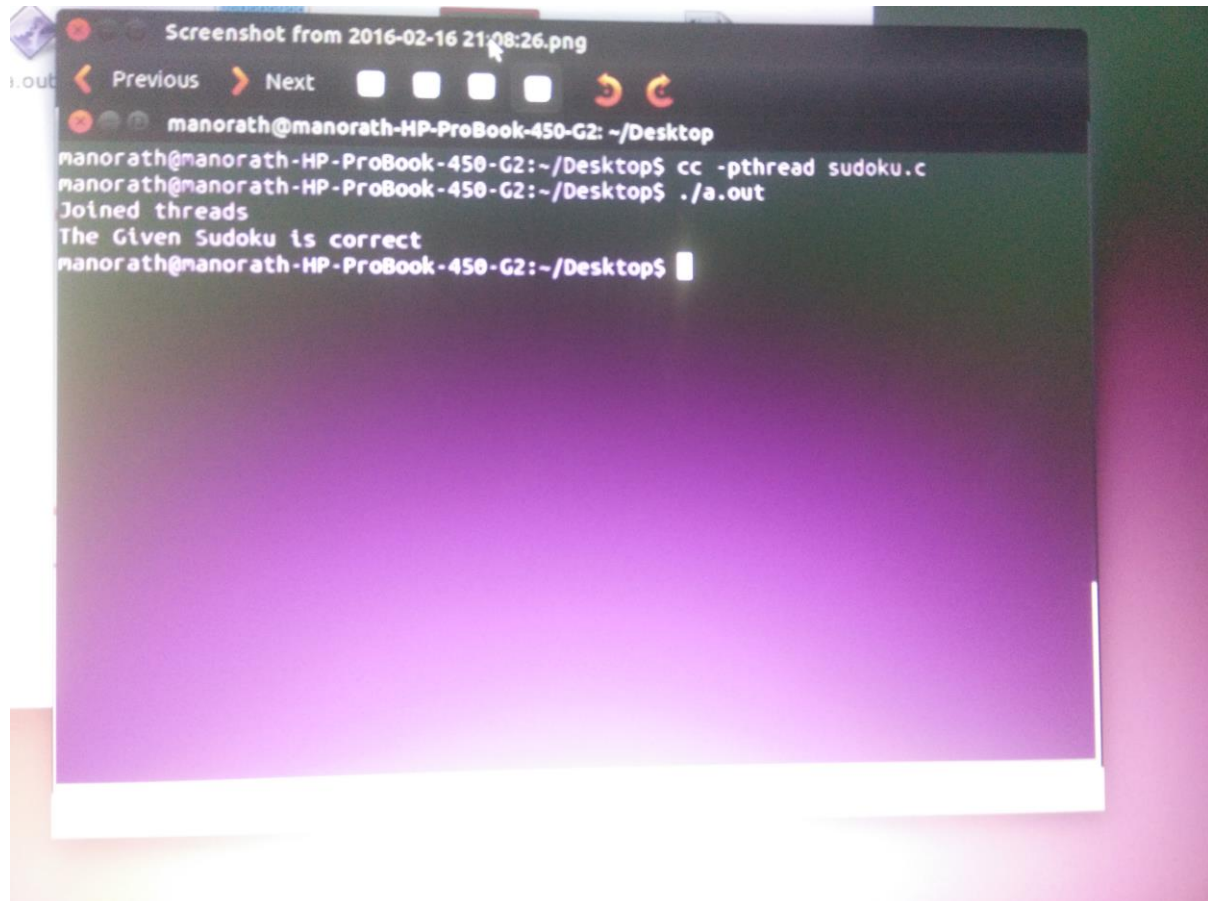
    join();

    printf("Joined threads\n");

```

```
sleep(5);  
final();  
return 0;  
}
```

The Output



```
manorath@manorath-HP-ProBook-450-G2: ~/Desktop$ cc -pthread sudoku.c  
manorath@manorath-HP-ProBook-450-G2: ~/Desktop$ ./a.out  
Joined threads  
The Given Sudoku is correct  
manorath@manorath-HP-ProBook-450-G2: ~/Desktop$
```