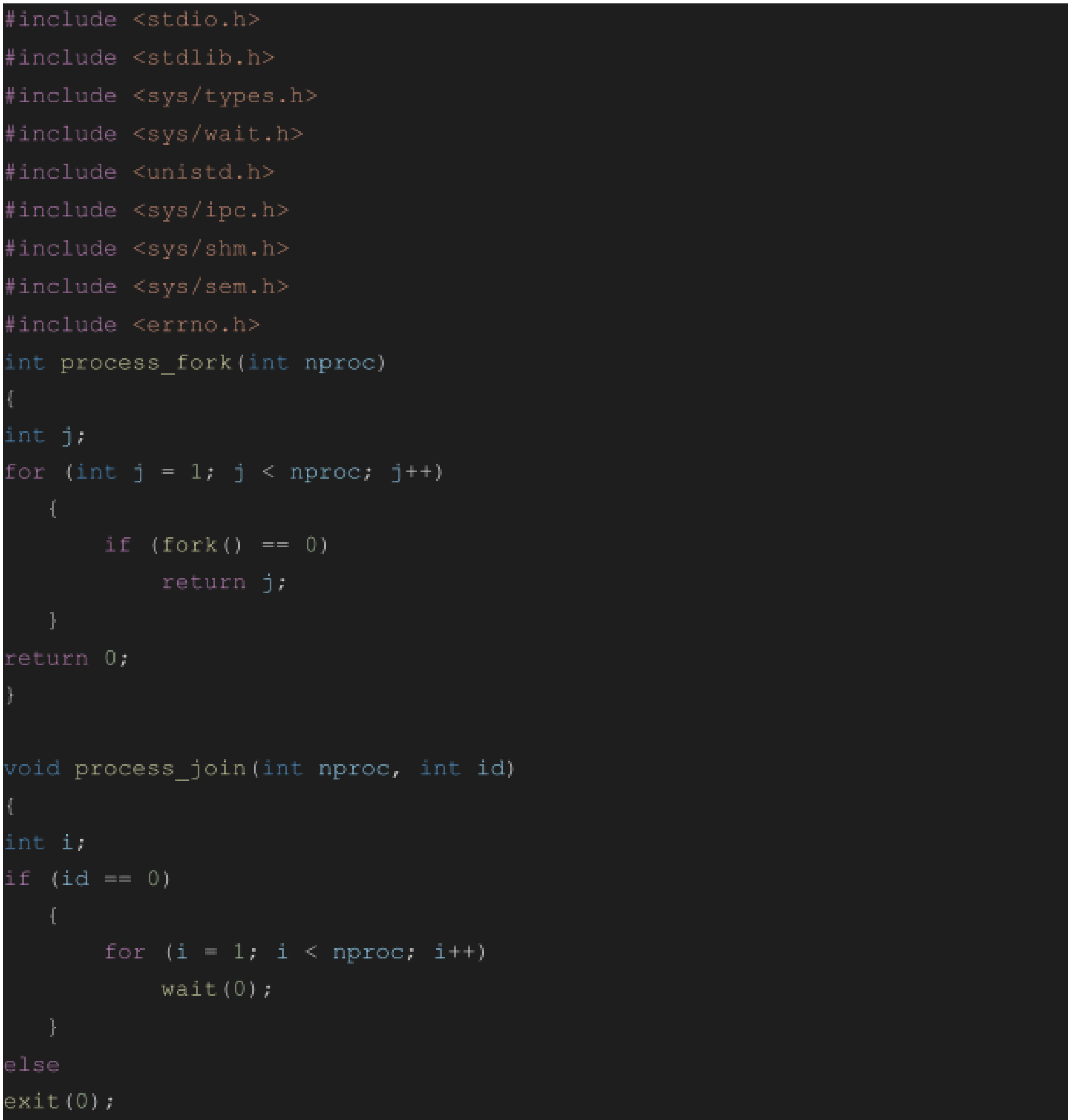
Name :- Nisarg Amlani
Roll :- CE001
Id :- 22ceueg082

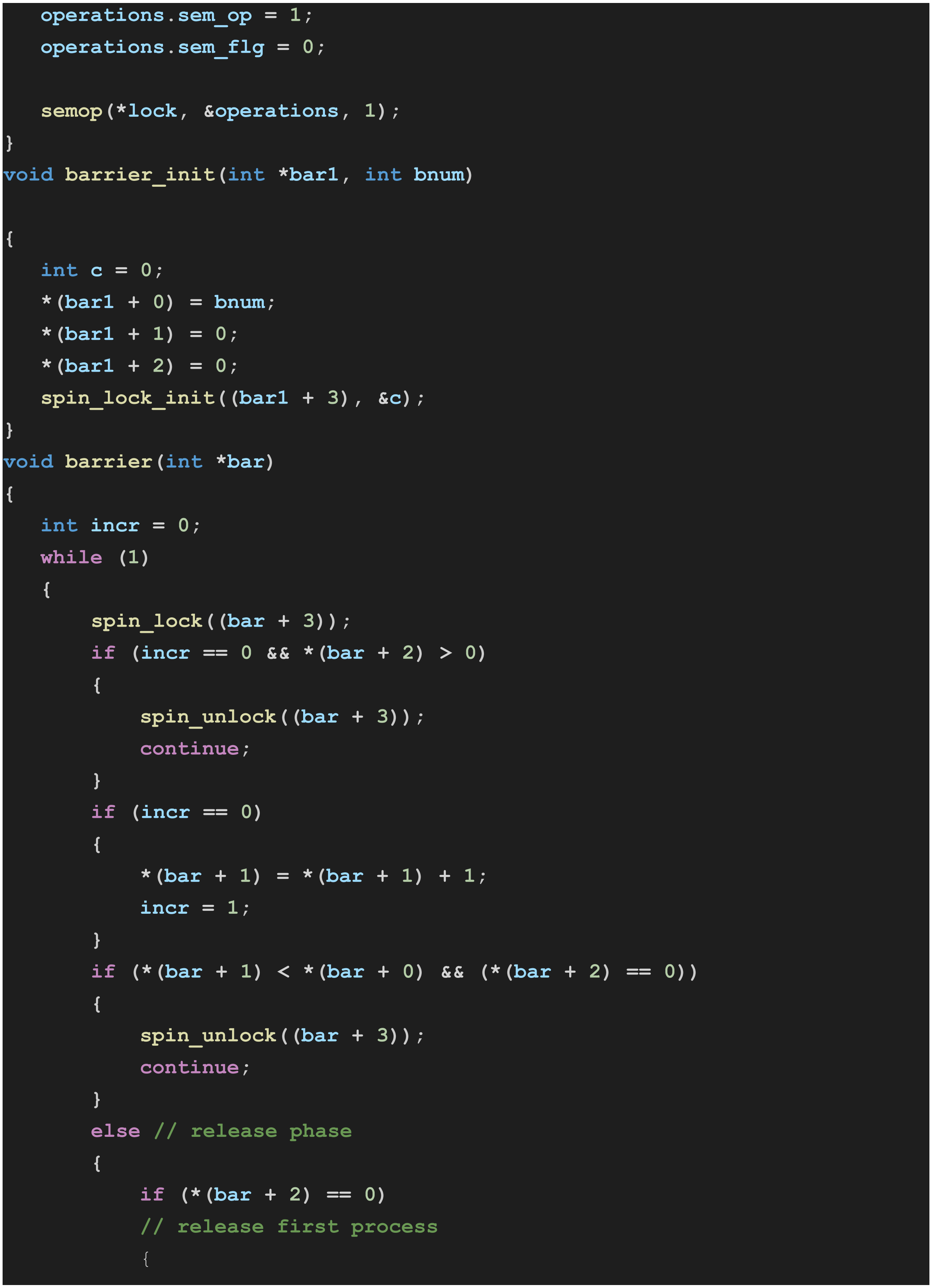
LAB-12

1) To the header.h file created in LAB 10 include barrier_init() and barrier() function.

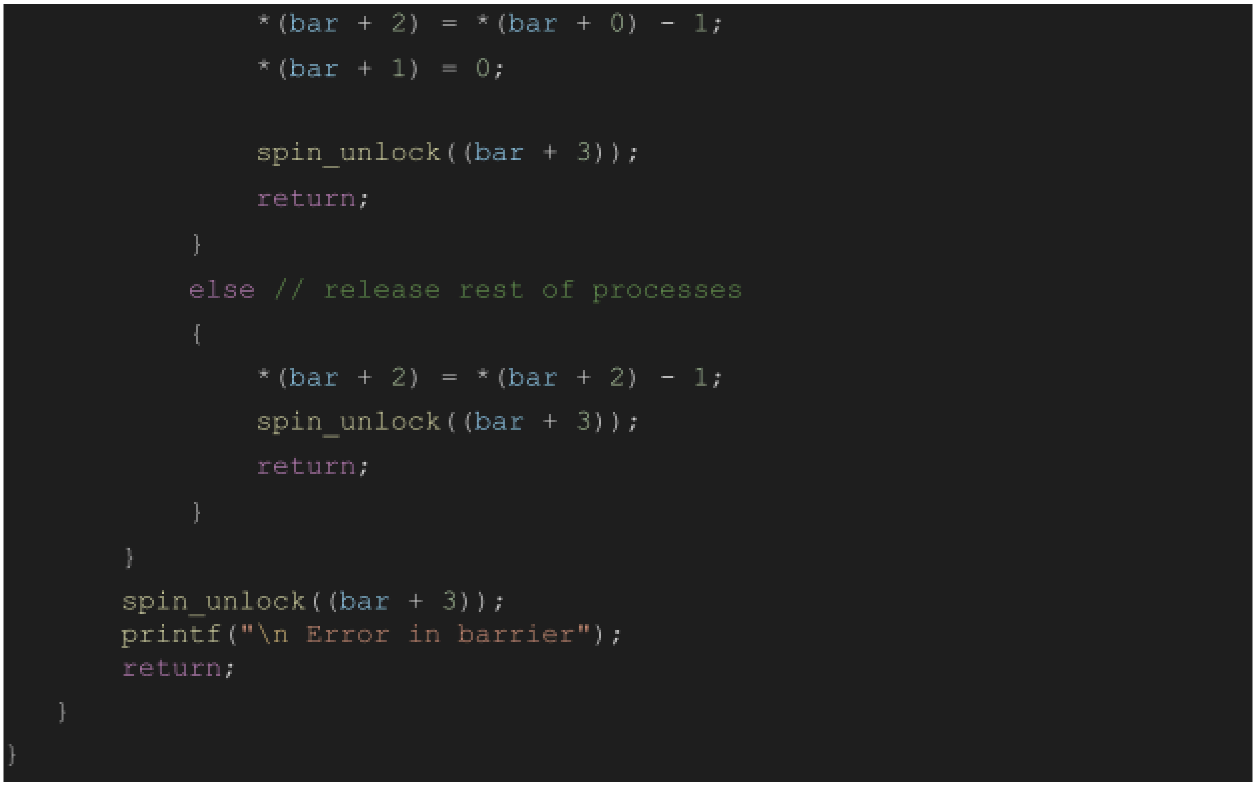


```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/sem.h>
#include <errno.h>
int process_fork(int nproc)
{
int j;
for (int j = 1; j < nproc; j++)
   {
       if (fork() == 0)
           return j;
   }
return 0;
}

void process_join(int nproc, int id)
{
int i;
if (id == 0)
   {
       for (i = 1; i < nproc; i++)
           wait(0);
   }
else
exit(0);
```

```
{
    *shmid = shmget(IPC_PRIVATE, size, 0666 | IPC_CREAT);
    if (*shmid == -1)
    {
        perror("shmget failed");
        exit(1);
    }
    char *shm_addr = (char *)shmat(*shmid, 0, 0);
    if (shm_addr == (char *)-1)
    {
        perror("shmat failed");
        exit(1);
    }
    return shm_addr;
}

void spin_lock_init(int *lock, int *condition)

{
    int control;
    *lock = semget(IPC_PRIVATE, 1, 0666 | IPC_CREAT);
    if (*condition == 1)
        control = 0;
    else
        control = 1;

    semctl(*lock, 0, SETVAL, control);
}
void spin_lock(int *lock)
{
    struct sembuf operations;
    operations.sem_num = 0;
    operations.sem_op = -1;
    operations.sem_flg = 0;
    semop(*lock, &operations, 1);
}
void spin_unlock(int *lock)
{
    struct sembuf operations;
    operations.sem_num = 0;
```



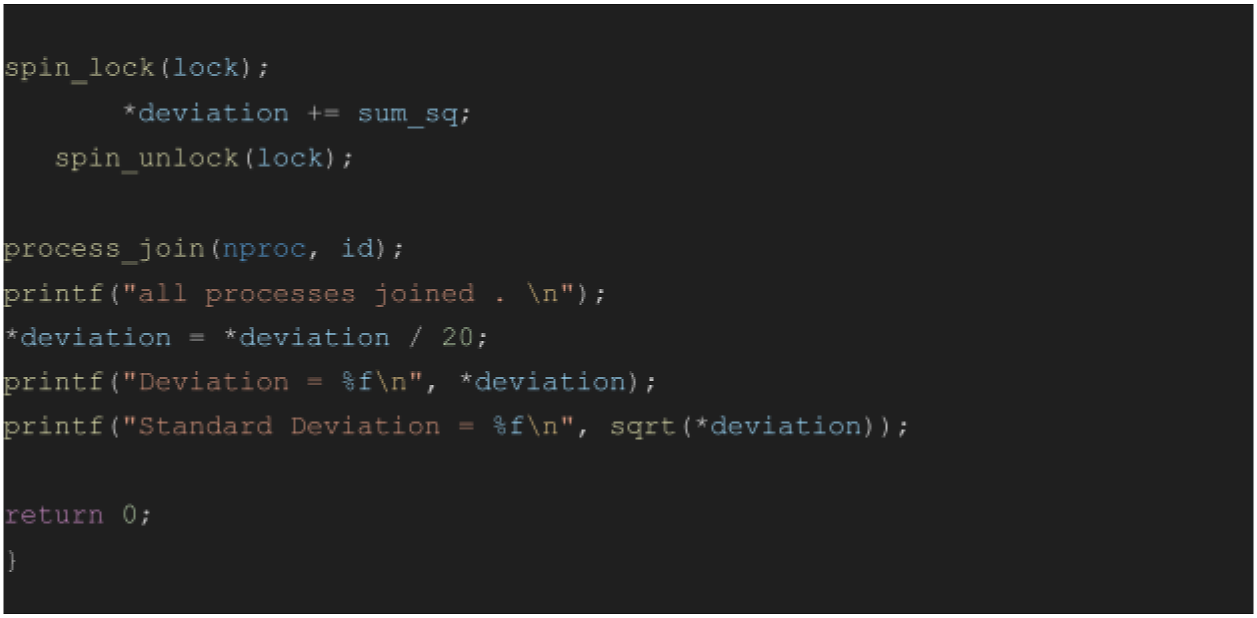
```
    operations.sem_op = 1;
    operations.sem_flg = 0;

    semop(*lock, &operations, 1);
}
void barrier_init(int *bar1, int bnum)

{
    int c = 0;
    *(bar1 + 0) = bnum;
    *(bar1 + 1) = 0;
    *(bar1 + 2) = 0;
    spin_lock_init((bar1 + 3), &c);
}
void barrier(int *bar)
{
    int incr = 0;
    while (1)
    {
        spin_lock((bar + 3));
        if (incr == 0 && *(bar + 2) > 0)
        {
            spin_unlock((bar + 3));
            continue;
        }
        if (incr == 0)
        {
            *(bar + 1) = *(bar + 1) + 1;
            incr = 1;
        }
        if (*(bar + 1) < *(bar + 0) && (*(bar + 2) == 0))
        {
            spin_unlock((bar + 3));
            continue;
        }
        else // release phase
        {
            if (*(bar + 2) == 0)
            // release first process
            {
```



```
                *(bar + 2) = *(bar + 0) - 1;
                *(bar + 1) = 0;

                spin_unlock((bar + 3));
                return;
            }
            else // release rest of processes
            {
                *(bar + 2) = *(bar + 2) - 1;
                spin_unlock((bar + 3));
                return;
            }
        }
        spin_unlock((bar + 3));
        printf("\n Error in barrier");
        return;
    }
}
```

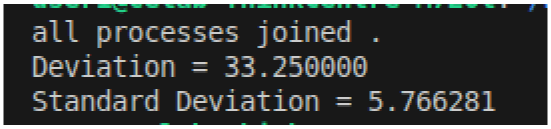
2)WAP to Calculate standard deviation with help of barrier

```
#include "header.h"
#include <math.h>

#define nproc 4

int main()
{
   float a[20];
   for (int i=0; i<20; i++)
       a[i] = i;

     int shmid, *lock, *bar;
   lock = (int *)shared(sizeof(int), &shmid);
   bar = (int *)shared(sizeof(int) * 4, &shmid);
   spin_lock_init(lock, &(int){0});
   barrier_init(bar, nproc);
   float *avg, *deviation, sum = 0.0, sum_sq = 0.0;
   deviation = (float *)shared(sizeof(float), &shmid);
   avg = (float *)shared(sizeof(float), &shmid);
   *avg = 0.0;
   *deviation = 0.0;

   int id = process_fork(nproc);

   for (int i=id; i<20; i+=nproc){
       sum = sum + a[i];
   }
   spin_lock(lock);
       *avg = *avg + sum/20;
   spin_unlock(lock);

   barrier(bar);

   sum = 0.0;
   for(int i=id; i<20; i+=nproc)
       sum_sq += pow(a[i] - *avg, 2);
```



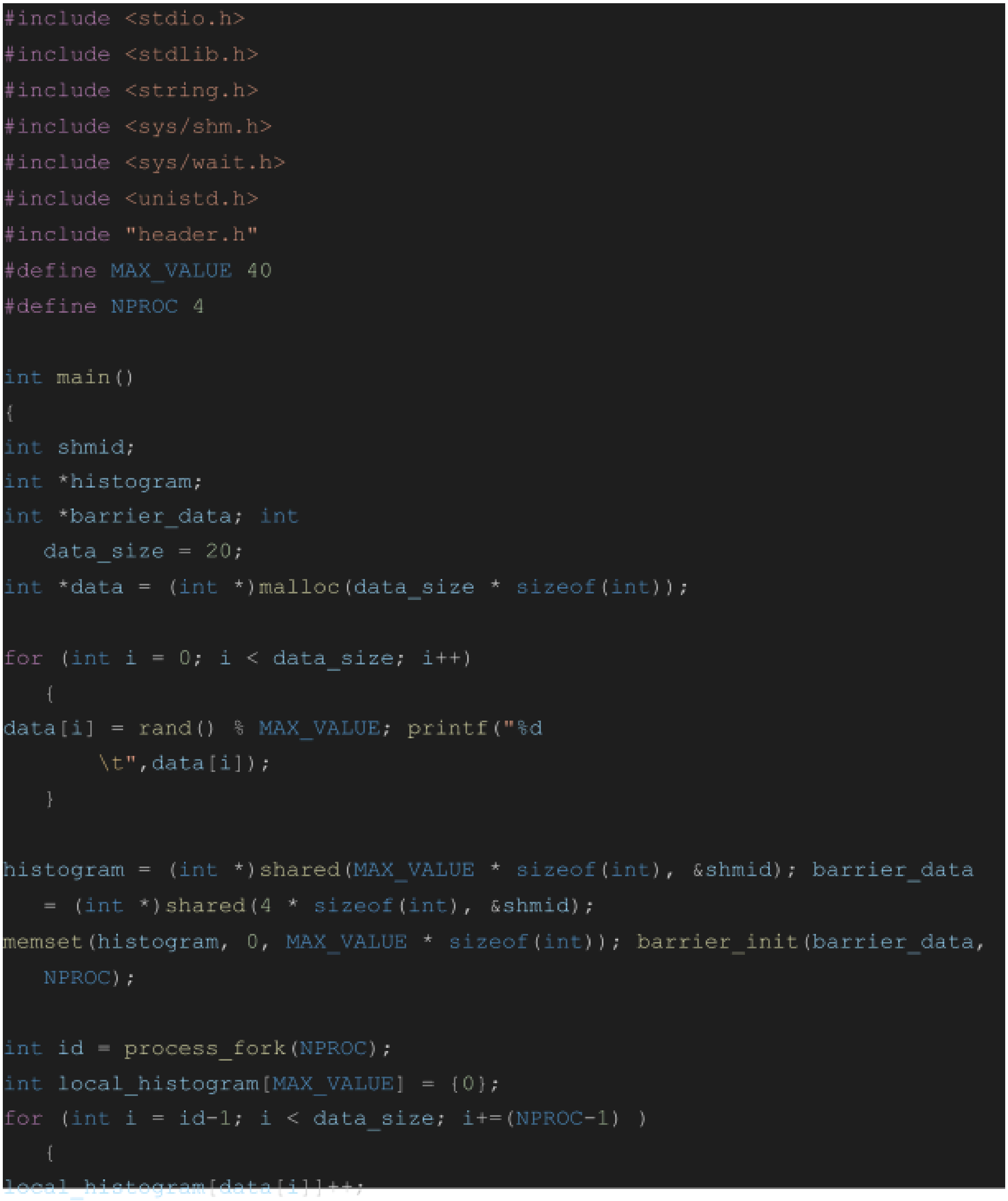
```
spin_lock(lock);
       *deviation += sum_sq;
   spin_unlock(lock);

process_join(nproc, id);
printf("all processes joined . \n");
*deviation = *deviation / 20;
printf("Deviation = %f\n", *deviation);
printf("Standard Deviation = %f\n", sqrt(*deviation));

return 0;
}
```

Output:

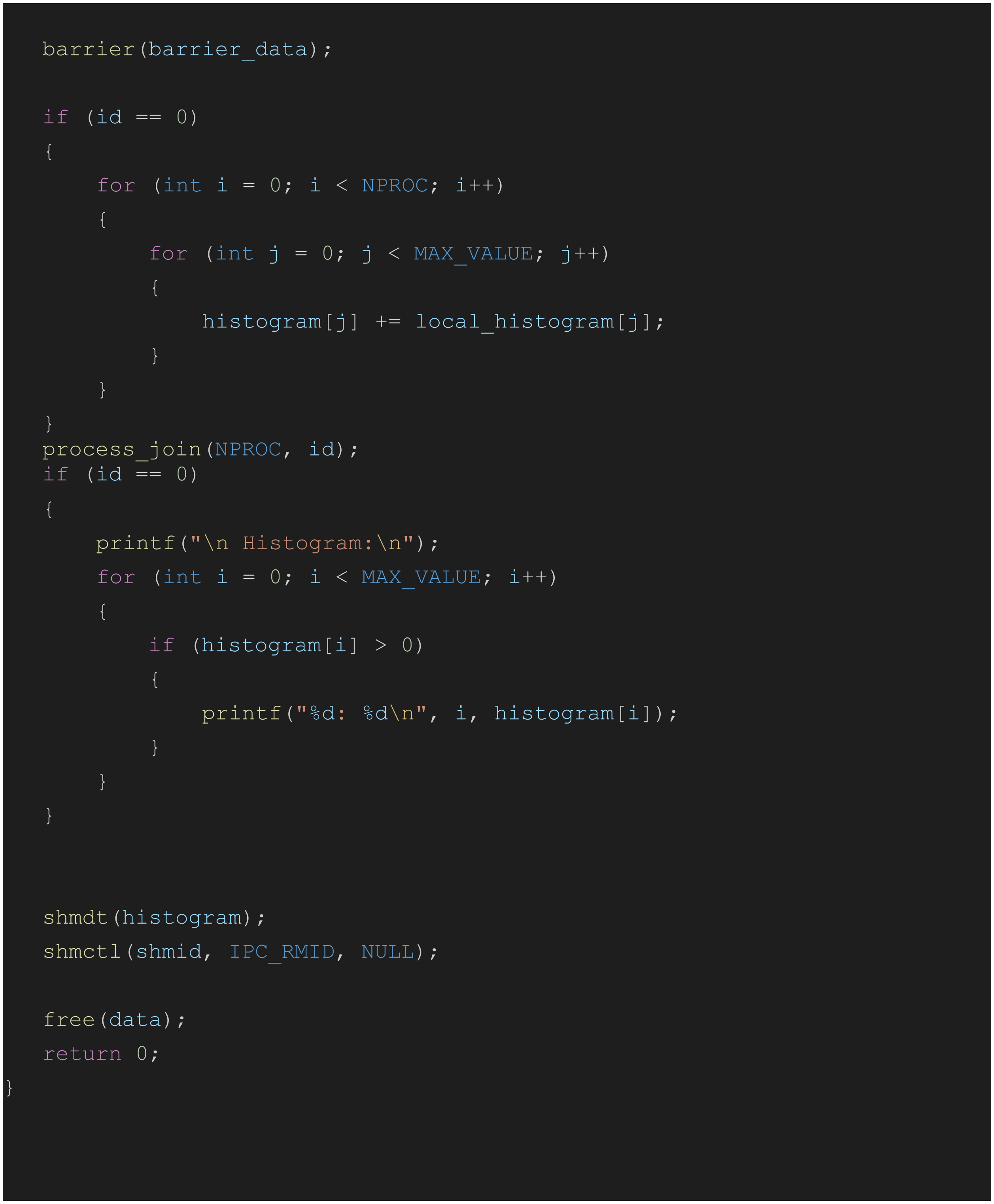


```
 all processes joined .
 Deviation = 33.250000
 Standard Deviation = 5.766281
```

3)Write a parallel program to implement Histogram with barrier.

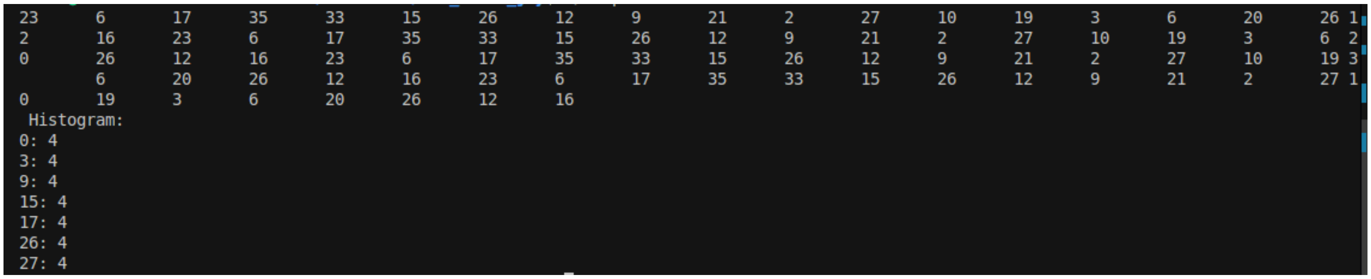


```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/shm.h>
#include <sys/wait.h>
#include <unistd.h>
#include "header.h"
#define MAX_VALUE 40
#define NPROC 4

int main()
{
int shmid;
int *histogram;
int *barrier_data; int
   data_size = 20;
int *data = (int *)malloc(data_size * sizeof(int));

for (int i = 0; i < data_size; i++)
   {
data[i] = rand() % MAX_VALUE; printf("%d
       \t",data[i]);
   }

histogram = (int *)shared(MAX_VALUE * sizeof(int), &shmid); barrier_data
   = (int *)shared(4 * sizeof(int), &shmid);
memset(histogram, 0, MAX_VALUE * sizeof(int)); barrier_init(barrier_data,
   NPROC);

int id = process_fork(NPROC);
int local_histogram[MAX_VALUE] = {0};
for (int i = id-1; i < data_size; i+=(NPROC-1) )
   {
local_histogram[data[i]]++;
```



```
    barrier(barrier_data);

    if (id == 0)
    {
        for (int i = 0; i < NPROC; i++)
        {
            for (int j = 0; j < MAX_VALUE; j++)
            {
                histogram[j] += local_histogram[j];
            }
        }
    }
    process_join(NPROC, id);
    if (id == 0)
    {
        printf("\n Histogram:\n");
        for (int i = 0; i < MAX_VALUE; i++)
        {
            if (histogram[i] > 0)
            {
                printf("%d: %d\n", i, histogram[i]);
            }
        }
    }



    shmdt(histogram);
    shmctl(shmid, IPC_RMID, NULL);

    free(data);
    return 0;
}
```

Output:



```
23      6       17      35      33      15      26      12      9       21      2       27      10      19      3       6       20      26 1
2       16      23      6       17      35      33      15      26      12      9       21      2       27      10      19      3       6  2
0       26      12      16      23      6       17      35      33      15      26      12      9       21      2       27      10      19 3
        6       20      26      12      16      23      6       17      35      33      15      26      12      9       21      2       27 1
0       19      3       6       20      26      12      16
 Histogram:
0: 4
3: 4
9: 4
15: 4
17: 4
26: 4
27: 4
```