Name : Nisarg .k. Amlani
Roll : Ce001
Id : 22ceueg082

Lab : 11

---

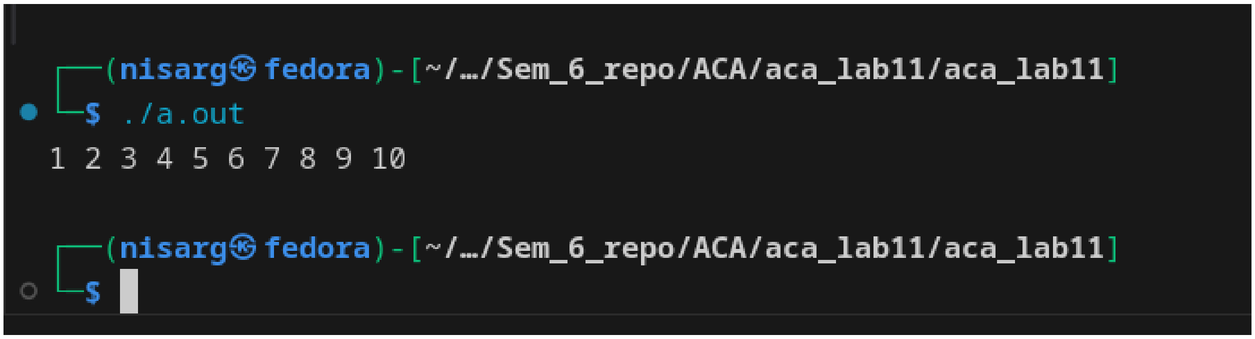
1. To the header.h file created in LAB 10 include spin_lock_init() and spin_lock() and spin_unlock() functions.

```
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#include <stdlib.h>
#include <semaphore.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/sem.h>

int process_fork(int nproc)
{

    int j;
    for (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);
}

char *shared(int size, int *shmid)
{

    *shmid = shmget(IPC_PRIVATE, size, 0666 | IPC_CREAT);
    return (shmat(*shmid, 0, 0));
}

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);
}
```

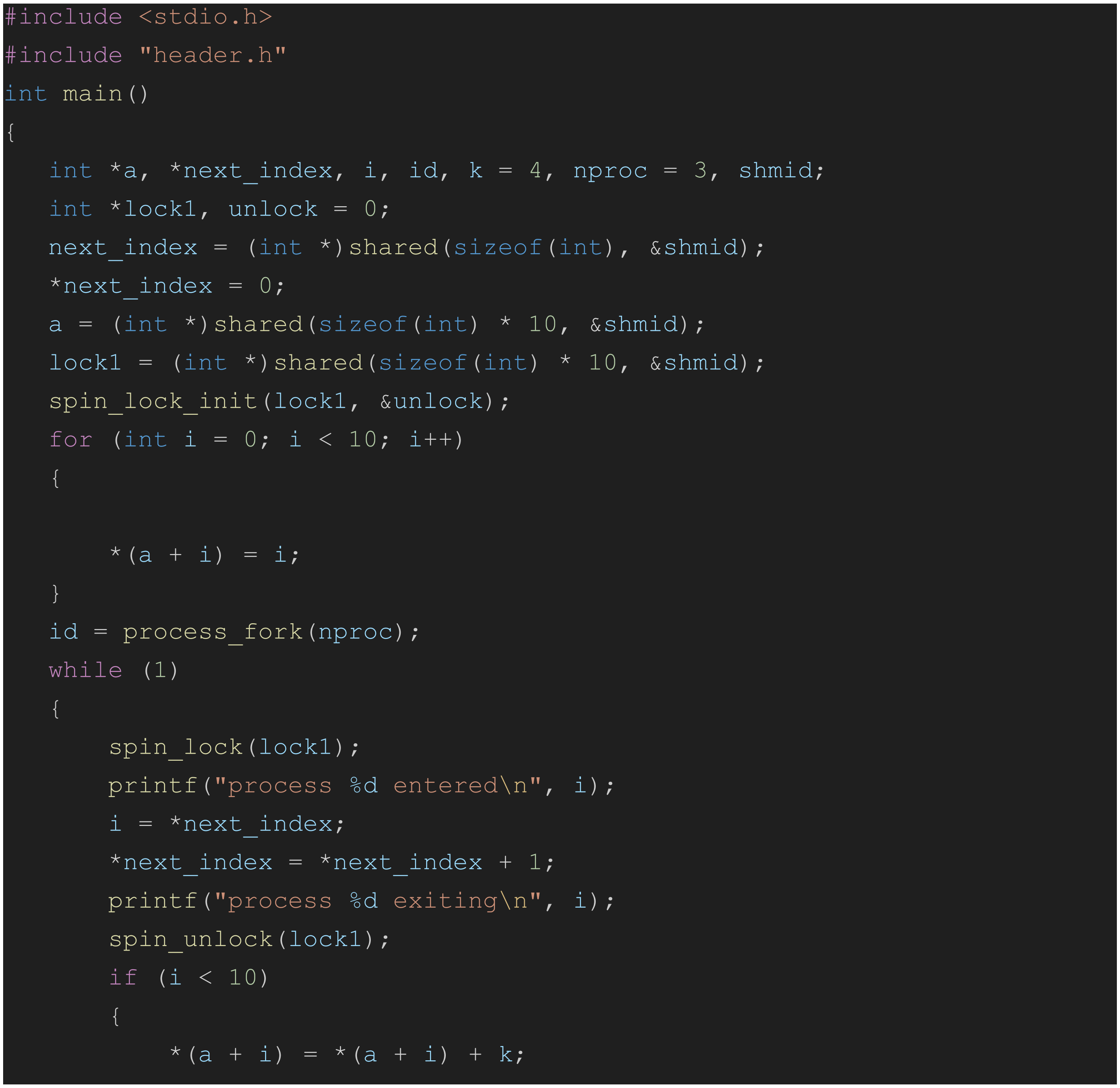
2. WAP to add constant to array using self-scheduling.

```
#include "header.h"

int main(){
    int shmid;
    int *arr = (int*)shared(10*sizeof(int),&shmid);
    int n = 10;
    for(int i = 0 ; i < n ; i++)
        arr[i] = i;
    int n_proc = 4;
    int i = 0;
    int *index = (int*)shared(sizeof(int),&shmid);
    *index = 0;

    int id = process_fork(n_proc);
    while(1){
        i = *index;
        (*index)++;
        if(i >= n)
            break;
        arr[i] = arr[i] + 1;
           }
    process_join(n_proc,id);

    for(i = 0 ; i < n ; i++)
        printf("%d ",arr[i]);
    printf("\n");
}
```

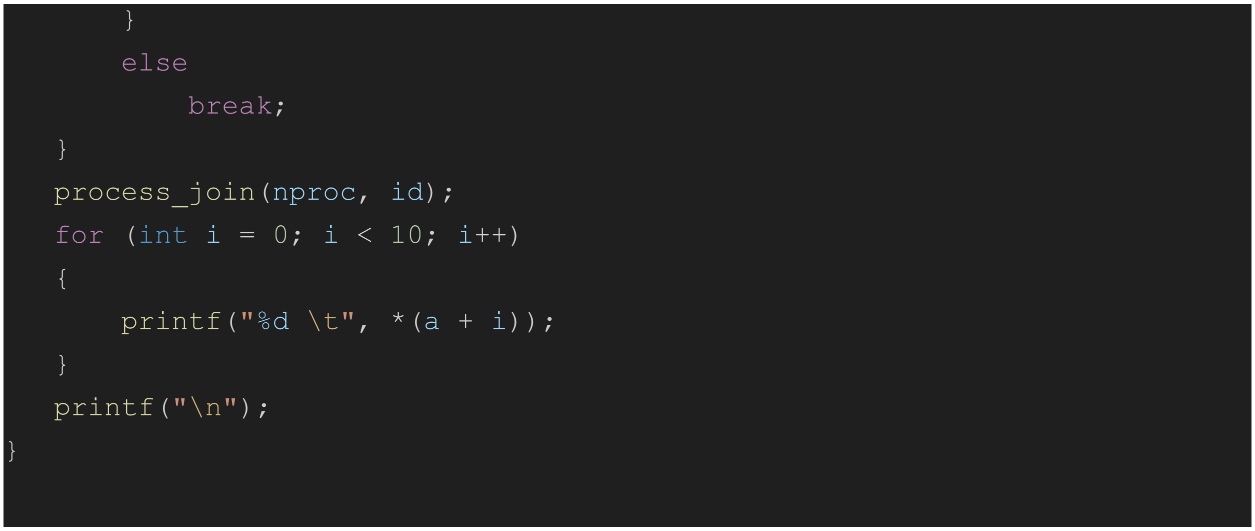


```
┌──(nisarg㉿fedora)-[~/…/Sem_6_repo/ACA/aca_lab11/aca_lab11]
└─$ ./a.out
1 2 3 4 5 6 7 8 9 10

┌──(nisarg㉿fedora)-[~/…/Sem_6_repo/ACA/aca_lab11/aca_lab11]
└─$ 
```

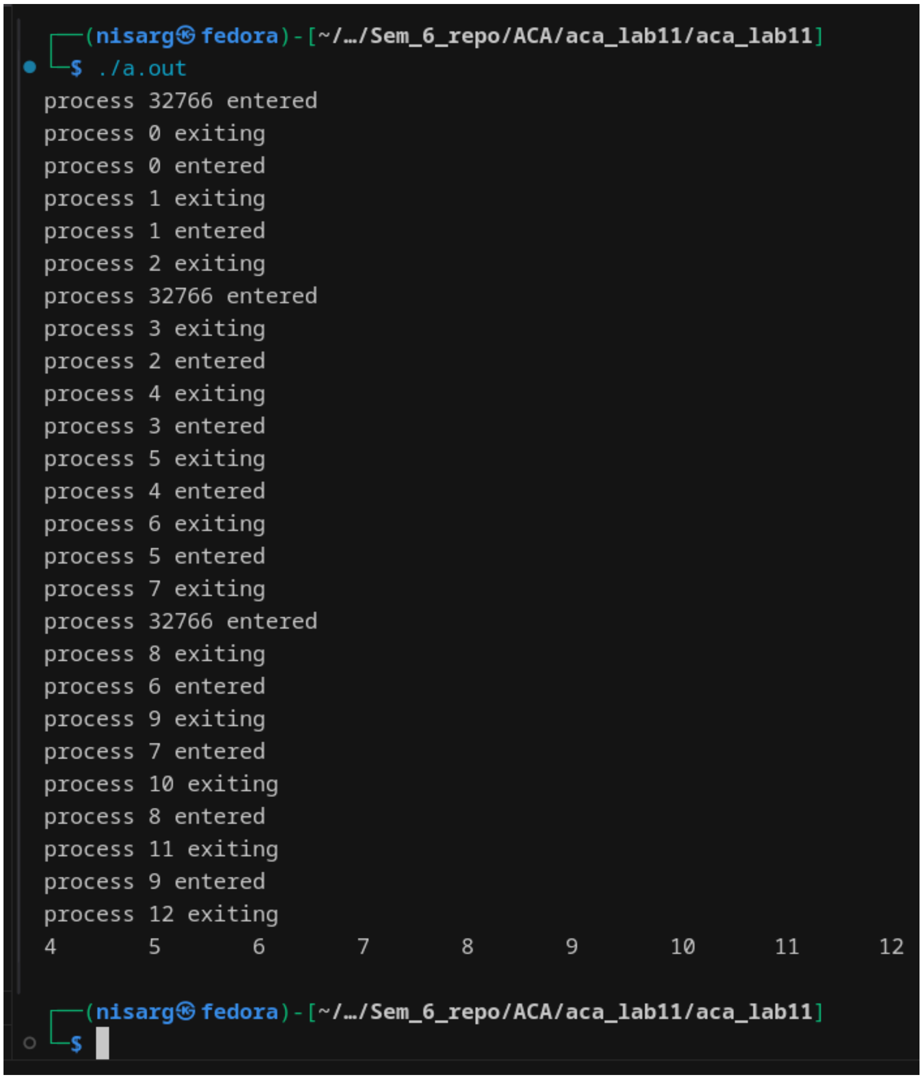
3. Write a parallel program to implement program 2 with locking



```c
#include <stdio.h>
#include "header.h"
int main()
{
   int *a, *next_index, i, id, k = 4, nproc = 3, shmid;
   int *lock1, unlock = 0;
   next_index = (int *)shared(sizeof(int), &shmid);
   *next_index = 0;
   a = (int *)shared(sizeof(int) * 10, &shmid);
   lock1 = (int *)shared(sizeof(int) * 10, &shmid);
   spin_lock_init(lock1, &unlock);
   for (int i = 0; i < 10; i++)
   {

       *(a + i) = i;
   }
   id = process_fork(nproc);
   while (1)
   {
       spin_lock(lock1);
       printf("process %d entered\n", i);
       i = *next_index;
       *next_index = *next_index + 1;
       printf("process %d exiting\n", i);
       spin_unlock(lock1);
       if (i < 10)
       {
           *(a + i) = *(a + i) + k;
```



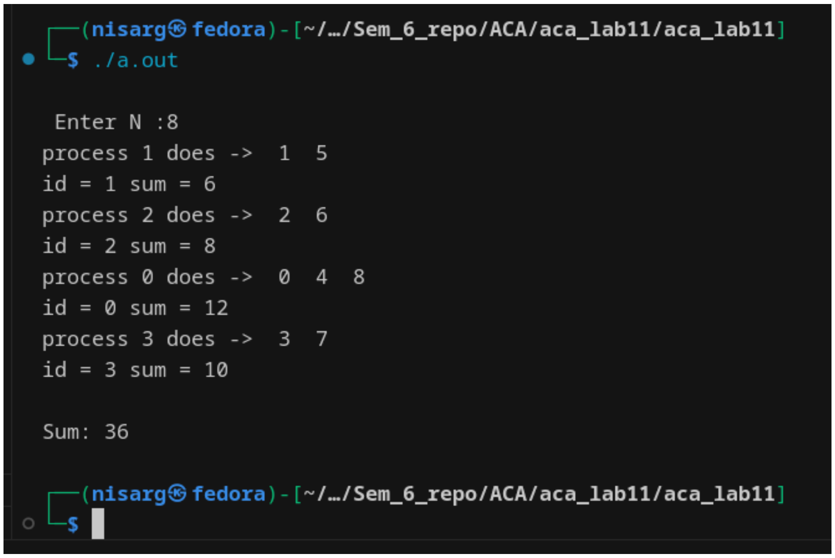
```
        }
        else
            break;
    }
    process_join(nproc, id);
    for (int i = 0; i < 10; i++)
    {
        printf("%d \t", *(a + i));
    }
    printf("\n");
}
```



```
┌──(nisarg㉿fedora)-[~/…/Sem_6_repo/ACA/aca_lab11/aca_lab11]
└─$ ./a.out
process 32766 entered
process 0 exiting
process 0 entered
process 1 exiting
process 1 entered
process 2 exiting
process 32766 entered
process 3 exiting
process 2 entered
process 4 exiting
process 3 entered
process 5 exiting
process 4 entered
process 6 exiting
process 5 entered
process 7 exiting
process 32766 entered
process 8 exiting
process 6 entered
process 9 exiting
process 7 entered
process 10 exiting
process 8 entered
process 11 exiting
process 9 entered
process 12 exiting
4       5       6       7       8       9       10      11      12

┌──(nisarg㉿fedora)-[~/…/Sem_6_repo/ACA/aca_lab11/aca_lab11]
└─$ 
```

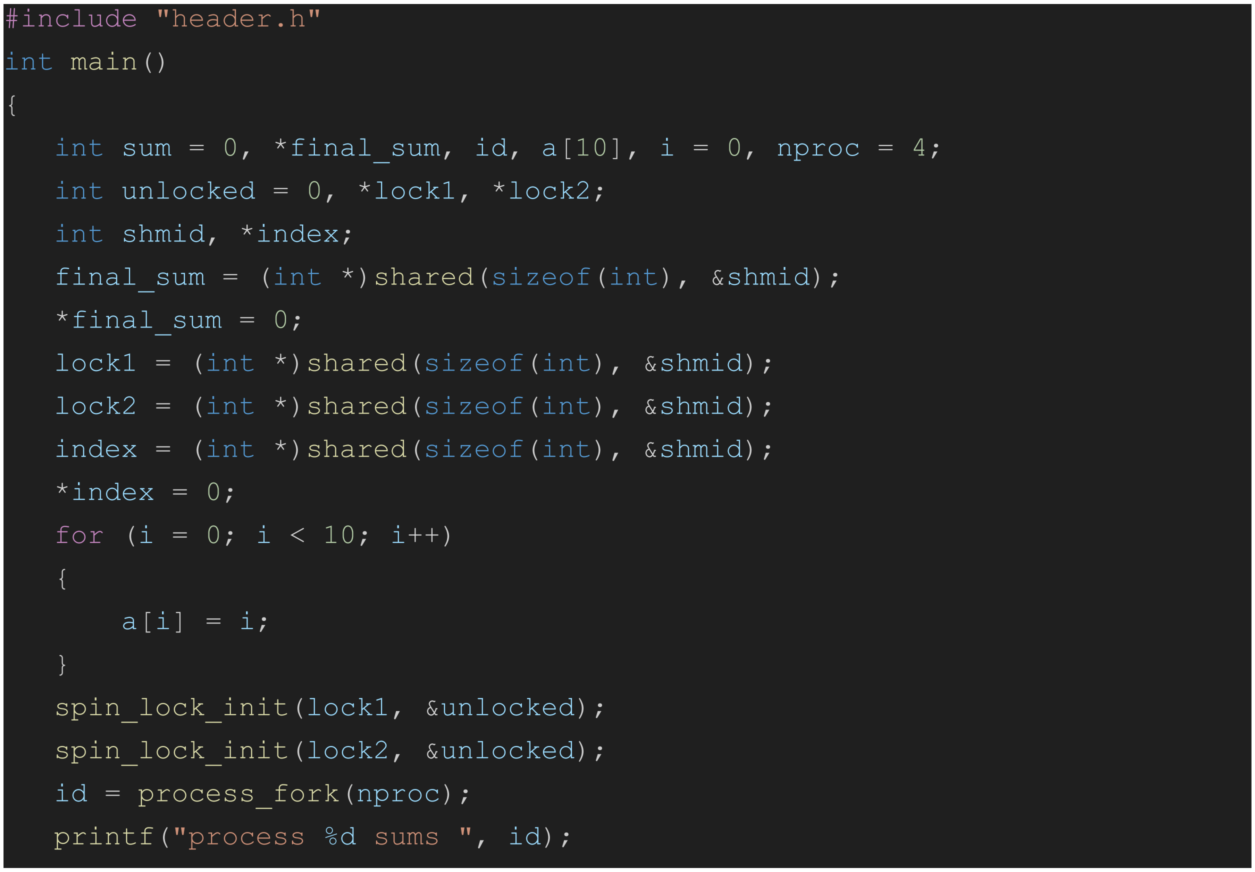
4. Write a parallel program to calculate sum of 1 to n numbers using m processes.

```
#include "header.h" //Declaration and scan values
int main()
{
    int sum = 0, *final_sum, id, N, i, shmid, nproc = 4;
    int locked = 1, unlocked = 0, *lock;
    lock = (int *)shared(sizeof(int), &shmid);
    final_sum = (int *)shared(sizeof(sum), &i);
    *final_sum = 0;
    printf("\n Enter N :");
    scanf("%d", &N);
    spin_lock_init(lock, &unlocked);
    id = process_fork(nproc);
    printf("process %d does ->  ", id);
    for (i = id; i <= N; i += nproc)
    {
        sum = sum + i;
        printf("%d  ", i);
    }
    printf("\n");
    spin_lock(lock);
    *final_sum = sum + *final_sum;
    printf("id = %d sum = %d \n", id, sum);
    spin_unlock(lock);
    process_join(4, id);
    printf("\nSum: %d\n", *final_sum);
    return 0;
}
```

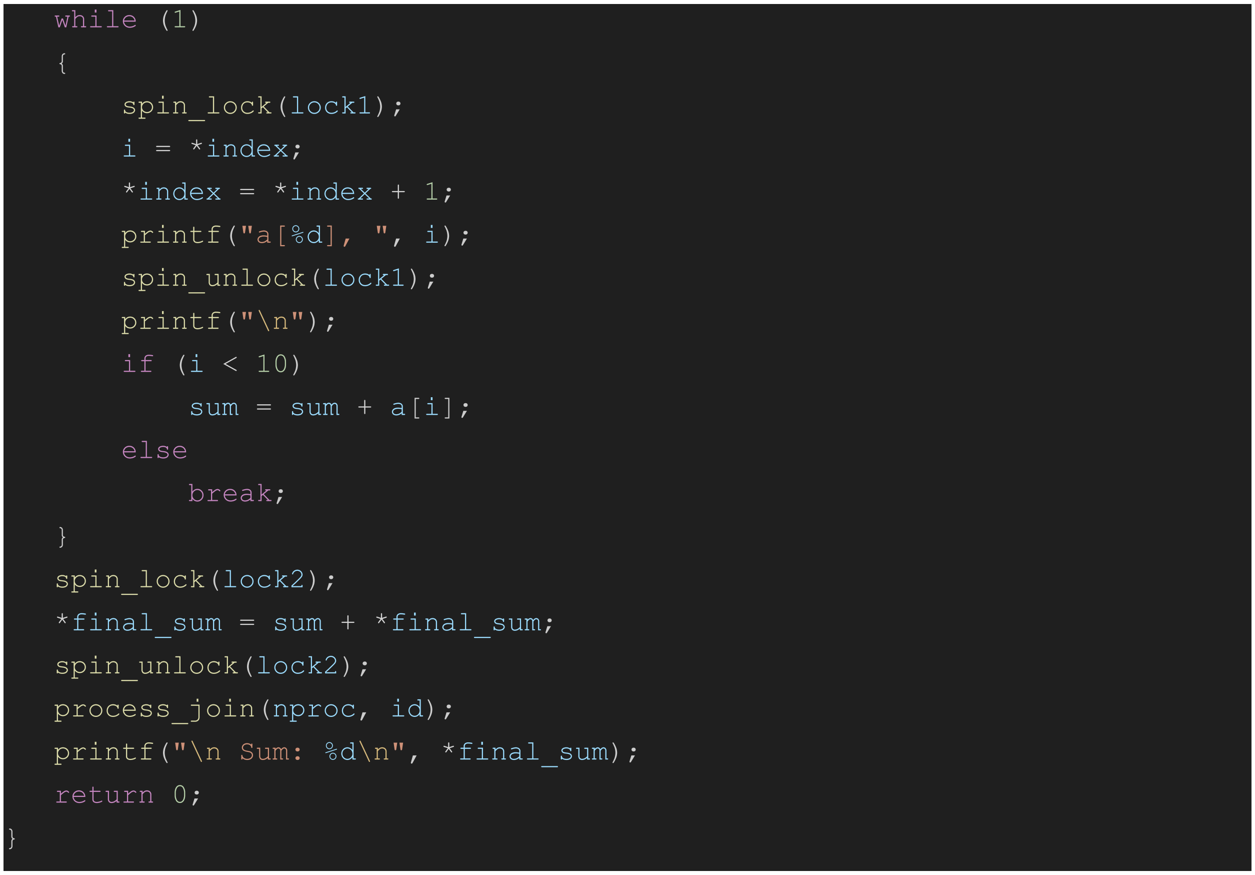


```
┌──(nisarg㉿fedora)-[~/…/Sem_6_repo/ACA/aca_lab11/aca_lab11]
└─$ ./a.out

 Enter N :8
process 1 does ->  1  5
id = 1 sum = 6
process 2 does ->  2  6
id = 2 sum = 8
process 0 does ->  0  4  8
id = 0 sum = 12
process 3 does ->  3  7
id = 3 sum = 10

Sum: 36

┌──(nisarg㉿fedora)-[~/…/Sem_6_repo/ACA/aca_lab11/aca_lab11]
└─$ 
```

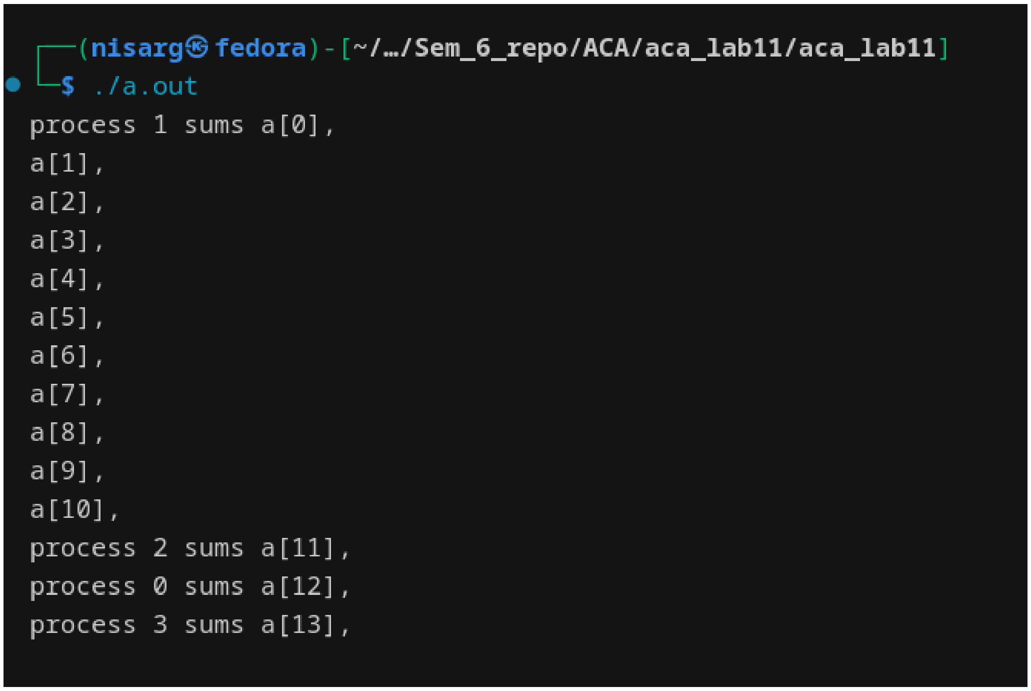
5. Write a parallel program to calculate sum of array of n elements using self-scheduling.



```c
#include "header.h"
int main()
{
   int sum = 0, *final_sum, id, a[10], i = 0, nproc = 4;
   int unlocked = 0, *lock1, *lock2;
   int shmid, *index;
   final_sum = (int *)shared(sizeof(int), &shmid);
   *final_sum = 0;
   lock1 = (int *)shared(sizeof(int), &shmid);
   lock2 = (int *)shared(sizeof(int), &shmid);
   index = (int *)shared(sizeof(int), &shmid);
   *index = 0;
   for (i = 0; i < 10; i++)
   {
       a[i] = i;
   }
   spin_lock_init(lock1, &unlocked);
   spin_lock_init(lock2, &unlocked);
   id = process_fork(nproc);
   printf("process %d sums ", id);
```



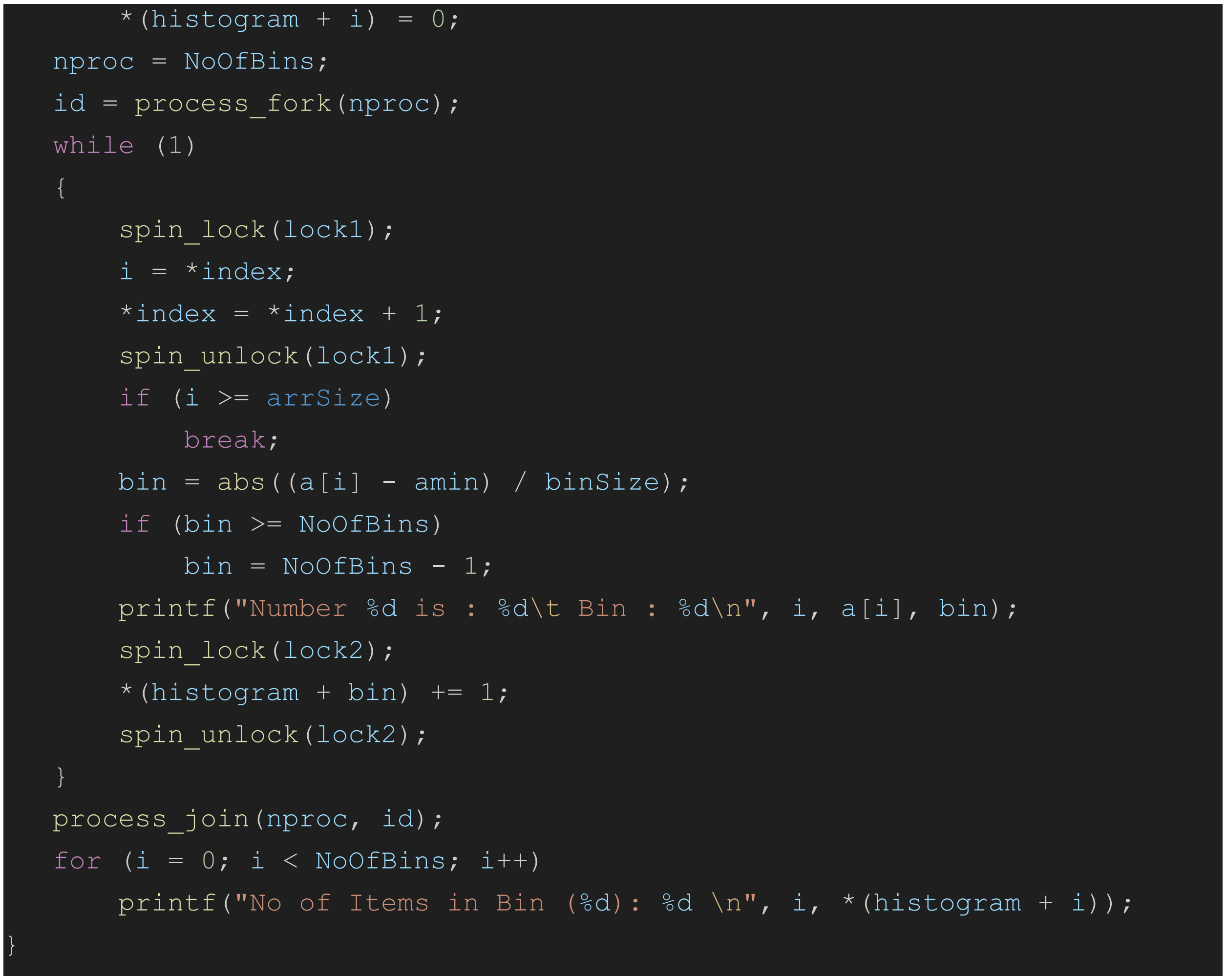
```c
    while (1)
    {
        spin_lock(lock1);
        i = *index;
        *index = *index + 1;
        printf("a[%d], ", i);
        spin_unlock(lock1);
        printf("\n");
        if (i < 10)
            sum = sum + a[i];
        else
            break;
    }
    spin_lock(lock2);
    *final_sum = sum + *final_sum;
    spin_unlock(lock2);
    process_join(nproc, id);
    printf("\n Sum: %d\n", *final_sum);
    return 0;
}
```



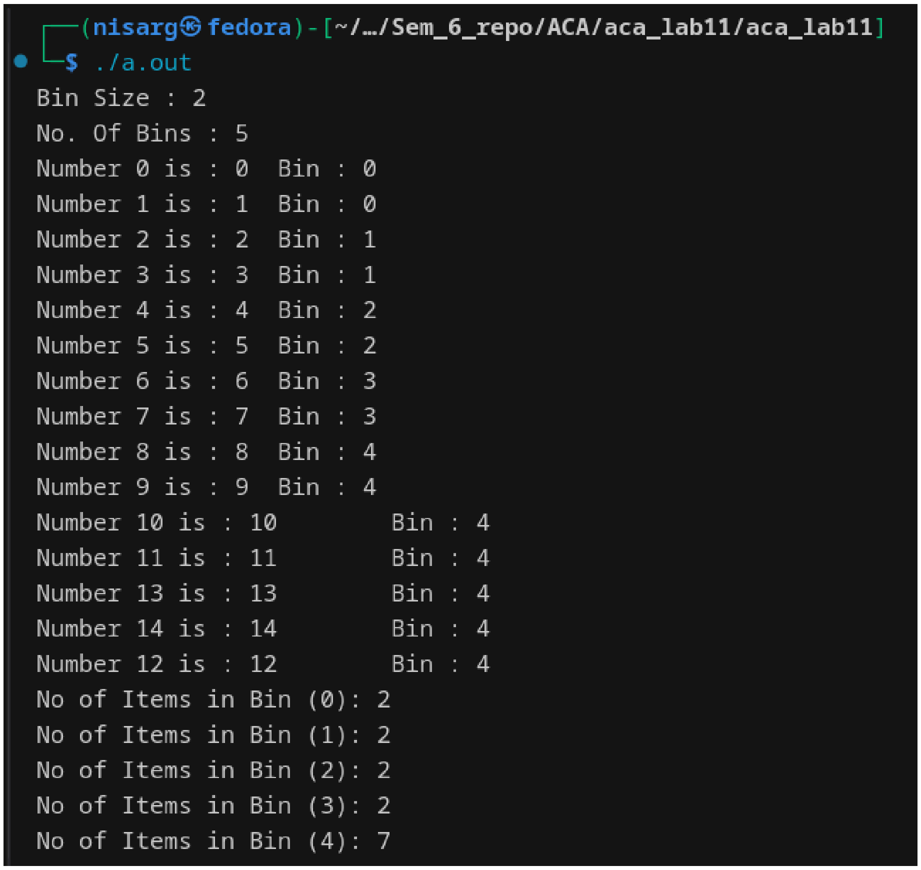
```
┌──(nisarg㉿fedora)-[~/…/Sem_6_repo/ACA/aca_lab11/aca_lab11]
└─$ ./a.out
process 1 sums a[0],
a[1],
a[2],
a[3],
a[4],
a[5],
a[6],
a[7],
a[8],
a[9],
a[10],
process 2 sums a[11],
process 0 sums a[12],
process 3 sums a[13],
```

6. Implementation of histogram in different ways
i. create histogram using self-scheduling

```
#include "header.h"
#include <stdio.h>
#define arrSize 15
int main()
{
   int a[arrSize];
   int i, *index, NoOfBins = 5, binSize, *histogram, *lock1, *lock2,
unlocked = 0, locked = 1;
   int shmidindex, shmidlock1, shmidlock2, shmidhist;
   int id, nproc, bin;
   int amin, amax;
   for (int i = 0; i < arrSize; i++)
   {
       // printf("enter a[%d]",i);
       a[i] = i;
   }
   amin = amax = a[0];
   for (i = 1; i < arrSize; i++)
   {
       if (amin > a[i])
           amin = a[i];
       if (amax < a[i])
           amax = a[i];
   }
   binSize = (amax - amin) / NoOfBins;
   index = (int *)shared(sizeof(int), &shmidindex);
   lock1 = (int *)shared(sizeof(int), &shmidlock1);
   lock2 = (int *)shared(sizeof(int), &shmidlock2);
   histogram = (int *)shared(sizeof(int) * NoOfBins, &shmidhist);
   printf("Bin Size : %d\n", binSize);
   printf("No. Of Bins : %d\n", NoOfBins);
   spin_lock_init(lock1, &unlocked);
   spin_lock_init(lock2, &unlocked);
   *index = 0;
   for (i = 0; i < NoOfBins; i++)
```

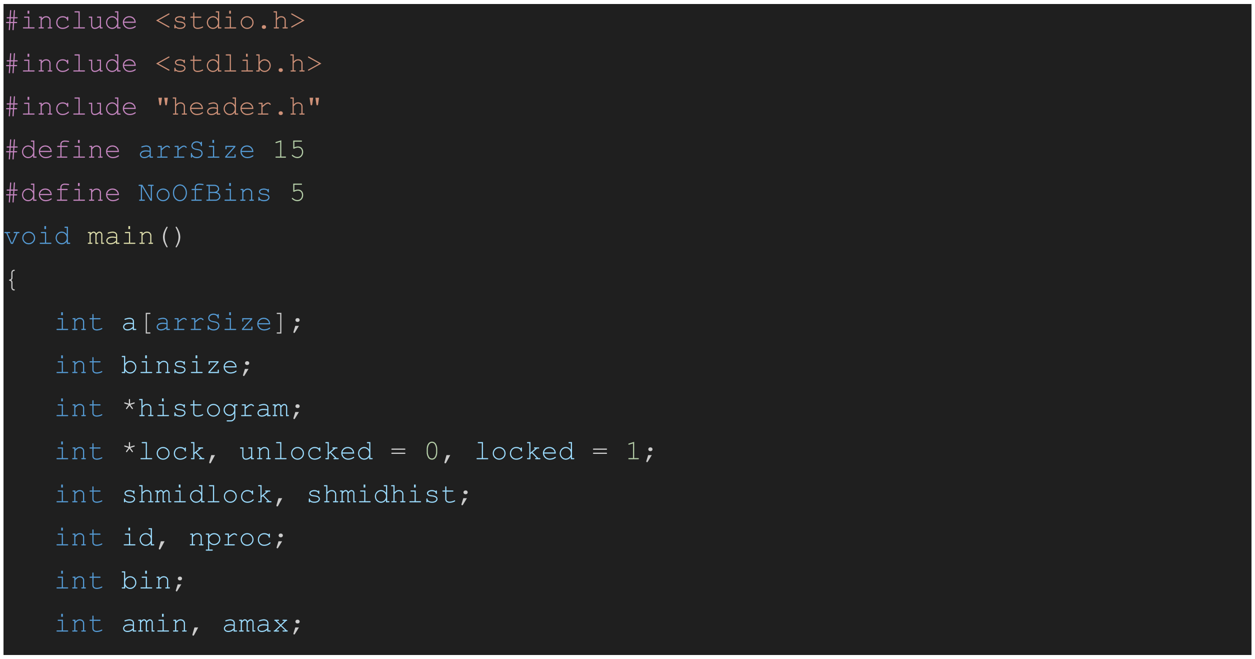


```
        *(histogram + i) = 0;
    nproc = NoOfBins;
    id = process_fork(nproc);
    while (1)
    {
        spin_lock(lock1);
        i = *index;
        *index = *index + 1;
        spin_unlock(lock1);
        if (i >= arrSize)
            break;
        bin = abs((a[i] - amin) / binSize);
        if (bin >= NoOfBins)
            bin = NoOfBins - 1;
        printf("Number %d is : %d\t Bin : %d\n", i, a[i], bin);
        spin_lock(lock2);
        *(histogram + bin) += 1;
        spin_unlock(lock2);
    }
    process_join(nproc, id);
    for (i = 0; i < NoOfBins; i++)
        printf("No of Items in Bin (%d): %d \n", i, *(histogram + i));
}
```

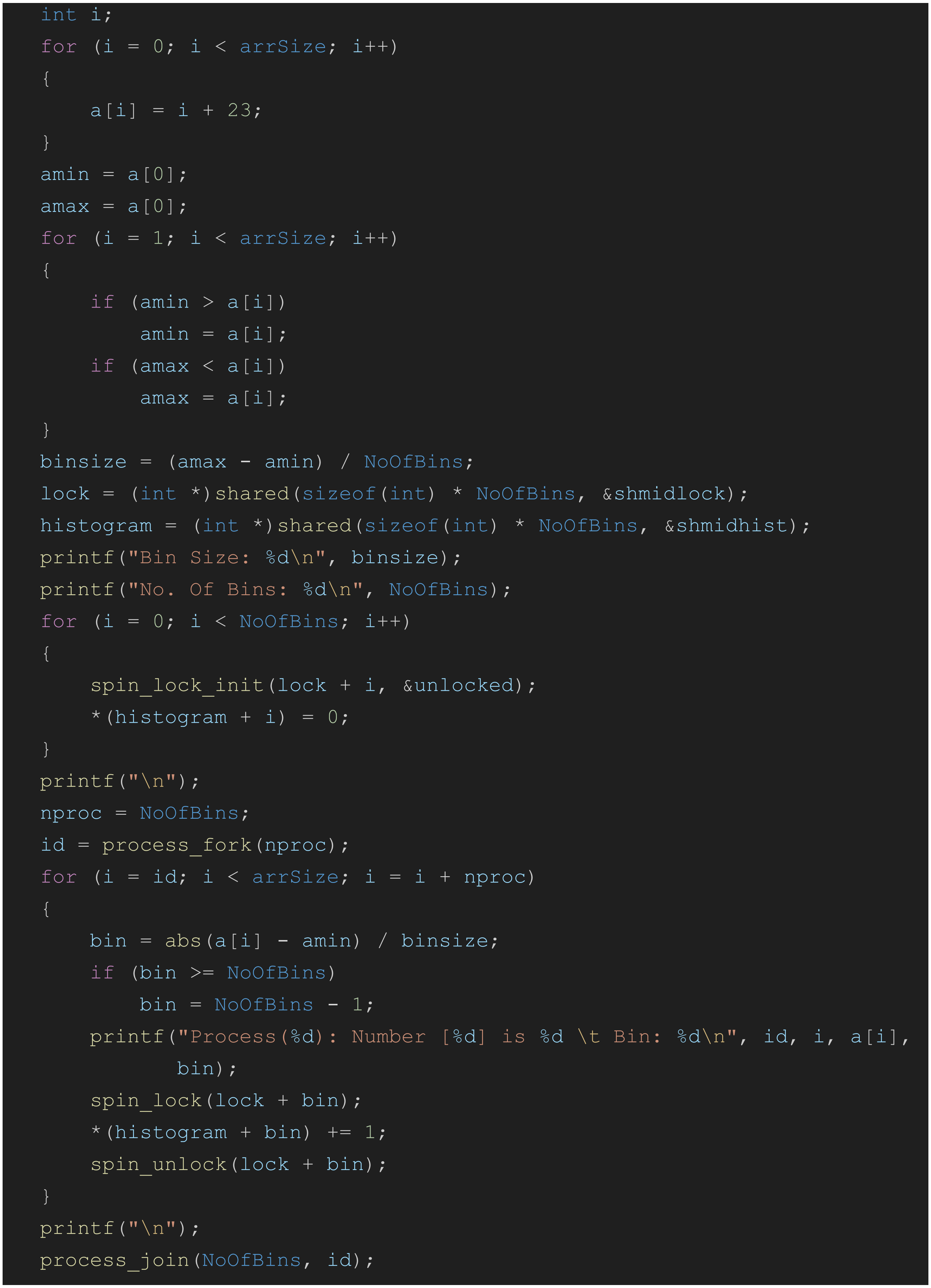


```
┌──(nisarg㉿fedora)-[~/…/Sem_6_repo/ACA/aca_lab11/aca_lab11]
└─$ ./a.out
Bin Size : 2
No. Of Bins : 5
Number 0 is : 0  Bin : 0
Number 1 is : 1  Bin : 0
Number 2 is : 2  Bin : 1
Number 3 is : 3  Bin : 1
Number 4 is : 4  Bin : 2
Number 5 is : 5  Bin : 2
Number 6 is : 6  Bin : 3
Number 7 is : 7  Bin : 3
Number 8 is : 8  Bin : 4
Number 9 is : 9  Bin : 4
Number 10 is : 10        Bin : 4
Number 11 is : 11        Bin : 4
Number 13 is : 13        Bin : 4
Number 14 is : 14        Bin : 4
Number 12 is : 12        Bin : 4
No of Items in Bin (0): 2
No of Items in Bin (1): 2
No of Items in Bin (2): 2
No of Items in Bin (3): 2
No of Items in Bin (4): 7
```

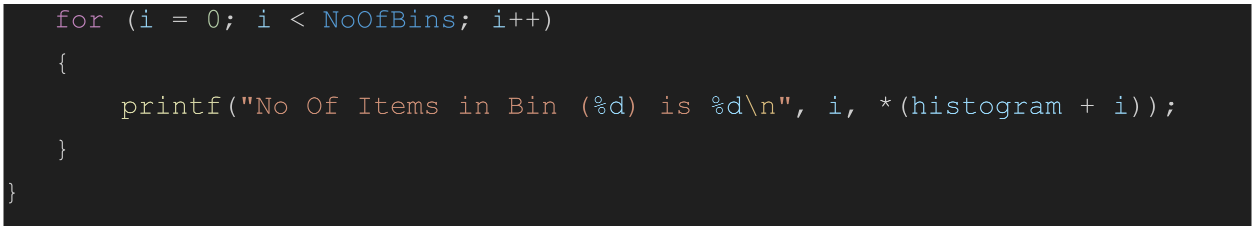
ii. implement histogram using loop splitting



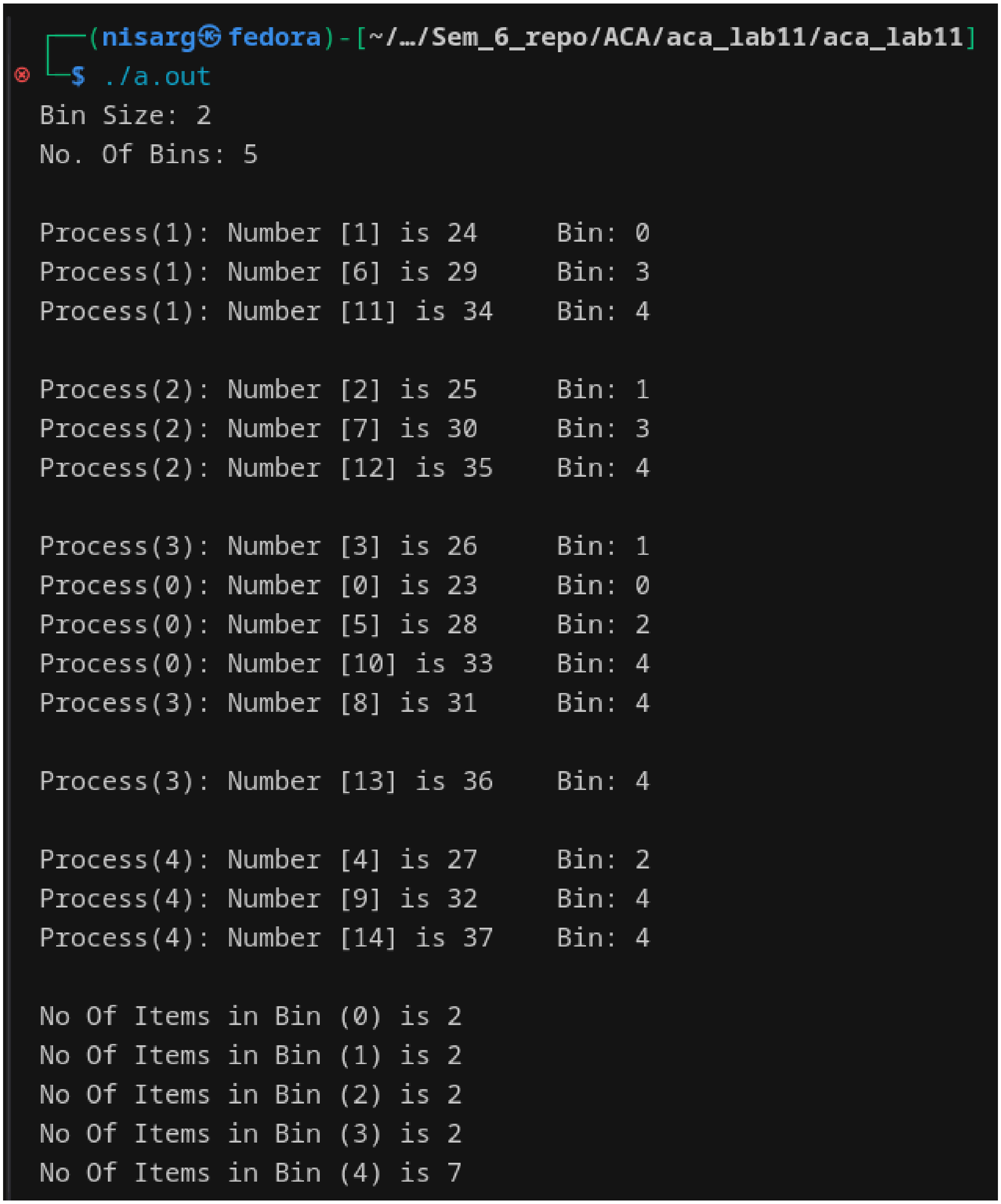
```c
#include <stdio.h>
#include <stdlib.h>
#include "header.h"
#define arrSize 15
#define NoOfBins 5
void main()
{
   int a[arrSize];
   int binsize;
   int *histogram;
   int *lock, unlocked = 0, locked = 1;
   int shmidlock, shmidhist;
   int id, nproc;
   int bin;
   int amin, amax;
```



```
int i;
for (i = 0; i < arrSize; i++)
{
    a[i] = i + 23;
}
amin = a[0];
amax = a[0];
for (i = 1; i < arrSize; i++)
{
    if (amin > a[i])
        amin = a[i];
    if (amax < a[i])
        amax = a[i];
}
binsize = (amax - amin) / NoOfBins;
lock = (int *)shared(sizeof(int) * NoOfBins, &shmidlock);
histogram = (int *)shared(sizeof(int) * NoOfBins, &shmidhist);
printf("Bin Size: %d\n", binsize);
printf("No. Of Bins: %d\n", NoOfBins);
for (i = 0; i < NoOfBins; i++)
{
    spin_lock_init(lock + i, &unlocked);
    *(histogram + i) = 0;
}
printf("\n");
nproc = NoOfBins;
id = process_fork(nproc);
for (i = id; i < arrSize; i = i + nproc)
{
    bin = abs(a[i] - amin) / binsize;
    if (bin >= NoOfBins)
        bin = NoOfBins - 1;
    printf("Process(%d): Number [%d] is %d \t Bin: %d\n", id, i, a[i],
           bin);
    spin_lock(lock + bin);
    *(histogram + bin) += 1;
    spin_unlock(lock + bin);
}
printf("\n");
process_join(NoOfBins, id);
```



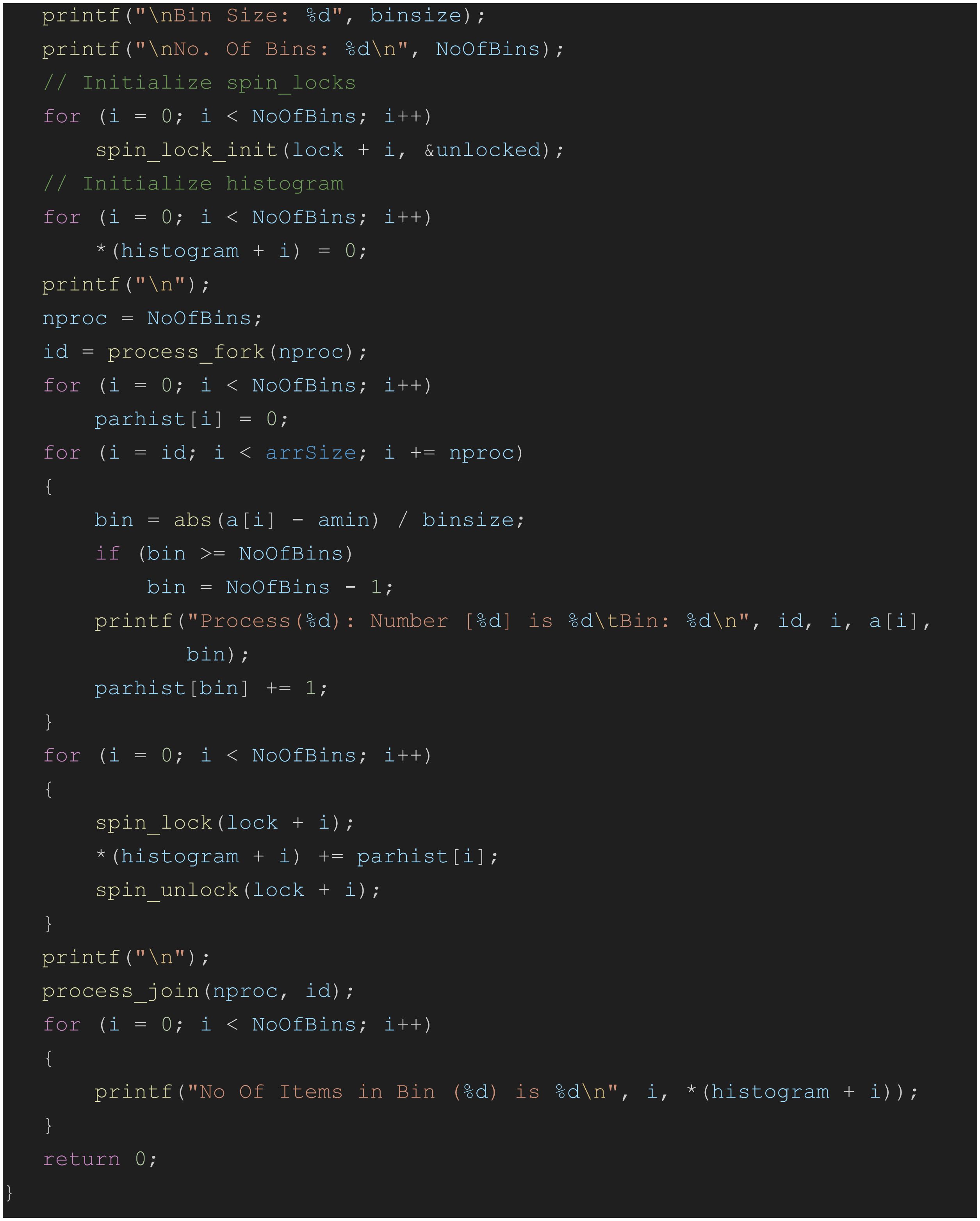
```c
    for (i = 0; i < NoOfBins; i++)
    {
        printf("No Of Items in Bin (%d) is %d\n", i, *(histogram + i));
    }
}
```



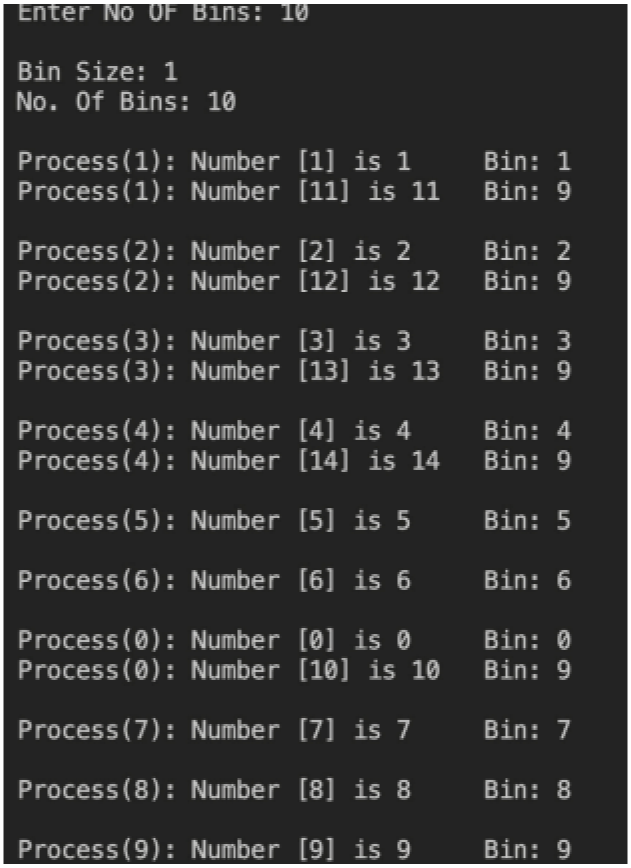
```
┌──(nisarg㉿fedora)-[~/…/Sem_6_repo/ACA/aca_lab11/aca_lab11]
└─$ ./a.out
Bin Size: 2
No. Of Bins: 5

Process(1): Number [1] is 24     Bin: 0
Process(1): Number [6] is 29     Bin: 3
Process(1): Number [11] is 34    Bin: 4

Process(2): Number [2] is 25     Bin: 1
Process(2): Number [7] is 30     Bin: 3
Process(2): Number [12] is 35    Bin: 4

Process(3): Number [3] is 26     Bin: 1
Process(0): Number [0] is 23     Bin: 0
Process(0): Number [5] is 28     Bin: 2
Process(0): Number [10] is 33    Bin: 4
Process(3): Number [8] is 31     Bin: 4

Process(3): Number [13] is 36    Bin: 4

Process(4): Number [4] is 27     Bin: 2
Process(4): Number [9] is 32     Bin: 4
Process(4): Number [14] is 37    Bin: 4

No Of Items in Bin (0) is 2
No Of Items in Bin (1) is 2
No Of Items in Bin (2) is 2
No Of Items in Bin (3) is 2
No Of Items in Bin (4) is 7
```

iii. implement using partial histogram

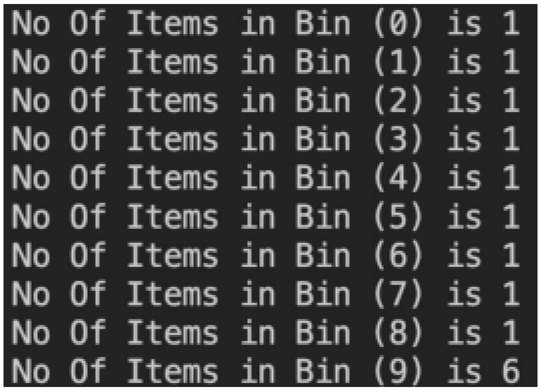
```
#include <stdio.h>
#include <stdlib.h>
#include "header.h"
#define arrSize 15
int main()
{
    int a[arrSize];
    int NoOfBins;
    int binsize;
    int *histogram;
    int parhist[NoOfBins];
    int *lock, unlocked = 0, locked = 1;
    int shmidlock, shmidhist, shmidparhist;
    int id, nproc;
    int bin;
    int amin, amax;
    int i;
    printf("Enter No OF Bins: ");
    scanf("%d", &NoOfBins);
    // Initialize an Array
    for (i = 0; i < arrSize; i++)
    {
        // printf("Enter a[%d]: ", i);
        // scanf("%d", &a[i]);
        a[i] = i;
    }
    amin = a[0];
    amax = a[0];
    for (i = 1; i < arrSize; i++)
    {
        if (amin > a[i])
            amin = a[i];
        if (amax < a[i])
            amax = a[i];
    }
    binsize = (amax - amin) / NoOfBins;
    lock = (int *)shared(sizeof(int) * NoOfBins, &shmidlock);
    histogram = (int *)shared(sizeof(int) * NoOfBins, &shmidhist);
```



```c
    printf("\nBin Size: %d", binsize);
    printf("\nNo. Of Bins: %d\n", NoOfBins);
    // Initialize spin_locks
    for (i = 0; i < NoOfBins; i++)
        spin_lock_init(lock + i, &unlocked);
    // Initialize histogram
    for (i = 0; i < NoOfBins; i++)
        *(histogram + i) = 0;
    printf("\n");
    nproc = NoOfBins;
    id = process_fork(nproc);
    for (i = 0; i < NoOfBins; i++)
        parhist[i] = 0;
    for (i = id; i < arrSize; i += nproc)
    {
        bin = abs(a[i] - amin) / binsize;
        if (bin >= NoOfBins)
            bin = NoOfBins - 1;
        printf("Process(%d): Number [%d] is %d\tBin: %d\n", id, i, a[i],
               bin);
        parhist[bin] += 1;
    }
    for (i = 0; i < NoOfBins; i++)
    {
        spin_lock(lock + i);
        *(histogram + i) += parhist[i];
        spin_unlock(lock + i);
    }
    printf("\n");
    process_join(nproc, id);
    for (i = 0; i < NoOfBins; i++)
    {
        printf("No Of Items in Bin (%d) is %d\n", i, *(histogram + i));
    }
    return 0;
}
```



```
Enter No Of Bins: 10

Bin Size: 1
No. Of Bins: 10

Process(1): Number [1] is 1     Bin: 1
Process(1): Number [11] is 11   Bin: 9

Process(2): Number [2] is 2     Bin: 2
Process(2): Number [12] is 12   Bin: 9

Process(3): Number [3] is 3     Bin: 3
Process(3): Number [13] is 13   Bin: 9

Process(4): Number [4] is 4     Bin: 4
Process(4): Number [14] is 14   Bin: 9

Process(5): Number [5] is 5     Bin: 5

Process(6): Number [6] is 6     Bin: 6

Process(0): Number [0] is 0     Bin: 0
Process(0): Number [10] is 10   Bin: 9

Process(7): Number [7] is 7     Bin: 7

Process(8): Number [8] is 8     Bin: 8

Process(9): Number [9] is 9     Bin: 9
```



```
No Of Items in Bin (0) is 1
No Of Items in Bin (1) is 1
No Of Items in Bin (2) is 1
No Of Items in Bin (3) is 1
No Of Items in Bin (4) is 1
No Of Items in Bin (5) is 1
No Of Items in Bin (6) is 1
No Of Items in Bin (7) is 1
No Of Items in Bin (8) is 1
No Of Items in Bin (9) is 6
```