```
//Ahmad Dar Khalil
//1120443
_____________________________________________________________________________


typedef int buffer_item;
#define BUFFER_SIZE 5

#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <windows.h>
#define sleep(x) Sleep(1000 * x)

#define TRUE 1


pthread_mutex_t mutex;


sem_t full, empty;

buffer_item buffer[BUFFER_SIZE];
```

```
int count;

pthread_t tid;
pthread_attr_t attr;

void *producer(void *param); // producer thread
void *consumer(void *param); // consumer thread

//********************************************************************************



int main(int arg[4]) {

  int i;

system("color 4F");

 double Sleep_time = atoi(arg[1]);
  int P_num = atoi(arg[2]); // Number of producer threads
  int C_num = atoi(arg[3]); // Number of consumer threads

  // Initialize buffer
  initializeData();

  // Create the producer threads
  for(i = 0; i < P_num; i++) {
    pthread_create(&tid,&attr,producer,NULL);
   }
```

```
  // Create the consumer threads
  for(i = 0; i < C_num; i++) {
    pthread_create(&tid,&attr,consumer,NULL);
  }

  // Sleep
  sleep(Sleep_time);

  // Exit the program
  printf("Exit the program\n");
  exit(0);
}


void initializeData() {

  //Create mutex lock
  pthread_mutex_init(&mutex, NULL);

  sem_init(&full, 0, 0);

  sem_init(&empty, 0, BUFFER_SIZE);

  // Get the default attributes
  pthread_attr_init(&attr);

  // init buffer
  count = 0;
}
```

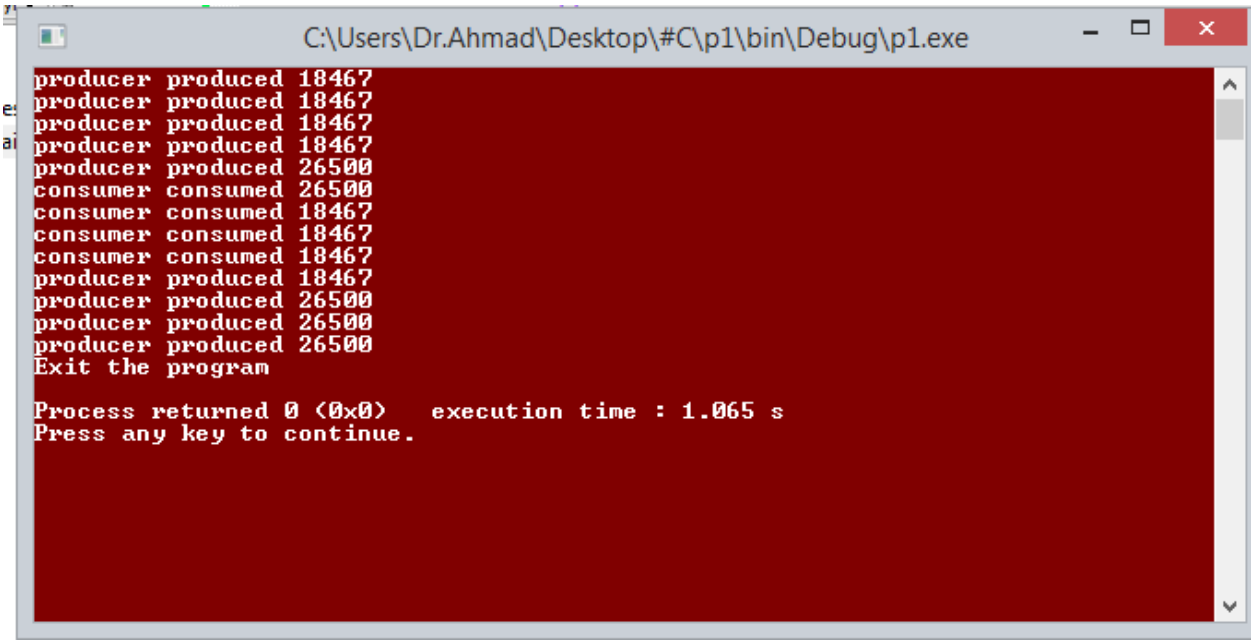
```
// Add an item to the buffer
int insert_item(buffer_item item) {
  //Add if buffer not full
  if(count < BUFFER_SIZE) {
    buffer[count] = item;
    count++;
    return 0;
  }
  else {
    return -1;
  }
}



int remove_item(buffer_item *item) {
  // Remove if buffer is not empty
  if(count > 0) {
    *item = buffer[(count-1)];
    count--;
    return 0;
  }
  else {
    return -1;
  }
}
```

```
// Producer Thread
void *producer(void *param) {
  buffer_item item;

  while(TRUE) {
    // sleep for a random period of time
    int Rand_num = rand() / 10000;
    sleep(Rand_num);

    // generate a random number
    item = rand();

    // acquire the empty lock
    sem_wait(&empty);
    // acquire the mutex lock
    pthread_mutex_lock(&mutex);

    if(insert_item(item)) {
      printf( " report error condition\n");
    }
    else {
      printf("producer produced %d\n", item);
    }
    // release the mutex lock
    pthread_mutex_unlock(&mutex);
    // signal full
    sem_post(&full);
  }
```

```
}
//********************************************************************************

// Consumer Thread
void *consumer(void *param) {
  buffer_item item;

  while(TRUE) {
    // sleep for a random period of time
    int Rand_num = rand() /10000;
    sleep(Rand_num);

    // aquire the full lock
    sem_wait(&full);
    // aquire the mutex lock
    pthread_mutex_lock(&mutex);
    if(remove_item(&item)) {
      printf( "report error condition\n");
    }
    else {
      printf("consumer consumed %d\n", item);
    }

    pthread_mutex_unlock(&mutex);

    sem_post(&empty);
  }
}
```

Sample output with parameters : 1,4,5 :

_____


C:\Users\Dr.Ahmad\Desktop\#C\p1\bin\Debug\p1.exe
producer produced 18467
producer produced 18467
producer produced 18467
producer produced 18467
producer produced 26500
consumer consumed 26500
consumer consumed 18467
consumer consumed 18467
consumer consumed 18467
producer produced 18467
producer produced 26500
producer produced 26500
producer produced 26500
Exit the program
Process returned 0 (0x0) execution time : 1.065 s
Press any key to continue.