Cons1.c

```
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>

#define MAXSIZE 20

typedef struct {
	int in, out, list[MAXSIZE];
	sem_t full, emp;
	pthread_mutex_t lock;
} Itemlist;

Itemlist A;
int item = 1, size;

void *producer(void *arg) {
	int id = *(int *)arg;
	free(arg);
	printf("Producer thread %d created\n", id + 1);
	while (item <= size) {
		sem_wait(&A.emp);
		pthread_mutex_lock(&A.lock);
		printf("Producer %d produced item %d\n", id + 1, item);
		A.list[A.in++ % MAXSIZE] = item++;
		pthread_mutex_unlock(&A.lock);
		sem_post(&A.full);
		sleep(1);
	}
	return NULL;
}

void *consumer(void *arg) {
	int id = *(int *)arg;
	free(arg);
	printf("Consumer thread %d created\n", id + 1);
	while (1) {
		sem_wait(&A.full);
		pthread_mutex_lock(&A.lock);
		printf("Consumer %d consumed item %d\n", id + 1, A.list[A.out++ % MAXSIZE]);
		pthread_mutex_unlock(&A.lock);
		sem_post(&A.emp);
	}
	return NULL;
}

int main() {
	int np, nc;
```

```
	A.in = A.out = 0;
	sem_init(&A.full, 0, 0);
	sem_init(&A.emp, 0, MAXSIZE);
	pthread_mutex_init(&A.lock, NULL);

	printf("Enter number of producers, consumers, and items to produce: ");
	scanf("%d %d %d", &np, &nc, &size);

	pthread_t threads[np + nc];
	for (int i = 0; i < np + nc; i++) {
		int *id = malloc(sizeof(int));
		*id = i < np ? i : i - np;
		pthread_create(&threads[i], NULL, i < np ? producer : consumer, id);
	}

	for (int i = 0; i < np + nc; i++) pthread_join(threads[i], NULL);
	return 0;
}
```