#include <iostream>

#define HAVE\_STRUCT\_TIMESPEC

#include <pthread.h>

#include <stdlib.h>

#include <stdio.h>

#include <cstdlib>

#define NOMINMAX

#ifdef \_WIN32

#include <conio.h>

#include <Windows.h>

#else

#include <unistd.h>

#endif

int count = 0;

size\_t n;

size\_t H;

static int over = 0;

pthread\_t\* bees;

int\* pr;

pthread\_mutex\_t mutex;

pthread\_cond\_t not\_full;

pthread\_cond\_t not\_empty;

int input() {

int n;

std::cin >> n;

bool test = true;

do {

if (!(test = std::cin.good())) {

std::cout << "Incorrect input. Try again." << std::endl;

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

}

} while (!test);

return n;

}

void\* Bear(void\* param) {

while (over <= 5) {

pthread\_mutex\_lock(&mutex);

while (count != H)

pthread\_cond\_wait(&not\_full, &mutex);

count = 0;

pthread\_mutex\_unlock(&mutex);

pthread\_cond\_broadcast(&not\_full);

printf("Bear ate\n");

Sleep(1000);

over++;

}

return NULL;

}

void\* Producer(void\* param) {

int pNum = \*((int\*)param);

int i;

while (1) {

pthread\_mutex\_lock(&mutex);

while (count == H) {

pthread\_cond\_wait(&not\_empty, &mutex);

}

count++;

pthread\_mutex\_unlock(&mutex);

pthread\_cond\_signal(&not\_empty);

printf("Bee %d fills the pot\n", pNum);

Sleep(1000);

}

return NULL;

}

int main() {

int i;

pthread\_mutex\_init(&mutex, NULL);

pthread\_cond\_init(&not\_full, NULL);

pthread\_cond\_init(&not\_empty, NULL);

do {

std::cout << "Input a number of bees <= 30:" << std::endl;

n = input();

} while (n <= 1 || n > 30);

do {

std::cout << "Input a number of sips less than the number of bees: <= 25:" << std::endl;

H = input();

} while (H <= 0 || H > 25 || H >= n);

bees = new pthread\_t[n];

pr = new int[n];

for (i = 0; i < n; i++) {

pr[i] = i + 1;

pthread\_create(&bees[i], NULL, Producer, (void\*)(pr + i));

}

pthread\_t c\_thread;

pthread\_create(&c\_thread, NULL, Bear, NULL);

int mNum = 0;

Bear((void\*)&mNum);

std::cout << "Bear ate";

delete[] bees;

delete[] pr;

return 0;

}