/*
    This central process will receive the value of its temperature via the first
    command line argument, and will wait for messages from the four outer
    processes, indicating what their temps are. Using a weighted formula, this
    central process will recompute its temp, and send that to all four outer
    processes, and it is up to the central process to notice when everyone
    has stabilized. This condition implies that the central process has now
    received the same temp from each of the outer processes, and therefore,
    there is no reason to continue to compute temps since they will never change
    again.

    This process will remove the queue it is receiving on, and it checks each
    and every msg system call for the status it returns; <0 when an error has
    occurred.
*/

#include <sys/errno.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <stdio.h>
#include "message.h"

int main(int argc, char *argv[])
{
    struct {
        long priority;
        int temp;
        int pid;
        int stable;
    } msgp;

    struct msqid_ds buffer; /* needed for IPC_RMID operation */

    int msg_ct, k, stable, temp, status;
    int all[5], old[5]; /* hold the previous and current temps from the outer
                         processs for stability comparison purposes. */
    int msqid;
    long key, qid1, qid2, qid3, qid4;

    if (argc < 2){
        printf("need a count in 1st arg\n");

        return 1;
    }

    temp = atoi(argv[1]);
    msgp.priority = 2;
    msgp.stable = 1;
    key = 20;
    all[0] = old[0] = -1;

    // create the mailbox for the central process
    msqid = msgget(key,0666 | IPC_CREAT);

    if ((qid1 = msgget(key + 1,0666 | IPC_CREAT)) < 0){
        perror("Central could not open an outer message queue.\n");
        return 1;
    }

    printf("qid1 = %d\n", qid1);
if ((qid2 = msgget(key + 2, 0666 | IPC_CREAT)) < 0){
    perror("Central could not open an outer message queue.\n");
    return 1;
}
if ((qid3 = msgget(key + 3, 0666 | IPC_CREAT)) < 0){
    perror("Central could not open an outer message queue.\n");
    return 1;
}
if ((qid4 = msgget(key + 4, 0666 | IPC_CREAT)) < 0){
    perror("Central could not open an outer message queue.\n");
    return 1;
}
do {
    for (msg_ct = 0; msg_ct < 4; msg_ct++){
        if ((status = msgrcv(msqid,&msgp,sizeof(msgp)-sizeof(long),2,0)) < 0){
            perror("Error when trying to receive message from outer process.\n");
            return 1;
        }
        all[msgp.pid] = msgp.temp;
    } /* receive all four messages before moving to temp update */
    stable = 0;
        stable = 1;
    for (k = 1; k <= 4; k++)
        old[k] = all[k];
    msgp.stable = stable;
    msgp.temp = temp;
    msgp.pid = 130; /* place a bogus value here to catch possible logic error in the multiplier process (when it sends temp back)*/
    msgp.priority = 2; /* ensure all messages are priority 2 */
    if ((status = msgsnd(qid1,&msgp,sizeof(msgp)-sizeof(long),0)) < 0){
        perror("Error sending to outer process.\n");
        return 1;
    }
    if ((status = msgsnd(qid2,&msgp,sizeof(msgp)-sizeof(long),0)) < 0){
        perror("Error sending to outer process.\n");
        return 1;
    }
    if ((status = msgsnd(qid3,&msgp,sizeof(msgp)-sizeof(long),0)) < 0){
        perror("Error sending to outer process.\n");
        return 1;
    }
    if ((status = msgsnd(qid4,&msgp,sizeof(msgp)-sizeof(long),0)) < 0){
        perror("Error sending to outer process.\n");
        return 1;
    }
} while (stable != 1);
printf("central process says the temp is %d\n",temp);
if (msgctl(msqid, IPC_RMID, &buffer) < 0){
    perror("Central process could not remove its message queue.\n");
}
return 1;
}