```c
// 5.1.a

#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>
#include <sys/wait.h>

#define N 1000

int main(){
    pid_t pid;

    for(int i=0; i<N; i++){
        pid=fork();
        if(pid==0){exit(0);}
        else if(pid==-1){
            printf("Failed to create process %d\n",i);
            perror("fork"); exit(errno);
        }
    }

    while(wait(NULL)!=-1);

    return 0;
}
```

```c
// 5.1.b

#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

#define N 1000

void *thread_func(void *arg){pthread_exit(NULL);}

int main(){
    int ret;
    pthread_t tids[N];

    for(int i=0; i<N; i++){
        ret = pthread_create(&tids[i], NULL, thread_func, NULL);
        if(ret!=0){
            printf("Failed to create thread %d (error %d)\n", i, ret);
            exit(ret);
        }
    }

    pthread_exit(NULL);
}

```

```c
// 5.2.a

#include <stdio.h>
#include <pthread.h>

void *thread_func(void *arg){
    long num = (long)arg;
    printf("%ld\n", num);
    pthread_exit(NULL);
}

int main(){
    long i;
    pthread_t tids[3];
    for(i=0; i<3; i++)
        pthread_create(&tids[i], NULL, thread_func, (void*)i);
    pthread_exit(NULL);
}

```

```c
// 5.2.b

#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

void *thread_func(void *arg){
    int *nums = (int *)arg;
    printf("%d\t%d\n", nums[0], nums[1]);
    free(arg);
    pthread_exit(NULL);
}

int main(){
    int i, *nums;
    pthread_t tids[3];
    for(i=0; i<3; i++) {
        nums = (int *)malloc(2*sizeof(int));
        nums[0] = i; nums[1] = i+1;
        pthread_create(&tids[i], NULL, thread_func, nums);
    }
    pthread_exit(NULL);
}
```

```c
// 5.2.c

#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

typedef struct{float f; char c;} thread_param;

void *thread_func(void *arg){
    thread_param *param = (thread_param *)arg;
    printf("%f\t%c\n", param->f, param->c);
    free(arg);
    pthread_exit(NULL);
}

int main(){
    pthread_t tids[3];

    for(int i=0; i<3; i++){
        thread_param *arg = (thread_param *)malloc(sizeof(thread_param));
        arg->f=(float)i;
        arg->c='a'+(int)i;
        pthread_create(&tids[i], NULL, thread_func, arg);
    }

    pthread_exit(NULL);
}

```

```c
// 5.3.a

#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

int sum(int min, int max){
    int i, result = 0;
    for(i=min; i<=max; i++) result += i;
    return result;
}

int partial_sum;

void *thread_func(void *arg){
    partial_sum = sum(51, 100);
    pthread_exit(NULL);
}

int main(){
    pthread_t tid;
    int total_sum;
    pthread_create(&tid, NULL, thread_func, NULL);
    total_sum = sum(1, 50);
    pthread_join(tid, NULL);
    total_sum += partial_sum;
    printf("total sum: %d\n", total_sum);
    pthread_exit(NULL);
}

```

```c
// 5.3.b

#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

int sum(int min, int max){
    int i, result = 0;
    for(i=min; i<=max; i++) result += i;
    return result;
}

void *thread_func(void *arg){
    long partial_sum = sum(51, 100);
    pthread_exit((void *)partial_sum);
}

int main(){
    pthread_t tid;
    void *retval;
    long total_sum;
    pthread_create(&tid, NULL, thread_func, NULL);
    total_sum = sum(1, 50);
    pthread_join(tid, &retval);
    total_sum += (long)retval;
    printf("total sum: %ld\n", total_sum);
    pthread_exit(NULL);
}

```

```c
// 5.4

#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

int sum(int min, int max){
    int i, result = 0;
    for(i=min; i<=max; i++) result += i;
    return result;
}

int partial_sum[10];

void *thread_func(void *arg){
    long i = (long)arg;
    partial_sum[i] = sum(i*10+1, i*10+10);
    pthread_exit(NULL);
}

int main(){
    long i;
    pthread_t tids[10];
    int total_sum = 0;

    for(i=0; i<10; i++)
        pthread_create(&tids[i], NULL, thread_func, (void *)i);

    for(i=0; i<10; i++){
        pthread_join(tids[i], NULL);
        total_sum += partial_sum[i];
    }

    printf("totalsum: %d\n", total_sum);
    return 0;
}
```

```c
// 5.5

#include <stdio.h>
#include <pthread.h>

int square_sum = 0; // square_sum will be accessed sequentially, one thread at a time

void *thread_func(void *arg){
    long i = (long)arg;
    square_sum += i * i;
    if(i<10) { // create next thread if n < 10
        pthread_t tid;
        pthread_create(&tid, NULL, thread_func, (void*)(i+1));
        pthread_join(tid, NULL);
    }
    pthread_exit(NULL);
}

int main(){
    long i = 1;
    pthread_t tid;
    pthread_create(&tid, NULL, thread_func, (void*)i); // start the chain with thread 1
    pthread_join(tid, NULL);
    printf("square sum: %d\n", square_sum);
    pthread_exit(NULL);
}

```

```c
// 5.6

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

#define N 10
#define N_THREADS 2
#define TARGET 3

int *A;

void *thread_func(void *arg) {
    long i = (long)arg, count = 0;
    for(int j=i*(N/N_THREADS); j<(i+1)*(N/N_THREADS); j++)
        if(A[j]==TARGET) count++;
    pthread_exit((void *)count);
}

int main() {
    pthread_t tids[N_THREADS];
    void *retval;
    long total = 0;

    A = (int *)malloc(N*sizeof(int));
    srandom(getpid());
    for (int i = 0; i < N; i++) {
        A[i] = random() % 5;
        printf("%d\n", A[i]);
    }

    for (long i = 0; i < N_THREADS; i++)
        pthread_create(&tids[i], NULL, thread_func, (void *)i);

    for (int i = 0; i < N_THREADS; i++) {
        pthread_join(tids[i], &retval);
        total += (long)retval;
    }

    printf("count=%ld\n", total);

    free(A);
    return 0;
}

```

```c
// 5.7.a

#include <stdio.h>
#include <pthread.h>

unsigned long int counter=0;

void *thread_func(void *arg){
    for(unsigned long int i=0; i<1000000; i++){
        counter++;
    }
}

int main(){
    pthread_t tid;

    pthread_create(&tid, NULL, thread_func, NULL);

    thread_func(NULL);

    pthread_join(tid, NULL);

    printf("counter: %lu\n", counter);
    pthread_exit(NULL);
}

```

```c
// 5.7.b

#include <stdio.h>
#include <pthread.h>

unsigned long int counter=0;
pthread_mutex_t counter_mutex=PTHREAD_MUTEX_INITIALIZER;

void *thread_func(void *arg){
    for(unsigned long int i=0; i<1000000; i++){
        pthread_mutex_lock(&counter_mutex);
        counter++;
        pthread_mutex_unlock(&counter_mutex);
    }
}

int main(){
    pthread_t tid;

    pthread_create(&tid, NULL, thread_func, NULL);

    thread_func(NULL);

    pthread_join(tid, NULL);

    printf("counter: %lu\n", counter);
    pthread_mutex_destroy(&counter_mutex);
    pthread_exit(NULL);
}

```

```c
// 5.7.c

#include <stdio.h>
#include <pthread.h>

unsigned long int counter=0;
pthread_mutex_t mutex_main=PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t mutex_sec=PTHREAD_MUTEX_INITIALIZER;

void *thread_func(void *arg){
    for(unsigned long int i=0; i<1000000; i++){
        pthread_mutex_lock(&mutex_sec);
        counter++;
        printf("sec: %ld\n", counter);
        pthread_mutex_unlock(&mutex_main);
    }
}

int main(){
    pthread_t tid;

    pthread_mutex_lock(&mutex_sec);

    pthread_create(&tid, NULL, thread_func, NULL);

    for(unsigned long int i=0; i<1000000; i++){
        pthread_mutex_lock(&mutex_main);
        counter++;
        printf("main: %ld\n", counter);
        pthread_mutex_unlock(&mutex_sec);
    }

    pthread_join(tid, NULL);

    printf("counter: %lu\n", counter);
    pthread_mutex_destroy(&mutex_main);
    pthread_mutex_destroy(&mutex_sec);
    pthread_exit(NULL);
}

```

```c
// 5.8

#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

int sum(int min, int max){
    int i, result = 0;
    for(i=min; i<=max; i++) result += i;
    return result;
}

int total_sum = 0;
pthread_mutex_t sum_mutex = PTHREAD_MUTEX_INITIALIZER;

void *thread_func(void *arg){
    long i = (long)arg;
    int partial_sum = sum(i*10+1, i*10+10);
    pthread_mutex_lock(&sum_mutex);
    total_sum += partial_sum;
    pthread_mutex_unlock(&sum_mutex);
    pthread_exit(NULL);
}

int main(){
    long i;
    pthread_t tids[10];

    for(i=0; i<10; i++)
        pthread_create(&tids[i], NULL, thread_func, (void *)i);

    for(i=0; i<10; i++){
        pthread_join(tids[i], NULL);
    }

    printf("total sum: %d\n", total_sum);
    pthread_mutex_destroy(&sum_mutex);
    return 0;
}

```

```c
// 5.9

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

#define N 10
#define N_THREADS 2
#define TARGET 3

int *A;
long total_count = 0;
pthread_mutex_t total_mutex = PTHREAD_MUTEX_INITIALIZER;

void *thread_func(void *arg){
    long i = (long)arg, count = 0;
    for(int j=i*(N/N_THREADS); j<(i+1)*(N/N_THREADS); j++)
        if(A[j]==TARGET) count++;
    pthread_mutex_lock(&total_mutex);
    total_count += count;
    pthread_mutex_unlock(&total_mutex);
    pthread_exit(NULL);
}

int main(){
    pthread_t tids[N_THREADS];

    A = (int *)malloc(N*sizeof(int));
    srandom(getpid());
    for(int i = 0; i < N; i++){
        A[i] = random() % 5;
        printf("%d\n", A[i]);
    }

    for(long i = 0; i < N_THREADS; i++)
        pthread_create(&tids[i], NULL, thread_func, (void *)i);

    for(int i = 0; i < N_THREADS; i++)
        pthread_join(tids[i], NULL);

    printf("count=%ld\n", total_count);

    free(A);
    pthread_mutex_destroy(&total_mutex);
    return 0;
}
```