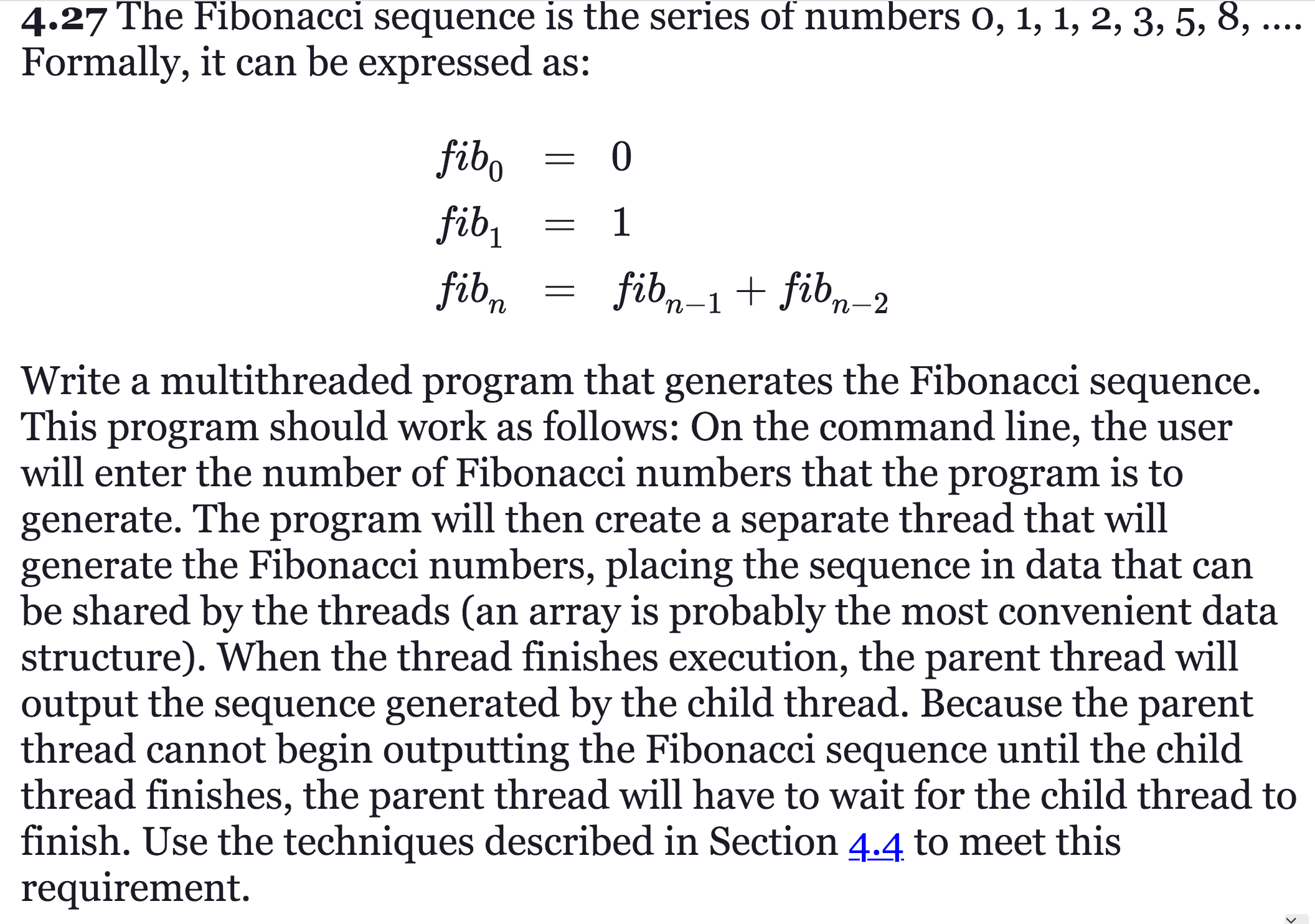
CS-149 Homework-5



#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

// Structure to pass arguments to the thread

typedef struct {

int \*sequence;

int length;

} ThreadArgs;

// Function to generate Fibonacci sequence

void \*generateFibonacci(void \*arg) {

ThreadArgs \*args = (ThreadArgs \*)arg;

int length = args->length;

int \*sequence = args->sequence;

if (length > 0) {

sequence[0] = 0;

}

if (length > 1) {

sequence[1] = 1;

}

for (int i = 2; i < length; i++) {

sequence[i] = sequence[i - 1] + sequence[i - 2];

}

pthread\_exit(NULL);

}

int main(int argc, char \*argv[]) {

if (argc != 2) {

fprintf(stderr, "Usage: %s <number>\n", argv[0]);

return 1;

}

int length = atoi(argv[1]);

if (length <= 0) {

fprintf(stderr, "Please enter a positive integer.\n");

return 1;

}

int \*sequence = (int \*)malloc(length \* sizeof(int));

if (!sequence) {

fprintf(stderr, "Memory allocation failed.\n");

return 1;

}

pthread\_t thread;

ThreadArgs args = { sequence, length };

if (pthread\_create(&thread, NULL, generateFibonacci, &args)) {

fprintf(stderr, "Thread creation failed.\n");

return 1;

}

if (pthread\_join(thread, NULL)) {

fprintf(stderr, "Thread join failed.\n");

return 1;

}

printf("Generated Fibonacci Sequence:\n");

for (int i = 0; i < length; i++) {

printf("%d ", sequence[i]);

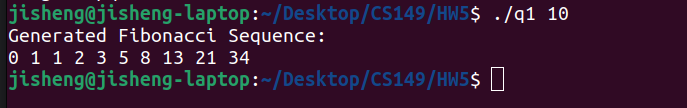
}

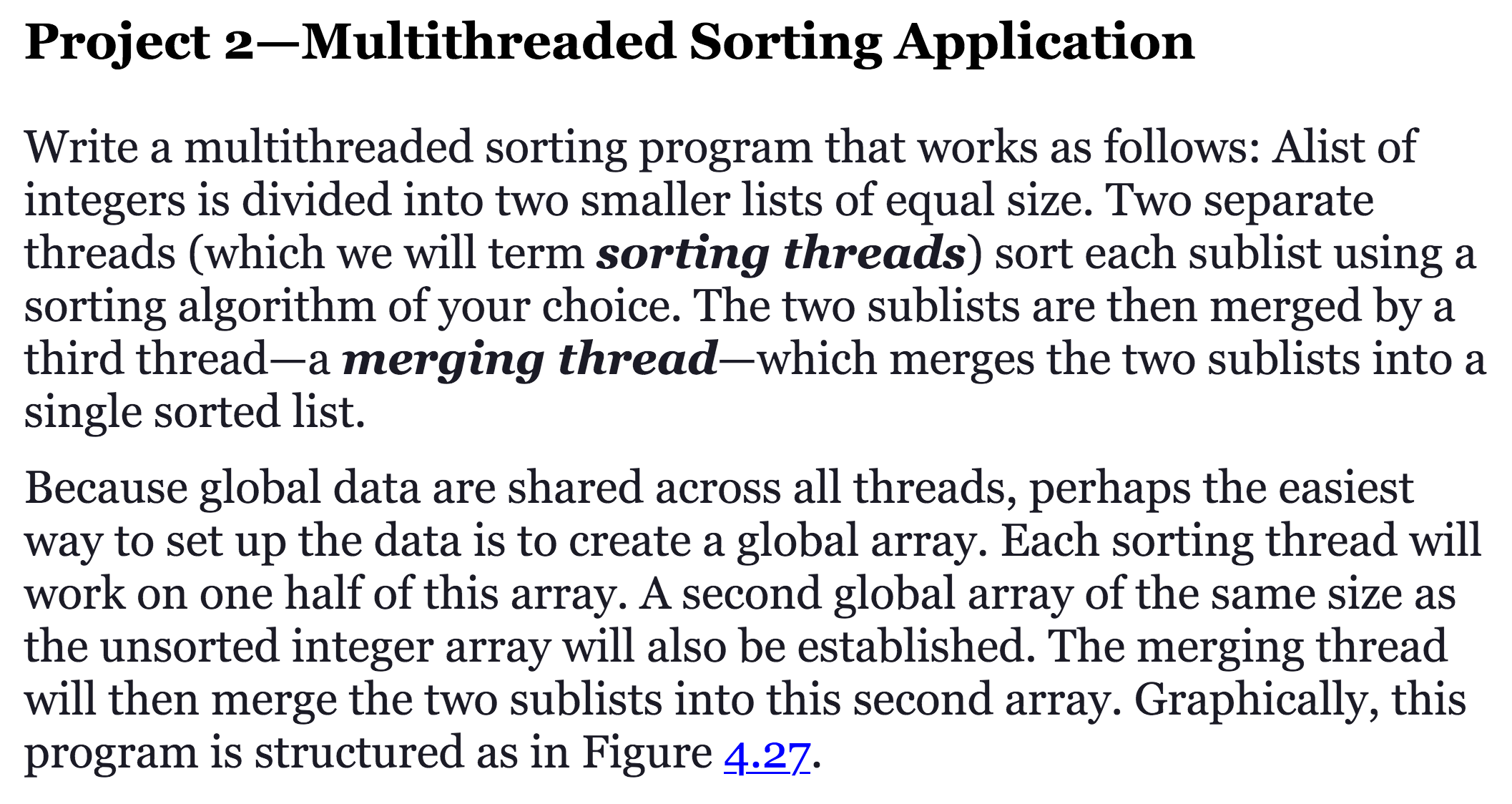
printf("\n");

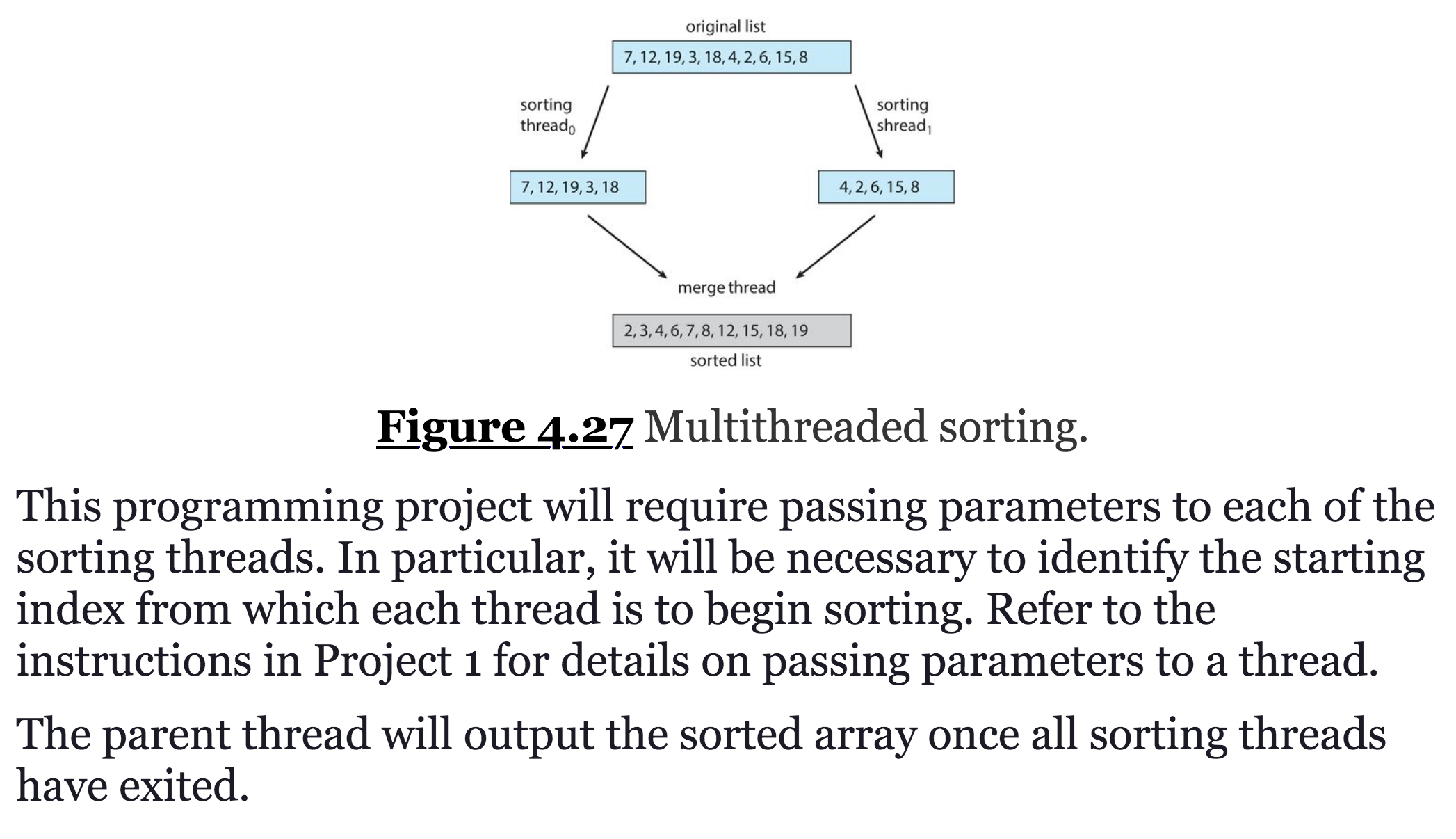
free(sequence);

return 0;

}







Change the above program to have 4 sort threads and 3 merge threads.

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#define SIZE 10

int list[SIZE] = {9, 7, 5, 3, 1, 2, 4, 6, 8, 10};

int sortedList[SIZE];

void \*sort(void \*arg) {

int start = \*((int \*)arg);

int end = start + SIZE / 2;

for (int i = start; i < end; i++) {

for (int j = start; j < end - 1; j++) {

if (list[j] > list[j + 1]) {

int temp = list[j];

list[j] = list[j + 1];

list[j + 1] = temp;

}

}

}

pthread\_exit(NULL);

}

void \*merge(void \*arg) {

int start = \*((int \*)arg);

int end = start + SIZE / 4;

int mid = start + SIZE / 2;

int i = start, j = mid, k = start;

while (i < mid && j < end) {

if (list[i] < list[j]) {

sortedList[k++] = list[i++];

} else {

sortedList[k++] = list[j++];

}

}

while (i < mid) {

sortedList[k++] = list[i++];

}

while (j < end) {

sortedList[k++] = list[j++];

}

pthread\_exit(NULL);

}

int main() {

pthread\_t sortThreads[4];

pthread\_t mergeThreads[3];

int threadArgs[4] = {0, SIZE / 4, SIZE / 2, 3 \* SIZE / 4};

for (int i = 0; i < 4; i++) {

pthread\_create(&sortThreads[i], NULL, sort, &threadArgs[i]);

}

for (int i = 0; i < 4; i++) {

pthread\_join(sortThreads[i], NULL);

}

for (int i = 0; i < 3; i++) {

pthread\_create(&mergeThreads[i], NULL, merge, &threadArgs[i]);

}

for (int i = 0; i < 3; i++) {

pthread\_join(mergeThreads[i], NULL);

}

printf("Sorted List:\n");

for (int i = 0; i < SIZE; i++) {

printf("%d ", sortedList[i]);

}

printf("\n");

return 0;

}

