

Operating

systems

LAB - 08

* Faculty name: Sir Usama
* Student ID: k224779
* Student name: Muhammad Bilal

**Q1:**

**(a)**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#define ARRAY\_SIZE 10000000

int main() {

float \*A = (float \*)malloc(ARRAY\_SIZE \* sizeof(float));

float \*B = (float \*)malloc(ARRAY\_SIZE \* sizeof(float));

float \*C = (float \*)malloc(ARRAY\_SIZE \* sizeof(float));

for (int i = 0; i < ARRAY\_SIZE; i++) {

A[i] = i; // Example initialization, modify as needed

B[i] = i \* 2; // Example initialization, modify as needed

}

for (int i = 0; i < ARRAY\_SIZE; i++) {

C[i] = A[i] + B[i];

}

C

printf("First few elements of C: ");

for (int i = 0; i < 10; i++) {

printf("%.2f ", C[i]);

}

printf("\n");

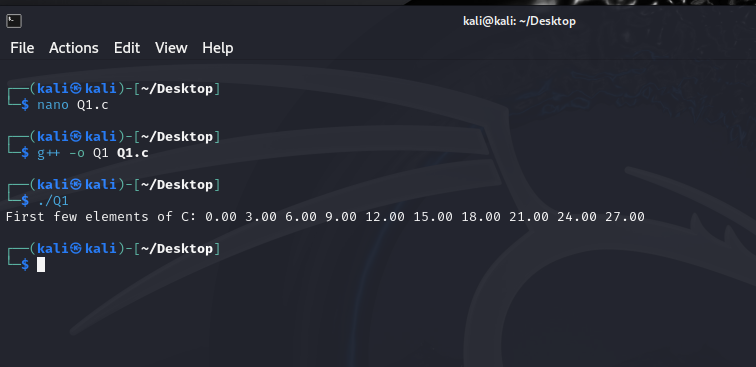
free(A);

free(B);

free(C);

return 0;

}



**(b)**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#define ARRAY\_SIZE 10000000

#define NUM\_THREADS 10

float A[ARRAY\_SIZE], B[ARRAY\_SIZE], C[ARRAY\_SIZE];

typedef struct {

int thread\_id;

int start\_index;

int end\_index;

} ThreadData;

void \*element\_wise\_addition(void \*arg) {

ThreadData \*data = (ThreadData \*)arg;

int start = data->start\_index;

int end = data->end\_index;

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

C[i] = A[i] + B[i];

}

pthread\_exit(NULL);

}

int main() {

pthread\_t threads[NUM\_THREADS];

ThreadData thread\_data[NUM\_THREADS];

for (int i = 0; i < ARRAY\_SIZE; i++) {

A[i] = i; // Example initialization, modify as needed

B[i] = i \* 2; // Example initialization, modify as needed

}

// Create threads

int chunk\_size = ARRAY\_SIZE / NUM\_THREADS;

for (int i = 0; i < NUM\_THREADS; i++) {

thread\_data[i].thread\_id = i;

thread\_data[i].start\_index = i \* chunk\_size;

thread\_data[i].end\_index = (i + 1) \* chunk\_size;

pthread\_create(&threads[i], NULL, element\_wise\_addition, (void \*)&thread\_data[i]);

}

for (int i = 0; i < NUM\_THREADS; i++) {

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

}

printf("First few elements of C: ");

for (int i = 0; i < 10; i++) {

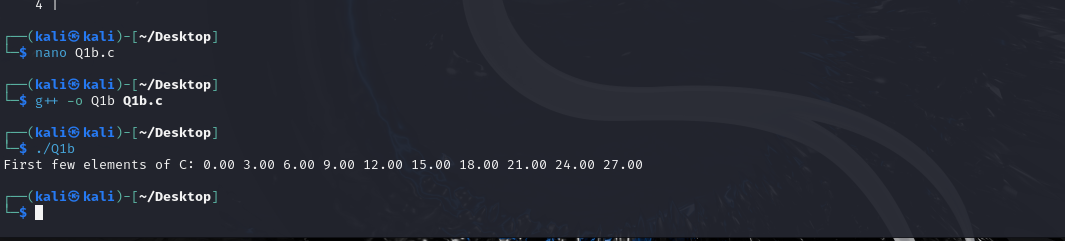
printf("%.2f ", C[i]);

}

printf("\n");

return 0;

}



**Q2:**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <cstdint>

#define MAX\_NUMBERS 100

int numbers[MAX\_NUMBERS];

int num\_count;

typedef struct {

int \*data;

int size;

} ThreadData;

void \*find\_average(void \*arg) {

ThreadData \*data = (ThreadData \*)arg;

int \*numbers = data->data;

int size = data->size;

int sum = 0;

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

sum += numbers[i];

}

float average = (float)sum / size;

pthread\_exit((void \*)(intptr\_t)average);

}

void \*find\_max(void \*arg) {

ThreadData \*data = (ThreadData \*)arg;

int \*numbers = data->data;

int size = data->size;

int max = numbers[0];

for (int i = 1; i < size; i++) {

if (numbers[i] > max) {

max = numbers[i];

}

}

pthread\_exit((void \*)(intptr\_t)max);

}

void \*find\_min(void \*arg) {

ThreadData \*data = (ThreadData \*)arg;

int \*numbers = data->data;

int size = data->size;

int min = numbers[0];

for (int i = 1; i < size; i++) {

if (numbers[i] < min) {

min = numbers[i];

}

}

pthread\_exit((void \*)(intptr\_t)min);

}

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

if (argc < 2) {

printf("Usage: %s <num1> <num2> ... <numN>\n", argv[0]);

return 1;

}

num\_count = argc - 1;

for (int i = 0; i < num\_count; i++) {

numbers[i] = atoi(argv[i + 1]);

}

pthread\_t threads[3];

ThreadData thread\_data[3];

void \*thread\_return;

thread\_data[0].data = numbers;

thread\_data[0].size = num\_count;

pthread\_create(&threads[0], NULL, find\_average, (void \*)&thread\_data[0]);

thread\_data[1].data = numbers;

thread\_data[1].size = num\_count;

pthread\_create(&threads[1], NULL, find\_max, (void \*)&thread\_data[1]);

thread\_data[2].data = numbers;

thread\_data[2].size = num\_count;

pthread\_create(&threads[2], NULL, find\_min, (void \*)&thread\_data[2]);

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

pthread\_join(threads[i], &thread\_return);

if (i == 0)

printf("The average value is %.2f.\n", (float)(intptr\_t)thread\_return);

else if (i == 1)

printf("The maximum value is %d.\n", (int)(intptr\_t)thread\_return);

else

printf("The minimum value is %d.\n", (int)(intptr\_t)thread\_return);

}

return 0;

}

