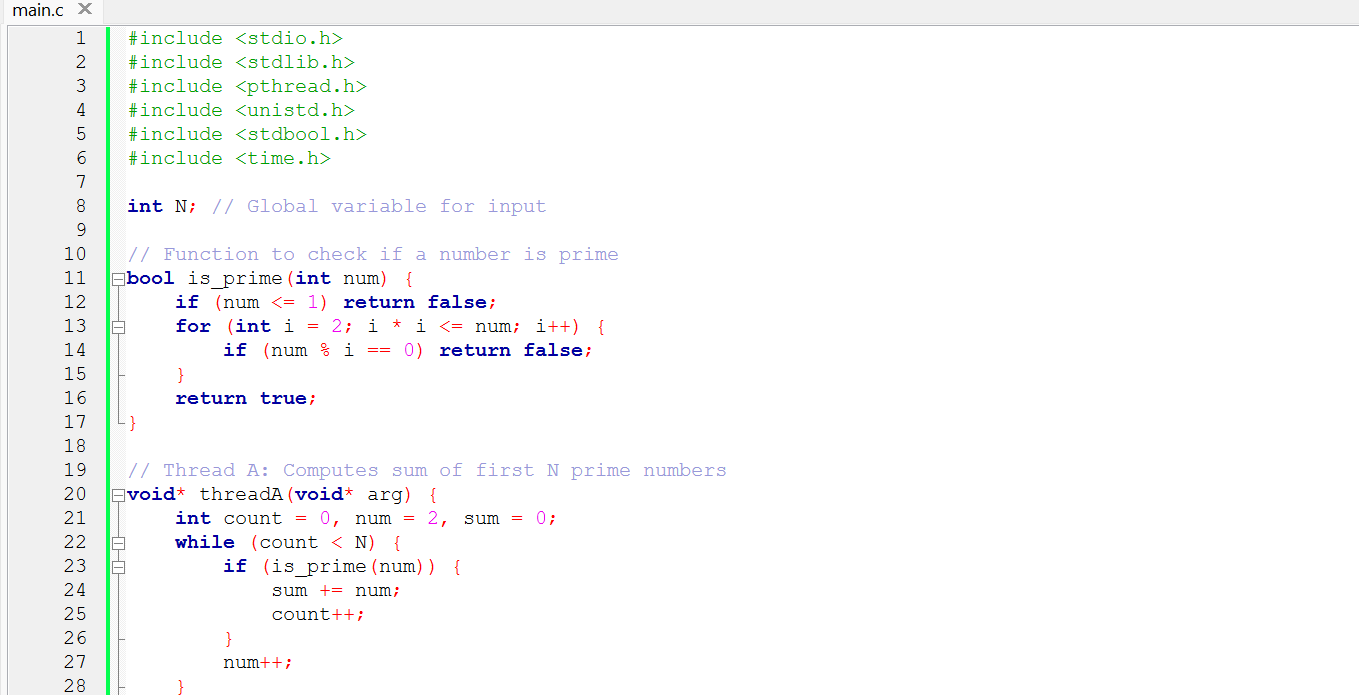
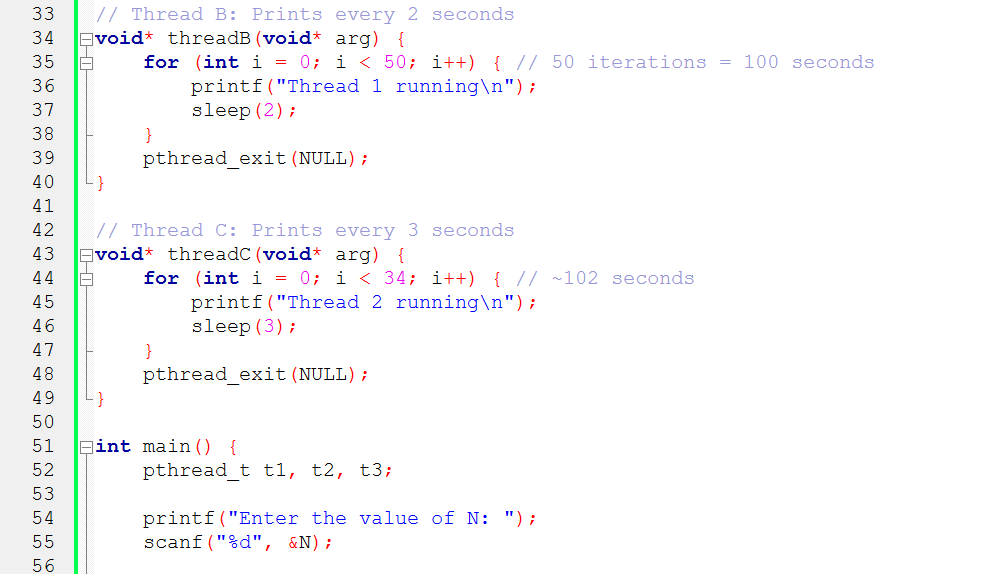
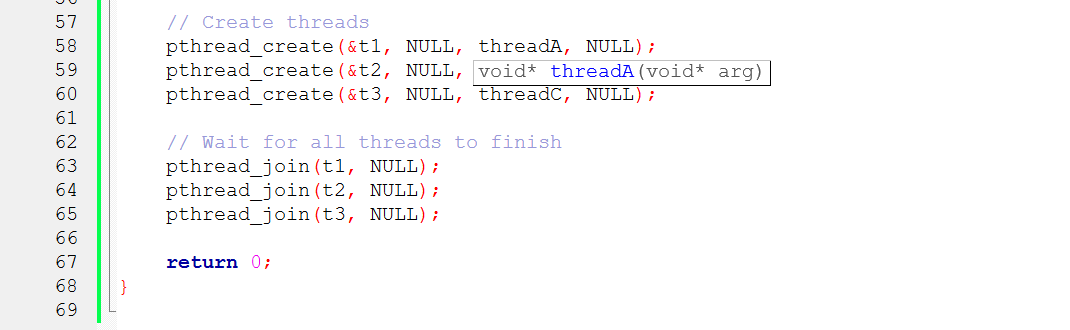


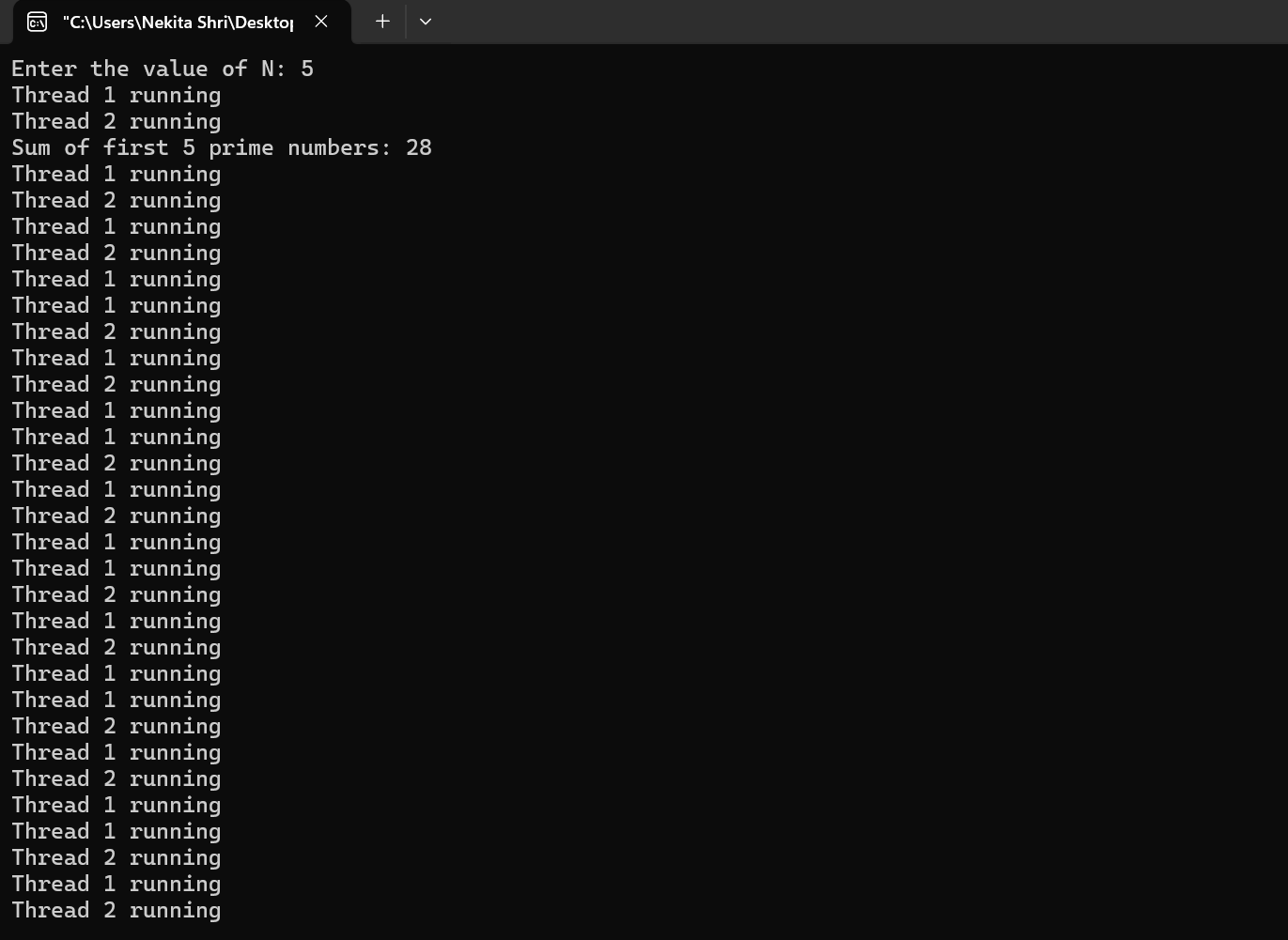
**CODE:**

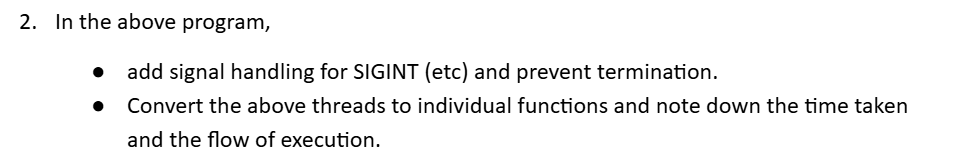






**OUTPUT:**





#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

#include <stdbool.h>

#include <signal.h>

#include <time.h>

int N; // Input number of primes

// Signal handler to prevent Ctrl+C termination

void sigint\_handler(int sig) {

printf("\nSIGINT received, but termination prevented. Program continues...\n");

}

// Timer helper

double time\_diff(struct timespec start, struct timespec end) {

return (end.tv\_sec - start.tv\_sec) + (end.tv\_nsec - start.tv\_nsec) / 1e9;

}

// Check if number is prime

bool is\_prime(int num) {

if (num <= 1) return false;

for (int i = 2; i \* i <= num; i++) {

if (num % i == 0) return false;

}

return true;

}

// Thread A function

void\* threadA(void\* arg) {

struct timespec start, end;

clock\_gettime(CLOCK\_REALTIME, &start);

printf("Thread A started\n");

int count = 0, num = 2, sum = 0;

while (count < N) {

if (is\_prime(num)) {

sum += num;

count++;

}

num++;

}

clock\_gettime(CLOCK\_REALTIME, &end);

printf("Thread A finished: Sum of first %d primes is %d\n", N, sum);

printf("Thread A execution time: %.6f seconds\n", time\_diff(start, end));

pthread\_exit(NULL);

}

// Thread B function

void\* threadB(void\* arg) {

struct timespec start, end;

clock\_gettime(CLOCK\_REALTIME, &start);

printf("Thread B started\n");

for (int i = 0; i < 50; i++) {

printf("Thread 1 running\n");

sleep(2);

}

clock\_gettime(CLOCK\_REALTIME, &end);

printf("Thread B finished. Execution time: %.6f seconds\n", time\_diff(start, end));

pthread\_exit(NULL);

}

// Thread C function

void\* threadC(void\* arg) {

struct timespec start, end;

clock\_gettime(CLOCK\_REALTIME, &start);

printf("Thread C started\n");

for (int i = 0; i < 34; i++) {

printf("Thread 2 running\n");

sleep(3);

}

clock\_gettime(CLOCK\_REALTIME, &end);

printf("Thread C finished. Execution time: %.6f seconds\n", time\_diff(start, end));

pthread\_exit(NULL);

}

int main() {

// Set signal handler

signal(SIGINT, sigint\_handler);

pthread\_t t1, t2, t3;

printf("Enter the value of N: ");

scanf("%d", &N);

// Start threads

pthread\_create(&t1, NULL, threadA, NULL);

pthread\_create(&t2, NULL, threadB, NULL);

pthread\_create(&t3, NULL, threadC, NULL);

// Join threads

pthread\_join(t1, NULL);

pthread\_join(t2, NULL);

pthread\_join(t3, NULL);

printf("Main thread finished.\n");

return 0;

}