National Textile University, Faisalabad



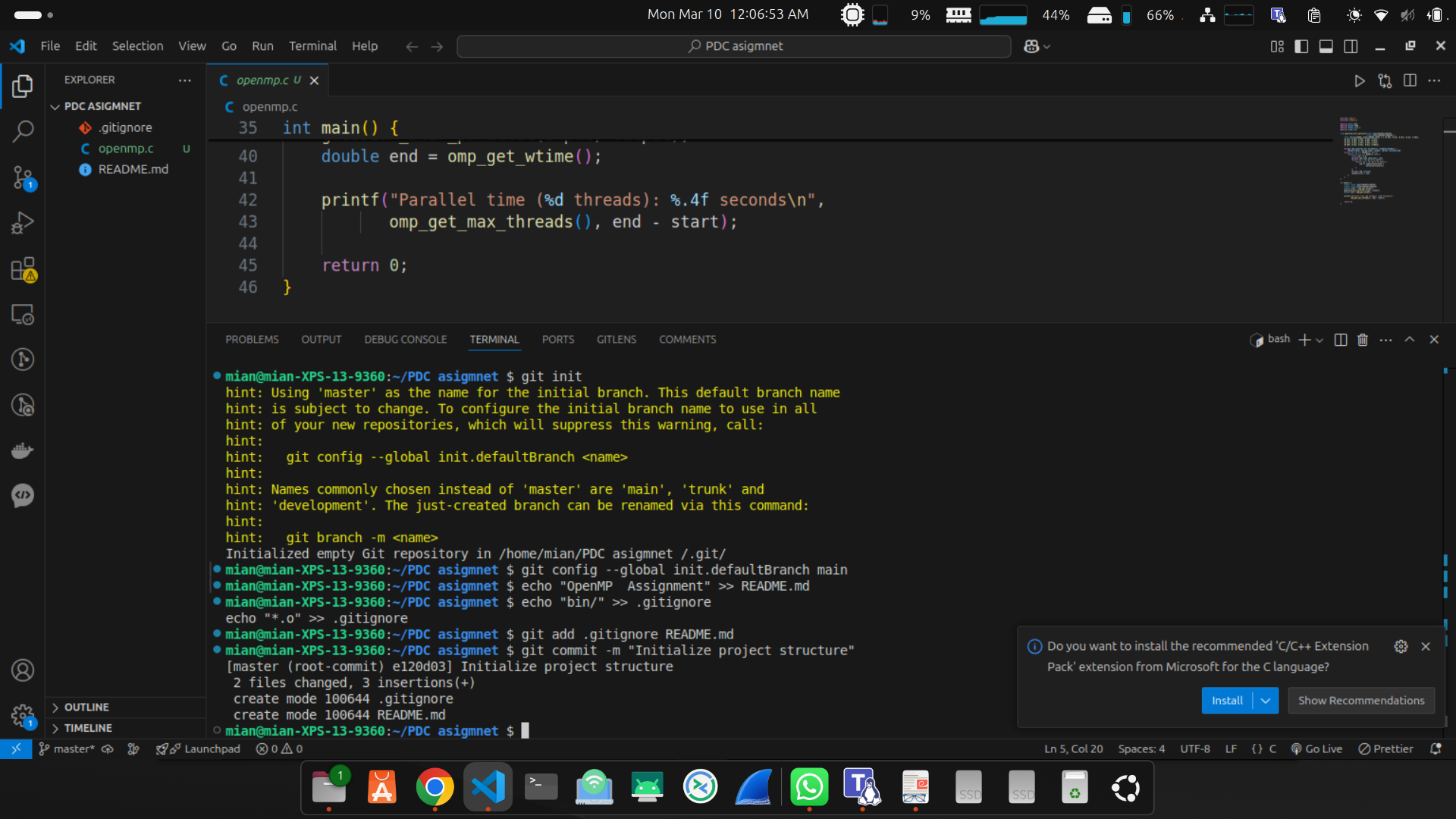
**Department of Computer Science**

|  |  |
| --- | --- |
| **Name:** | Abdullah |
| **Class:** | BSCS-A 6th |
| **Registration No:** | 22-NTU-CS-1137 |
| **Activity:** | Assignment |
| **Course Name:** | Parallel and Distributed Computing |
| **Submitted To:** | *Sir. Nasir Mahmood* |
| **Submission Date:** | 9 March, 2025 |

# Project s Git Initialization:

## Commit 1: Initialize project structure

Created a Git repository and added files like .gitignore to ignore unnecessary files and READMD.md with a brief project overview.



*Figure 1*

### Implemented the Code of Sequential Gaussian Blur

**Commit 2: Adding Sequential Gaussian Blur Implementation**Here I implemented the Gaussian blur filter using a sequential approach without OpenMP. The program was executed 10 times with results recorded and averaged. The implementation processes a **2048x2048 image matrix** using a **5x5 Gaussian kernel** with standard deviation σ=1.0.

**Code Structure:**

* gaussian\_blur(): Applies the Gaussian kernel to the input image matrix
* measure\_execution\_time(): Initializes random pixel values (0-255) and measures execution time

# Code

#include <omp.h>

#include <stdio.h>

#define WIDTH 2048

#define HEIGHT 2048

#define KERNEL\_SIZE 5

#define SIGMA 1.0

void gaussian\_blur\_parallel(float input[HEIGHT][WIDTH],

float output[HEIGHT][WIDTH]) {

float kernel[KERNEL\_SIZE][KERNEL\_SIZE] = { {0.003, 0.013, 0.022, 0.013, 0.003},

{0.013, 0.059, 0.097, 0.059, 0.013},

{0.022, 0.097, 0.159, 0.097, 0.022},

{0.013, 0.059, 0.097, 0.059, 0.013},

{0.003, 0.013, 0.022, 0.013, 0.003}};

#pragma omp parallel for collapse(2) schedule(dynamic) \

default(none) shared(input, output, kernel)

for(int i = 2; i < HEIGHT-2; i++) {

for(int j = 2; j < WIDTH-2; j++) {

float sum = 0.0;

#pragma omp simd reduction(+:sum)

for(int ki = -2; ki <= 2; ki++) {

for(int kj = -2; kj <= 2; kj++) {

sum += input[i+ki][j+kj] \*

kernel[ki+2][kj+2];

}

}

#pragma omp critical

output[i][j] = sum;

}

}

}

int main() {

static float input[HEIGHT][WIDTH];

static float output[HEIGHT][WIDTH];

double start = omp\_get\_wtime();

gaussian\_blur\_parallel(input, output);

double end = omp\_get\_wtime();

printf("Parallel time (%d threads): %.4f seconds\n",

omp\_get\_max\_threads(), end - start);

return 0;

}

# output

