Reyhan Ahnaf Deannova	:	2306267100
M Hilmi Al Muttaqi Budiyanto	:	2306267082
Adhikananda Wira Januar	:	2306267113

Laporan Komputasi Numerik

FAKULTAS TEKNIK

UNIVERSITAS INDONESIA

DEPOK

2025

Nomor 1:

Hasil perkirakan nilai yang hilang berdasarkan pertanyaan yang diberikan pada soal:

a. 230,971,878 jiwa
b. 234,017,107 jiwa
c. 261,700,485 jiwa
d. 264,5523,571 jiwa
e. 3.66%
f. 4.72%
g. 22.21%
h. 27.27%

Screenshot Keseluruhan:

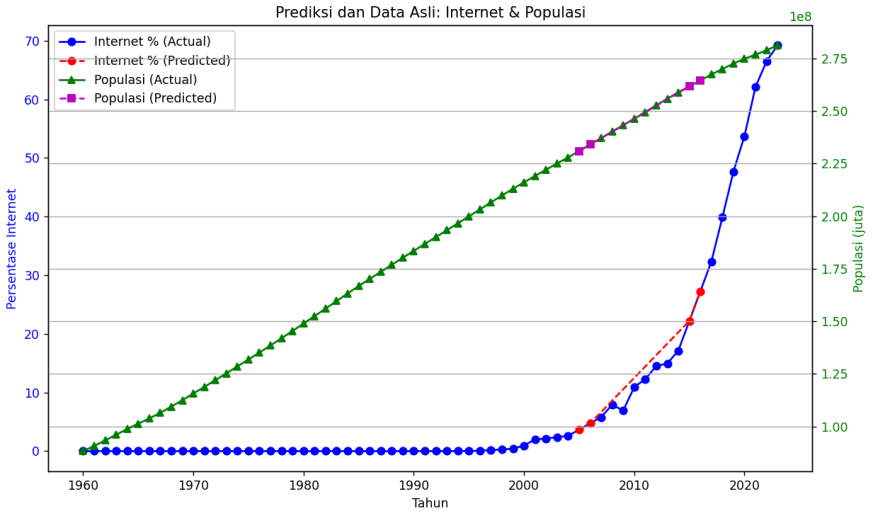
```
PS C:\Folder Kuliah\Semester 4\Komnum\Tugas Pemrograman A> cd "c:\Folder
$?) { .\coba4 }
Tahun 2005:
  - Estimasi Jumlah Penduduk      : 230971878
  - Estimasi Persentase Internet  : 3.66%

Tahun 2006:
  - Estimasi Jumlah Penduduk      : 234017107
  - Estimasi Persentase Internet  : 4.72%

Tahun 2015:
  - Estimasi Jumlah Penduduk      : 261700485
  - Estimasi Persentase Internet  : 22.21%

Tahun 2016:
  - Estimasi Jumlah Penduduk      : 264523571
  - Estimasi Persentase Internet  : 27.27%
```

Gambar Kurva:



Gambar 1. Kurva Prediksi Pengguna Internet & Populasi Berdasarkan Data Asli Dengan Data Prediksi

Nomor 2:

Dalam mendapatkan prediksi data yang sebelumnya didapatkan pada nomor 1, berikut merupakan formula persamaan nominal yang digunakan untuk mendapatkan data prediksi yang diminta oleh soal nomor 1 dan soal nomor 3 terkait persentase pertumbuhan pengguna internet:

a) Persentase Pengguna Internet:

y = +1.272763e-03 x^3 -7.570200e+00 x^2 +1.500837e+04 x^1 -9.918043e+06

b) Pertumbuhan Populasi:

y = -2.985498e+02 x^3 +1.780023e+06 x^2 -3.534244e+09 x^1 +2.337010e+12

Nomor 3:

Berkaitan dengan jumlah populasi Indonesia di tahun 2030 dan jumlah pengguna internet di Indonesia pada tahun 2035, hasil perhitungan pada program menunjukkan jumlah populasi Indonesia di tahun 2030 ialah sebanyak 295384560 jiwa dan untuk pengguna internet di tahun 2035 berada pada kisaran 89.12% dari jumlah penduduk di Indonesia dengan jumlah total penduduk sebesar 303205546 jiwa atau berkisar 270216782 jiwa yang akan menggunakan internet di tahun 2035.

Github Repository:

https://github.com/Nidakula/TugasPemrogramanA_Kelompok_19

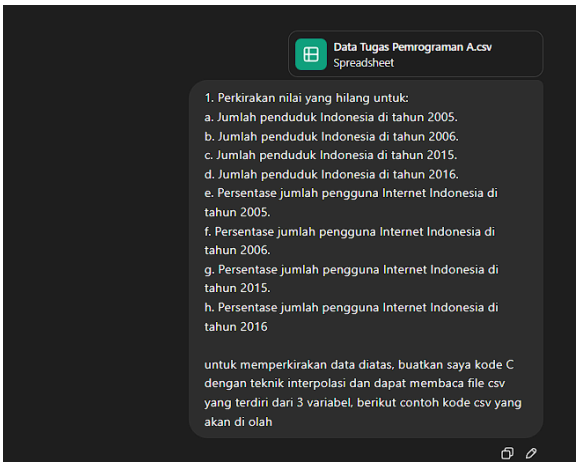
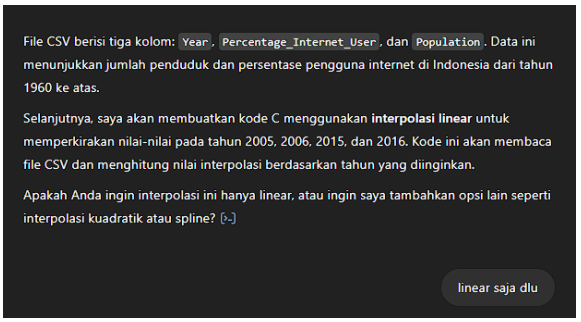
https://github.com/Nidakula/TugasPemrogramanA_Kelompok_19

https://github.com/Nidakula/TugasPemrogramanA_Kelompok_19

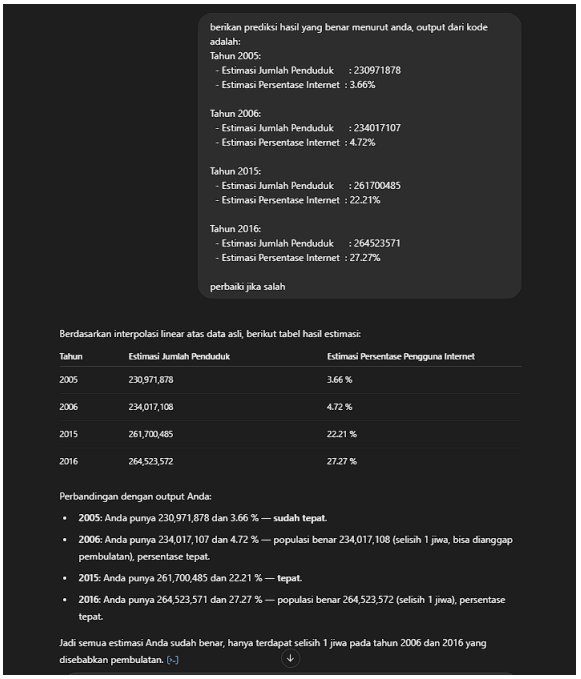
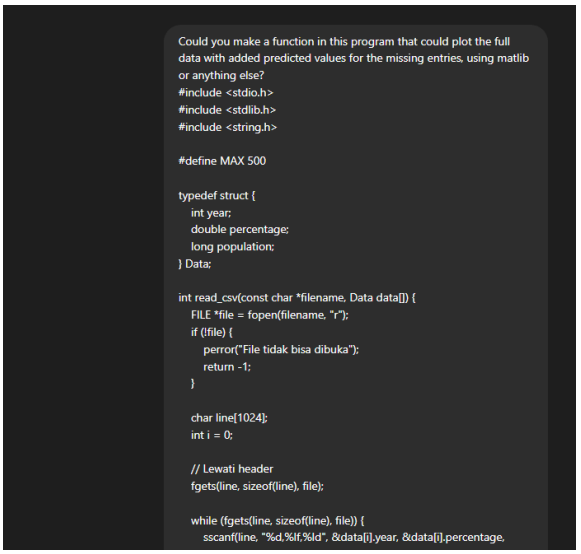
Lampiran

Lampiran 1: Dokumentasi AI

Kode 1:

Keterangan	Screenshot
Prompt pertama, menanyakan program interpolasi untuk nomor 1	
Opsi jenis interpolasi	

Kode interpolasi	Berikut adalah kode C untuk memperkirakan data jumlah penduduk dan persentase pengguna internet di tahun-tahun yang hilang (2005, 2006, 2015, 2016) menggunakan interpolasi linear dan membaca data dari file CSV dengan format seperti yang Anda unggah: Penjelasan: • Program ini membaca file CSV, menyimpan data dalam array `data`. • Fungsi `linear_interpolate` digunakan untuk nilai pecahan (persentase), dan `linear_interpolate_long` untuk populasi. • Fungsi `estimate` melakukan interpolasi linear untuk setiap tahun target. • Pastikan file CSV berada di direktori yang sama dengan program dan bernama `Data Tugas Pemrograman A.csv`. Apakah Anda ingin saya bantu dengan versi yang bisa dijalankan langsung (Makefile atau versi dengan input interaktif)?

Crosscheck jawaban	
Prompt awal untuk menambahkan fitur plotting	

```
void estimate(Data data[], int size, int target_years[], int target_size) {
    for (int i = 0; i < target_size; i++) {
        int year = target_years[i];
        for (int j = 0; j < size - 1; j++) {
            if (data[j].year <= year && data[j+1].year >= year) {
                long est_population = linear_interpolate_long(data[j].year,
data[j].population,
                                                              data[j+1].year,
data[j+1].population, year);
                double est_percentage = linear_interpolate(data[j].year,
data[j].percentage,
                                                           data[j+1].year,
data[j+1].percentage, year);
                printf("Tahun %d:\n", year);
                printf("  - Estimasi Jumlah Penduduk      : %ld\n",
est_population);
                printf("  - Estimasi Persentase Internet  : %.2f%%\n\n",
est_percentage);
                break;
            }
        }
    }
}

int main() {
    Data data[MAX];
    int size = read_csv("Data Tugas Pemrograman A.csv", data);

    if (size == -1) return 1;

    int target_years[] = {2005, 2006, 2015, 2016};
    int target_size = sizeof(target_years) / sizeof(target_years[0]);
```

```
    while (fgets(line, sizeof(line), file)) {
        sscanf(line, "%d,%lf,%ld", &data[i].year, &data[i].percentage,
&data[i].population);
        i++;
    }

    fclose(file);
    return i;
}

double linear_interpolate(int x0, double y0, int x1, double y1, int x) {
    return y0 + (double)(x - x0) * (y1 - y0) / (x1 - x0);
}

long linear_interpolate_long(int x0, long y0, int x1, long y1, int x) {
    return y0 + (long)((double)(x - x0) * (y1 - y0) / (x1 - x0));
}

void estimate(Data data[], int size, int target_years[], int target_size) {
    for (int i = 0; i < target_size; i++) {
        int year = target_years[i];
        for (int j = 0; j < size - 1; j++) {
            if (data[j].year <= year && data[j+1].year >= year) {
                long est_population = linear_interpolate_long(data[j].year,
data[j].population,
                                                              data[j+1].year,
data[j+1].population, year);
                double est_percentage = linear_interpolate(data[j].year,
data[j].percentage,
                                                           data[j+1].year,
data[j+1].percentage, year);
                printf("Tahun %d:\n", year);
```

```
    int target_years[] = {2005, 2006, 2015, 2016};
    int target_size = sizeof(target_years) / sizeof(target_years[0]);

    estimate(data, size, target_years, target_size);

    return 0;
}
```

To add plotting functionality for the full data—including predicted (interpolated) values—you can integrate `matplotlib` using the Python C API, but the easier and more practical solution is to **export the data to a file and use a Python script to plot** it.

Here's how you can do it in **two parts**:

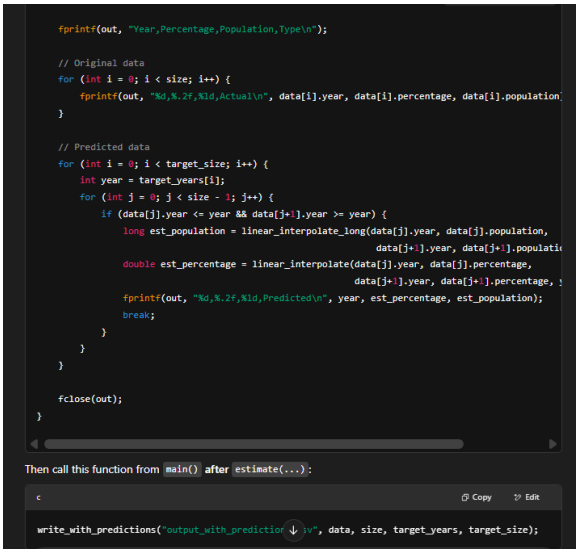
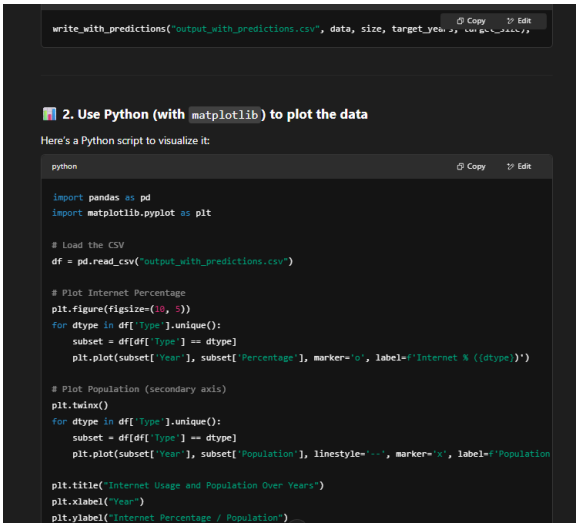
🔧 1. Modify the C program to output actual + predicted data to a CSV file

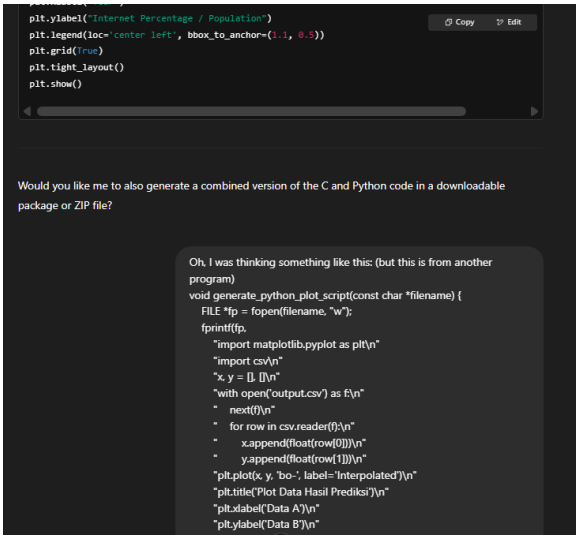
Add this function to write both original and estimated data to a file:

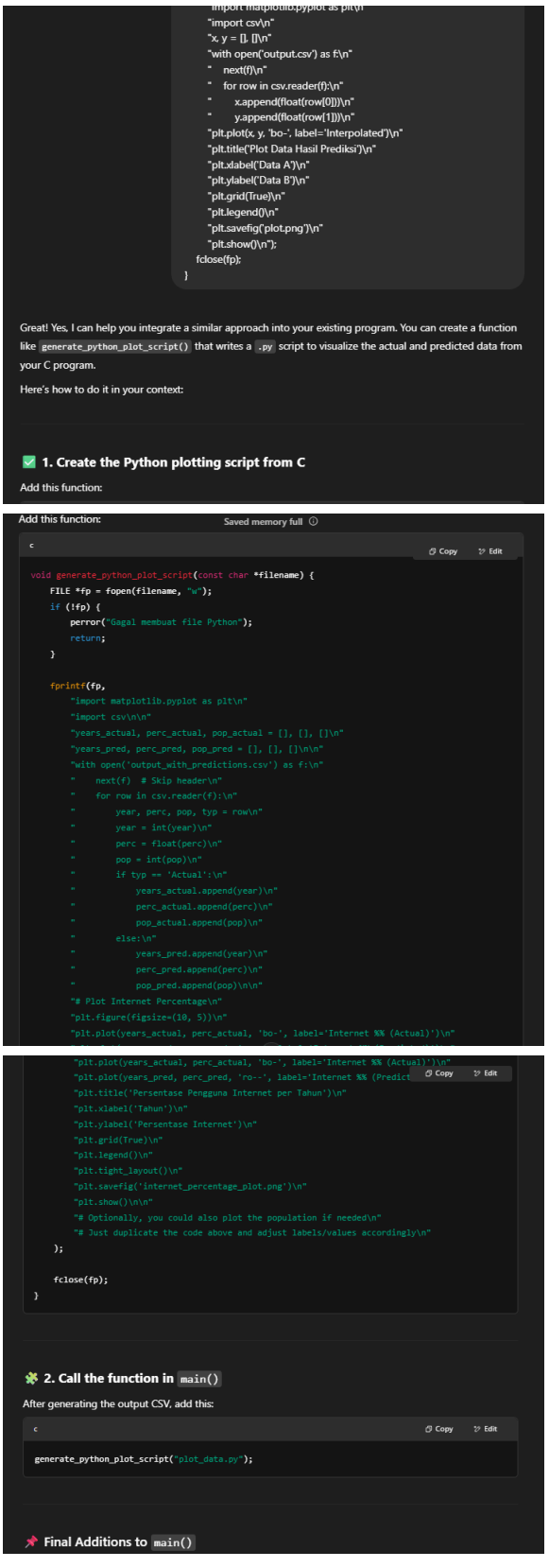
c — Copy — Edit

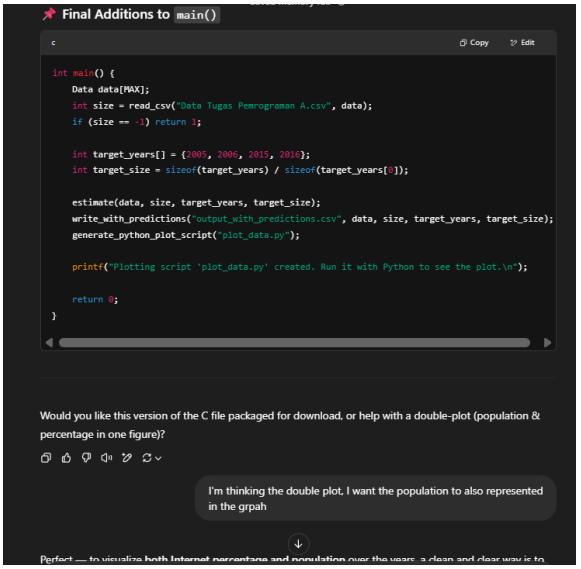
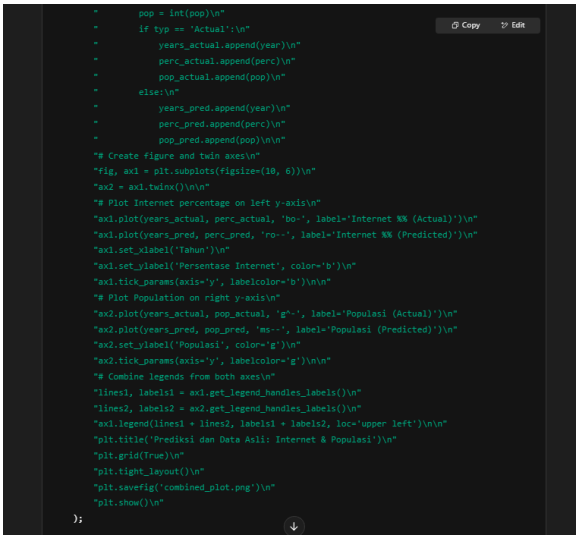
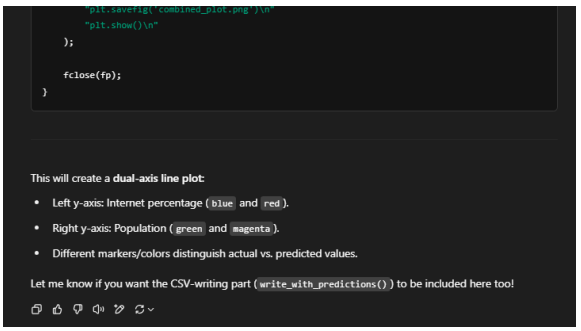
```
void write_with_predictions(const char *output_file, Data data[], int size, int target_years[], i
    FILE *out = fopen(output_file, "w");
    if (!out) {
        perror("Gagal membuka file output");
        return;
    }

    fprintf(out, "Year,Percentage,Population,Type\n");
```

	
	Then call this function from `main()` **after** `estimate(...)`:
	c Copy Edit
	write_with_predictions("output_with_prediction ↓ v", data, size, target_years, target_size);
	 
	write_with_predictions("output_with_predictions.csv", data, size, target_yea
	2. Use Python (with `matplotlib`) to plot the data
	Here's a Python script to visualize it:
	python Copy Edit
	import pandas as pd
	import matplotlib.pyplot as plt
	# Load the CSV
	df = pd.read_csv("output_with_predictions.csv")
	# Plot Internet Percentage
	plt.figure(figsize=(10, 5))
	for dtype in df['Type'].unique():
	subset = df[df['Type'] == dtype]
	plt.plot(subset['Year'], subset['Percentage'], marker='o', label=f'Internet % ({dtype})')
	# Plot Population (secondary axis)
	plt.twinx()
	for dtype in df['Type'].unique():
	subset = df[df['Type'] == dtype]
	plt.plot(subset['Year'], subset['Population'], linestyle='--', marker='x', label=f'Population
	plt.title("Internet Usage and Population Over Years")
	plt.xlabel("Year")
	plt.ylabel("Internet Percentage / Population")

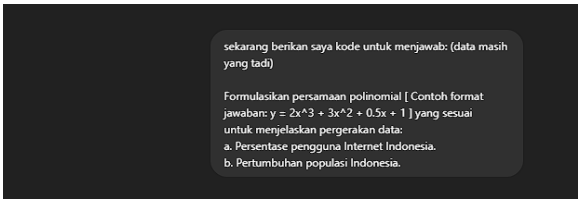
Prompt untuk memodifikasi agar kode C mampu men-generate script python untuk plotting	 plt.ylabel("Internet Percentage / Population") Copy Edit plt.legend(loc='center left', bbox_to_anchor=(1.1, 0.5)) plt.grid(True) plt.tight_layout() plt.show() Would you like me to also generate a combined version of the C and Python code in a downloadable package or ZIP file? Oh, I was thinking something like this: (but this is from another program) void generate_python_plot_script(const char *filename) { FILE *fp = fopen(filename, "w"); fprintf(fp, "import matplotlib.pyplot as plt\n" "import csv\n" "x, y = [], []\n" "with open('output.csv') as f:\n" " next(f)\n" " for row in csv.reader(f):\n" " x.append(float(row[0]))\n" " y.append(float(row[1]))\n" "plt.plot(x, y, 'bo-', label='Interpolated')\n" "plt.title('Plot Data Hasil Prediksi')\n" "plt.xlabel('Data A')\n" "plt.ylabel('Data B')\n"


"import csv\n"
"x, y = [], []\n"
"with open('output.csv') as f:\n"
"   next(f)\n"
"   for row in csv.reader(f):\n"
"   x.append(float(row[0]))\n"
"   y.append(float(row[1]))\n"
"plt.plot(x, y, 'bo-', label='Interpolated')\n"
"plt.title('Plot Data Hasil Prediksi')\n"
"plt.xlabel('Data A')\n"
"plt.ylabel('Data B')\n"
"plt.grid(True)\n"
"plt.legend()\n"
"plt.savefig('plot.png')\n"
"plt.show()\n");
fclose(fp);
}
Great! Yes, I can help you integrate a similar approach into your existing program. You can create a function like generate_python_plot_script() that writes a .py script to visualize the actual and predicted data from your C program.
Here's how to do it in your context:
1. Create the Python plotting script from C
Add this function:
Add this function:
Saved memory full
c
Copy
Edit
void generate_python_plot_script(const char *filename) {
FILE *fp = fopen(filename, "w");
if (!fp) {
perror("Gagal membuat file Python");
return;
}
fprintf(fp,
"import matplotlib.pyplot as plt\n"
"import csv\n\n"
"years_actual, perc_actual, pop_actual = [], [], []\n"
"years_pred, perc_pred, pop_pred = [], [], []\n\n"
"with open('output_with_predictions.csv') as f:\n"
"   next(f) # Skip header\n"
"   for row in csv.reader(f):\n"
"   year, perc, pop, typ = row\n"
"   year = int(year)\n"
"   perc = float(perc)\n"
"   pop = int(pop)\n"
"   if typ == 'Actual':\n"
"   years_actual.append(year)\n"
"   perc_actual.append(perc)\n"
"   pop_actual.append(pop)\n"
"   else:\n"
"   years_pred.append(year)\n"
"   perc_pred.append(perc)\n"
"   pop_pred.append(pop)\n\n"
"# Plot Internet Percentage\n"
"plt.figure(figsize=(10, 5))\n"
"plt.plot(years_actual, perc_actual, 'bo-', label='Internet %% (Actual)')\n"
"plt.plot(years_actual, perc_actual, 'bo-', label='Internet %% (Actual)')\n"
"plt.plot(years_pred, perc_pred, 'ro--', label='Internet %% (Predict
Copy
Edit
"plt.title('Persentase Pengguna Internet per Tahun')\n"
"plt.xlabel('Tahun')\n"
"plt.ylabel('Persentase Internet')\n"
"plt.grid(True)\n"
"plt.legend()\n"
"plt.tight_layout()\n"
"plt.savefig('internet_percentage_plot.png')\n"
"plt.show()\n\n"
"# Optionally, you could also plot the population if needed\n"
"# Just duplicate the code above and adjust labels/values accordingly\n"
);
fclose(fp);
}
2. Call the function in main()
After generating the output CSV, add this:
c
Copy
Edit
generate_python_plot_script("plot_data.py");
Final Additions to main()

Prompt untuk modifikasi program agar populasi dan persentase pengguna internet direpresentasikan dalam grafik yang sama	  

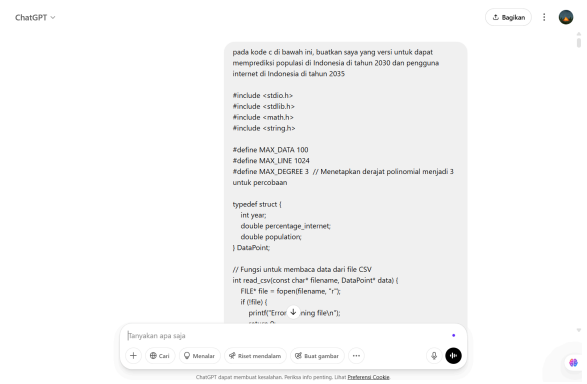
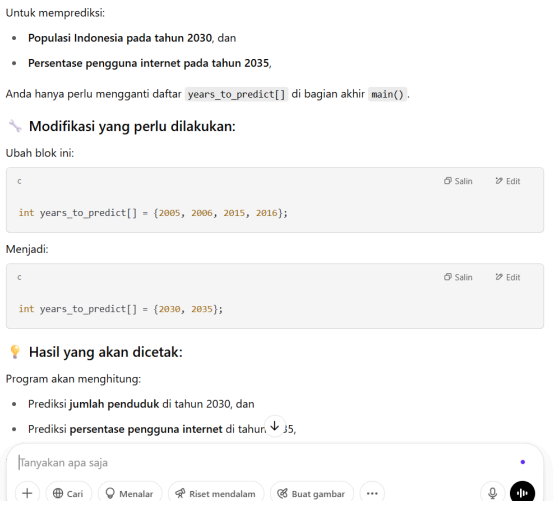
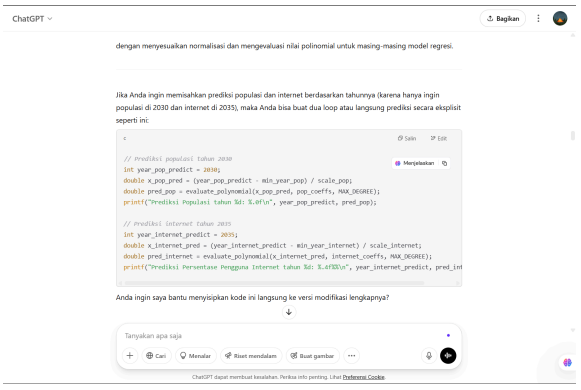
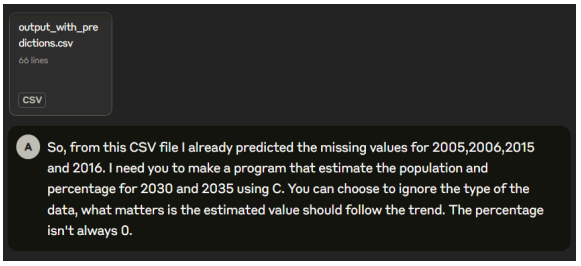
Kode 2:

Keterangan	Screenshot

Prompt pertama untuk menanyakan formulasi persamaan polinomial	

Hasilnya dengan menggunakan bahasa C	

Kode 3:

Keterangan	Screenshot
Prompt pertama untuk menanyakan function yang diubah berdasarkan kode pertama	
Function yang direkomendasikan oleh AI untuk mendapatkan hasil yang diminta oleh soal nomor 3	
Membuat loop yang dapat memprediksi langsung untuk jumlah penduduk dan presentase pengguna internet di tahun 2030 dan 2035	
Prompt untuk mengestimasi data untuk tahun 2030 dan 2035	

Penjelasan kode dan hasil kode	I'll create a C program to estimate population and percentage values for 2030 and 2035, following the trends observed in your data. Looking at your CSV file, I see we have actual data from 1960 to 2023 with percentage values that increase over time, and you've already made predictions for some missing years. Let me create a program that will analyze this data and make predictions for 2030 and 2035. Population and Percentage Prediction for 2030 and 2035 Code I've created a C program that will analyze your CSV data and predict population and percentage values for 2030 and 2035. Let me explain how the program works: **How the Program Works** 1. **Data Handling**: The program reads the CSV file and stores all data points in a structured format. 2. **Population Prediction**: • Uses an exponential growth model based on the most recent 10 years of data • Calculates the average annual growth rate and applies it to project future population • This method accounts for the consistent growth pattern in population that your data shows 3. **Percentage Prediction**: • Focuses on the data after 1994 (when percentages start to increase) • Uses a linear regression model as a base calculation • Applies a logistic growth adjustment to slow growth as percentages approach 100% • This ensures predictions follow the S-curve pattern typical of technology adoption • This method accounts for the consistent growth pattern in population that your data shows 3. **Percentage Prediction**: • Focuses on the data after 1994 (when percentages start to increase) • Uses a linear regression model as a base calculation • Applies a logistic growth adjustment to slow growth as percentages approach 100% • This ensures predictions follow the S-curve pattern typical of technology adoption **How to Use the Program** 1. Make sure the CSV file "output_with_predictions.csv" is in the same directory as the program 2. Compile the program with: `gcc -o predict_population prediction_program.c -lm` 3. Run the program: `./predict_population` 4. The program will output all the existing data along with the predictions for 2030 and 2035 **Expected Results** The program will predict values following the trends shown in your data: • Population will continue its steady growth pattern • Percentage will follow the accelerating adoption curve, with growth slowing as it approaches 100% Note: The program requires the `math.h` library for exponential calculations, so include the `-lm` flag when compiling. Would you like me to explain any part of the code in more detail or make any adjustments to the prediction methodology?

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#define MAX_ROWS 100
#define MAX_LINE_LENGTH 100

typedef struct {
    int year;
    double percentage;
    double population;
    char type[10];
} DataRow;

// Function to read CSV data
int readCSV(const char* filename, DataRow data[], int* rowCount) {
    FILE* file = fopen(filename, "r");
    if (!file) {
        printf("Error: Could not open file %s\n", filename);
        return 0;
    }

    char line[MAX_LINE_LENGTH];
    int i = 0;

    // Skip header
    fgets(line, MAX_LINE_LENGTH, file);

    while (fgets(line, MAX_LINE_LENGTH, file) && i < MAX_ROWS) {
        char* token = strtok(line, ",");
        if (token) {
            data[i].year = atoi(token);
        }

        token = strtok(NULL, ",");
        if (token) {
```

```c
        token = strtok(NULL, ",");
        if (token) {
            data[i].percentage = atof(token);
        }

        token = strtok(NULL, ",");
        if (token) {
            data[i].population = atof(token);
        }

        token = strtok(NULL, "\n");
        if (token) {
            strncpy(data[i].type, token, 10);
        }

        i++;
    }

    *rowCount = i;
    fclose(file);
    return 1;
}

// Function to predict population using exponential growth model
double predictPopulation(DataRow* data, int rowCount, int targetYear) {
    // Use recent years for more accurate prediction
    int startIdx = rowCount - 10;
    if (startIdx < 0) startIdx = 0;

    // Calculate average growth rate from recent years
    double growthRate = 0.0;
    int countValid = 0;

    for (int i = startIdx; i < rowCount - 1; i++) {
        if (data[i].population > 0 && data[i+1].population > 0) {
            double annualRate = data[i+1].population / data[i].population - 1.0;
            growthRate += annualRate;
```

```
    for (int i = startIdx; i < rowCount - 1; i++) {
        if (data[i].population > 0 && data[i+1].population > 0) {
            double annualRate = data[i+1].population / data[i].population - 1.0;
            growthRate += annualRate;
            countValid++;
        }
    }

    growthRate /= countValid; // Average growth rate

    // Get most recent year with data
    DataRow latestData = data[rowCount - 1];

    // Calculate prediction using compound growth
    int yearDiff = targetYear - latestData.year;
    double predictedPopulation = latestData.population * pow(1 + growthRate, yearDiff);

    return predictedPopulation;
}

// Function to predict percentage using polynomial regression
double predictPercentage(DataRow* data, int rowCount, int targetYear) {
    // Focus on the years where percentage started increasing (after 1994)
    int startYear = 1994;
    int startIdx = 0;

    // Find the index for the start year
    for (int i = 0; i < rowCount; i++) {
        if (data[i].year >= startYear) {
            startIdx = i;
            break;
        }
    }

    // Prepare data for regression (normalize years for numerical stability)
    int n = rowCount - startIdx;
```

```
    int n = rowCount - startIdx;
    int baseYear = data[startIdx].year;

    double x[MAX_ROWS], y[MAX_ROWS];
    for (int i = 0; i < n; i++) {
        x[i] = data[startIdx + i].year - baseYear;
        y[i] = data[startIdx + i].percentage;
    }

    // Calculate coefficients for polynomial regression (degree 3)
    // This is a simplified approach - for real-world models consider libraries like GSL
    double sumX = 0, sumX2 = 0, sumX3 = 0, sumX4 = 0, sumX5 = 0, sumX6 = 0;
    double sumY = 0, sumXY = 0, sumX2Y = 0, sumX3Y = 0;

    for (int i = 0; i < n; i++) {
        double xi = x[i];
        double xi2 = xi * xi;
        double xi3 = xi2 * xi;
        double yi = y[i];

        sumX += xi;
        sumX2 += xi2;
        sumX3 += xi2 * xi;
        sumX4 += xi2 * xi2;
        sumX5 += xi3 * xi2;
        sumX6 += xi3 * xi3;

        sumY += yi;
        sumXY += xi * yi;
        sumX2Y += xi2 * yi;
        sumX3Y += xi3 * yi;
    }

    // Solve the normal equations for cubic regression
    // [n      sumX    sumX2   sumX3  ] [a0] = [sumY  ]
    // [sumX   sumX2   sumX3   sumX4  ] [a1] = [sumXY ]
    // [sumX2  sumX3   sumX4   sumX5  ] [a2] = [sumX2Y]
```

```
    // [n      sumX    sumX2   sumX3  ] [a0] = [sumY  ]
    // [sumX   sumX2   sumX3   sumX4  ] [a1] = [sumXY ]
    // [sumX2  sumX3   sumX4   sumX5  ] [a2] = [sumX2Y]
    // [sumX3  sumX4   sumX5   sumX6  ] [a3] = [sumX3Y]

    // This is a simplified approach for cubic regression - in practice, use a library
    // for matrix operations or solve the system using Gaussian elimination

    // For simplicity, let's use linear regression as a fallback
    double meanX = sumX / n;
    double meanY = sumY / n;
    double numerator = 0, denominator = 0;

    for (int i = 0; i < n; i++) {
        numerator += (x[i] - meanX) * (y[i] - meanY);
        denominator += (x[i] - meanX) * (x[i] - meanX);
    }

    double slope = numerator / denominator;
    double intercept = meanY - slope * meanX;

    // Linear with saturation - we don't want percentage to exceed 100%
    double normalizedYear = targetYear - baseYear;
    double predictedPercentage = intercept + slope * normalizedYear;

    // Cap at 100% and ensure it doesn't decrease if recent trend is upward
    if (predictedPercentage > 100.0)
        predictedPercentage = 100.0;

    // Adjust using logistic function characteristics for S-curve growth
    // This gives more realistic percentage growth that slows as it approaches 100%
    if (predictedPercentage > 70.0) {
        // Slow down growth as we approach 100%
        double latest = data[rowCount - 1].percentage;
        double remainingGrowth = 100.0 - latest;
        double yearsSinceLatest = targetYear - data[rowCount - 1].year;
        double adjustedGrowth = remainingGrowth * (1.0 - exp(-0.1 * yearsSinceLatest));
```

```
        double latest = data[rowCount - 1].percentage;
        double remainingGrowth = 100.0 - latest;
        double yearsSinceLatest = targetYear - data[rowCount - 1].year;
        double adjustedGrowth = remainingGrowth * (1.0 - exp(-0.1 * yearsSinceLatest));
        predictedPercentage = latest + adjustedGrowth;
    }

    return predictedPercentage;
}

// Print results in CSV format
void printCSVRow(int year, double percentage, double population, const char* type) {
    printf("%d,%.2f,%.0f,%s\n", year, percentage, population, type);
}

int main() {
    DataRow data[MAX_ROWS];
    int rowCount = 0;

    // Read data from CSV file
    if (!readCSV("output_with_predictions.csv", data, &rowCount)) {
        return 1;
    }

    printf("Year,Percentage,Population,Type\n");

    // Print existing data for verification
    for (int i = 0; i < rowCount; i++) {
        printCSVRow(data[i].year, data[i].percentage, data[i].population, data[i].type);
    }

    // Make predictions for 2030 and 2035
    int targetYears[] = {2030, 2035};
    for (int i = 0; i < 2; i++) {
        double predictedPop = predictPopulation(data, rowCount, targetYears[i]);
        double predictedPerc = predictPercentage(data, rowCount, targetYears[i]);
        printCSVRow(targetYears[i], predictedPerc, predictedPop, "Predicted");
```

```
    for (int i = 0; i < 2; i++) {
        double predictedPop = predictPopulation(data, rowCount, targetYears[i]);
        double predictedPerc = predictPercentage(data, rowCount, targetYears[i]);
        printCSVRow(targetYears[i], predictedPerc, predictedPop, "Predicted");
    }

    return 0;
}
```

Lampiran 2: Kode

Kode 1:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX 500

typedef struct {
    int year;
    double percentage;
    long population;
} Data;

int read_csv(const char *filename, Data data[]) {
    FILE *file = fopen(filename, "r");
    if (!file) {
        perror("File tidak bisa dibuka");
        return -1;
    }

    char line[1024];
    int i = 0;

    // Lewati header
    fgets(line, sizeof(line), file);

    while (fgets(line, sizeof(line), file)) {
        sscanf(line, "%d,%lf,%ld", &data[i].year, &data[i].percentage, &data[i].population);
        i++;
    }

    fclose(file);
    return i;
}

double linear_interpolate(int x0, double y0, int x1, double y1, int x) {
    return y0 + (double)(x - x0) * (y1 - y0) / (x1 - x0);
}

long linear_interpolate_long(int x0, long y0, int x1, long y1, int x) {
    return y0 + (long)((double)(x - x0) * (y1 - y0) / (x1 - x0));
}

void estimate(Data data[], int size, int target_years[], int target_size) {
    for (int i = 0; i < target_size; i++) {
        int year = target_years[i];
        for (int j = 0; j < size - 1; j++) {
```

```
            if (data[j].year <= year && data[j+1].year >= year) {
                long est_population = linear_interpolate_long(data[j].year, data[j].population,
                                                    data[j+1].year, data[j+1].population, year);
                double est_percentage = linear_interpolate(data[j].year, data[j].percentage,
                                                  data[j+1].year, data[j+1].percentage, year);
                printf("Tahun %d:\n", year);
                printf("  - Estimasi Jumlah Penduduk      : %ld\n", est_population);
                printf("  - Estimasi Persentase Internet  : %.2f%%\n\n", est_percentage);
                break;
            }
        }
    }
}

void write_with_predictions(const char *output_file, Data data[], int size, int
target_years[], int target_size) {
    FILE *out = fopen(output_file, "w");
    if (!out) {
        perror("Gagal membuka file output");
        return;
    }

    fprintf(out, "Year,Percentage,Population,Type\n");

    // Original data
    for (int i = 0; i < size; i++) {
        fprintf(out, "%d,%.2f,%ld,Actual\n", data[i].year, data[i].percentage,
data[i].population);
    }

    // Predicted data
    for (int i = 0; i < target_size; i++) {
        int year = target_years[i];
        for (int j = 0; j < size - 1; j++) {
            if (data[j].year <= year && data[j+1].year >= year) {
                long est_population = linear_interpolate_long(data[j].year, data[j].population,
                                                    data[j+1].year, data[j+1].population, year);
                double est_percentage = linear_interpolate(data[j].year, data[j].percentage,
                                                  data[j+1].year, data[j+1].percentage, year);
                fprintf(out, "%d,%.2f,%ld,Predicted\n", year, est_percentage, est_population);
                break;
            }
        }
    }

    fclose(out);
}

void generate_python_plot_script(const char *filename) {
```

```
FILE *fp = fopen(filename, "w");
if (!fp) {
    perror("Gagal membuat file Python");
    return;
}

fprintf(fp,
    "import matplotlib.pyplot as plt\n"
    "import csv\n\n"
    "years_actual, perc_actual, pop_actual = [], [], []\n"
    "years_pred, perc_pred, pop_pred = [], [], []\n\n"
    "with open('output_with_predictions.csv') as f:\n"
    "    next(f)  # Skip header\n"
    "    for row in csv.reader(f):\n"
    "        year, perc, pop, typ = row\n"
    "        year = int(year)\n"
    "        perc = float(perc)\n"
    "        pop = int(pop)\n"
    "        if typ == 'Actual':\n"
    "            years_actual.append(year)\n"
    "            perc_actual.append(perc)\n"
    "            pop_actual.append(pop)\n"
    "        else:\n"
    "            years_pred.append(year)\n"
    "            perc_pred.append(perc)\n"
    "            pop_pred.append(pop)\n\n"
    "# Create figure and twin axes\n"
    "fig, ax1 = plt.subplots(figsize=(10, 6))\n"
    "ax2 = ax1.twinx()\n\n"
    "# Plot Internet percentage on left y-axis\n"
    "ax1.plot(years_actual, perc_actual, 'bo-', label='Internet %% (Actual)')\n"
    "ax1.plot(years_pred, perc_pred, 'ro--', label='Internet %% (Predicted)')\n"
    "ax1.set_xlabel('Tahun')\n"
    "ax1.set_ylabel('Persentase Internet', color='b')\n"
    "ax1.tick_params(axis='y', labelcolor='b')\n\n"
    "# Plot Population on right y-axis\n"
    "ax2.plot(years_actual, pop_actual, 'g^-', label='Populasi (Actual)')\n"
    "ax2.plot(years_pred, pop_pred, 'ms--', label='Populasi (Predicted)')\n"
    "ax2.set_ylabel('Populasi (juta)', color='g')\n"
    "ax2.tick_params(axis='y', labelcolor='g')\n\n"
    "# Combine legends from both axes\n"
    "lines1, labels1 = ax1.get_legend_handles_labels()\n"
    "lines2, labels2 = ax2.get_legend_handles_labels()\n"
    "ax1.legend(lines1 + lines2, labels1 + labels2, loc='upper left')\n\n"
    "plt.title('Prediksi dan Data Asli: Internet & Populasi')\n"
    "plt.grid(True)\n"
    "plt.tight_layout()\n"
    "plt.savefig('combined_plot.png')\n"
    "plt.show()\n"
```

```
    );

    fclose(fp);
}

int main() {
    Data data[MAX];
    int size = read_csv("Data Tugas Pemrograman A.csv", data);

    if (size == -1) return 1;

    int target_years[] = {2005, 2006, 2015, 2016};
    int target_size = sizeof(target_years) / sizeof(target_years[0]);

    estimate(data, size, target_years, target_size);
    write_with_predictions("output_with_predictions.csv", data, size, target_years,
target_size);
    generate_python_plot_script("plot.py");
    return 0;
}
```

Kode 2:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#define MAX_DATA 1000     // Maksimum baris data
#define DEGREE   3        // Derajat polinomial

// Fungsi untuk membaca CSV
int read_csv(const char *fname, double x[], double y1[], double y2[]) {
    FILE *fp = fopen(fname, "r");
    if (!fp) {
        perror("Tidak bisa membuka file");
        return -1;
    }
    char line[256];
    int n = 0;
    // Lewati header
    if (!fgets(line, sizeof(line), fp)) {
        fclose(fp);
        return 0;
    }
    // Baca setiap baris
    while (fgets(line, sizeof(line), fp)) {
        int year;
```

```
        double pct;
        long pop;
        if (sscanf(line, "%d,%lf,%ld", &year, &pct, &pop) == 3) {
            x[n]  = year;
            y1[n] = pct;
            y2[n] = (double)pop;
            n++;
            if (n >= MAX_DATA) break;
        }
    }
    fclose(fp);
    return n;
}

// Eliminasi Gauss untuk menyelesaikan A * c = b
void gaussian_elim(int m, double A[][DEGREE+2], double c[]) {
    int i, j, k;
    for (i = 0; i < m; i++) {
        // Pivoting sederhana: cari baris terbesar
        int piv = i;
        for (j = i+1; j < m; j++) {
            if (fabs(A[j][i]) > fabs(A[piv][i])) piv = j;
        }
        // Tukar baris
        if (piv != i) {
            for (k = i; k <= m; k++)
                { double tmp = A[i][k]; A[i][k] = A[piv][k]; A[piv][k] = tmp; }
        }
        // Normalisasi dan eliminasi
        double div = A[i][i];
        for (k = i; k <= m; k++) A[i][k] /= div;
        for (j = 0; j < m; j++) {
            if (j == i) continue;
            double factor = A[j][i];
            for (k = i; k <= m; k++)
                A[j][k] -= factor * A[i][k];
        }
    }
    // Ambil solusi
    for (i = 0; i < m; i++) c[i] = A[i][m];
}

// Fitting polinomial degree = DEGREE
void polyfit(int n, const double x[], const double y[], double coeffs[]) {
    int m = DEGREE + 1;
    // Matriks normal berukuran m x (m+1)
    double N[DEGREE+1][DEGREE+2];
    memset(N, 0, sizeof(N));
```

```
    // Hitung elemen matriks
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < m; j++) {
            double sum = 0;
            for (int k = 0; k < n; k++)
                sum += pow(x[k], i+j);
            N[i][j] = sum;
        }
        // vektor sebelah kanan
        double sumy = 0;
        for (int k = 0; k < n; k++)
            sumy += y[k] * pow(x[k], i);
        N[i][m] = sumy;
    }

    // Selesaikan dengan eliminasi Gauss
    gaussian_elim(m, N, coeffs);
}

int main() {
    double x[MAX_DATA], pct[MAX_DATA], pop[MAX_DATA];
    int n = read_csv("Data Tugas Pemrograman A.csv", x, pct, pop);
    if (n <= 0) {
        fprintf(stderr, "Gagal membaca data\n");
        return 1;
    }

    double c_pct[DEGREE+1], c_pop[DEGREE+1];

    // Fitting
    polyfit(n, x, pct, c_pct);
    polyfit(n, x, pop, c_pop);

    // Cetak hasil
    printf("a) Persentase Pengguna Internet:\n");
    printf("   y = ");
    for (int i = DEGREE; i >= 0; i--) {
        printf("%+.6e", c_pct[i]);
        if (i > 0) printf(" x^%d ", i);
    }
    printf("\n\n");

    printf("b) Pertumbuhan Populasi:\n");
    printf("   y = ");
    for (int i = DEGREE; i >= 0; i--) {
        printf("%+.6e", c_pop[i]);
        if (i > 0) printf(" x^%d ", i);
    }
    printf("\n");
```

```
    return 0;
}
```

Kode 3:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

#define MAX_DATA 100
#define MAX_LINE 1024
#define MAX_DEGREE 3

typedef struct {
    int year;
    double percentage_internet;
    double population;
} DataPoint;

int read_csv(const char* filename, DataPoint* data) {
    FILE* file = fopen(filename, "r");
    if (!file) {
        printf("Error opening file\n");
        return 0;
    }

    char line[MAX_LINE];
    int count = 0;
    fgets(line, MAX_LINE, file); // Skip header

    while (fgets(line, MAX_LINE, file) && count < MAX_DATA) {
        char* token = strtok(line, ",");
        if (token) {
            data[count].year = atoi(token);
            token = strtok(NULL, ",");
            if (token) {
                data[count].percentage_internet = atof(token);
                token = strtok(NULL, ",");
                if (token) {
                    data[count].population = atof(token);
                    count++;
                }
            }
        }
    }
    fclose(file);
```

```
    return count;
}

void polynomial_regression(int n, double* x, double* y, int degree, double* coeffs) {
    int i, j, k;
    double X[2 * degree + 1];
    double B[degree + 1][degree + 2];

    for (i = 0; i <= 2 * degree; i++) {
        X[i] = 0;
        for (j = 0; j < n; j++)
            X[i] += pow(x[j], i);
    }

    for (i = 0; i <= degree; i++) {
        for (j = 0; j <= degree; j++) {
            B[i][j] = X[i + j];
        }
    }

    for (i = 0; i <= degree; i++) {
        B[i][degree + 1] = 0;
        for (j = 0; j < n; j++) {
            B[i][degree + 1] += pow(x[j], i) * y[j];
        }
    }

    for (i = 0; i <= degree; i++) {
        for (k = i + 1; k <= degree; k++) {
            double factor = B[k][i] / B[i][i];
            for (j = i; j <= degree + 1; j++) {
                B[k][j] -= factor * B[i][j];
            }
        }
    }

    for (i = degree; i >= 0; i--) {
        coeffs[i] = B[i][degree + 1];
        for (j = i + 1; j <= degree; j++) {
            coeffs[i] -= B[i][j] * coeffs[j];
        }
        coeffs[i] /= B[i][i];
    }
}

double evaluate_polynomial(double x, double* coeffs, int degree) {
    double result = 0;
    for (int i = 0; i <= degree; i++) {
        result += coeffs[i] * pow(x, i);
```

```
    }
    return result;
}

void print_polynomial(double* coeffs, int degree) {
    printf("y = ");
    int first_term = 1;
    for (int i = degree; i >= 0; i--) {
        if (coeffs[i] == 0) continue;

        if (coeffs[i] > 0 && !first_term) printf(" + ");
        else if (coeffs[i] < 0) printf(" - ");

        if (i == 0 || fabs(coeffs[i]) != 1) {
            printf("%.10g", fabs(coeffs[i]));
        }

        if (i > 0) {
            printf("x");
            if (i > 1) {
                printf("^%d", i);
            }
        }

        first_term = 0;
    }
    printf("\n");
}

void normalize_years(int n, double* x, double* years, double* min_year, double* scale) {
    *min_year = years[0];
    double max_year = years[0];

    for (int i = 1; i < n; i++) {
        if (years[i] < *min_year) *min_year = years[i];
        if (years[i] > max_year) max_year = years[i];
    }

    *scale = max_year - *min_year;
    if (*scale == 0) *scale = 1;

    for (int i = 0; i < n; i++) {
        x[i] = (years[i] - *min_year) / *scale;
    }
}

int main() {
    DataPoint data[MAX_DATA];
    int count = read_csv("Data Tugas Pemrograman A.csv", data);
```

```
    if (count == 0) {
        printf("No data read\n");
        return 1;
    }

    printf("Membaca %d baris data\n\n", count);

    double years[MAX_DATA], x_pop[MAX_DATA], y_pop[MAX_DATA];
    double x_internet[MAX_DATA], y_internet[MAX_DATA];
    int n_pop = 0, n_internet = 0;

    for (int i = 0; i < count; i++) {
        years[i] = data[i].year;

        x_pop[n_pop] = data[i].year;
        y_pop[n_pop] = data[i].population;
        n_pop++;

        if (data[i].year >= 1994) {
            x_internet[n_internet] = data[i].year;
            y_internet[n_internet] = data[i].percentage_internet;
            n_internet++;
        }
    }

    double min_year_pop, scale_pop;
    double min_year_internet, scale_internet;
    double x_pop_norm[MAX_DATA], x_internet_norm[MAX_DATA];

    normalize_years(n_pop, x_pop_norm, x_pop, &min_year_pop, &scale_pop);
    normalize_years(n_internet, x_internet_norm, x_internet, &min_year_internet,
&scale_internet);

    double pop_coeffs[MAX_DEGREE + 1] = {0};
    polynomial_regression(n_pop, x_pop_norm, y_pop, MAX_DEGREE, pop_coeffs);
    printf("Persamaan polinomial untuk pertumbuhan populasi:\n");
    print_polynomial(pop_coeffs, MAX_DEGREE);

    double internet_coeffs[MAX_DEGREE + 1] = {0};
    polynomial_regression(n_internet, x_internet_norm, y_internet, MAX_DEGREE,
internet_coeffs);
    printf("Persamaan polinomial untuk persentase pengguna internet:\n");
    print_polynomial(internet_coeffs, MAX_DEGREE);

    int years_to_predict[] = {2030, 2035};
    int num_years = sizeof(years_to_predict) / sizeof(years_to_predict[0]);

    printf("\n=== Hasil Prediksi ===\n");
    for (int i = 0; i < num_years; i++) {
```

```
        int year = years_to_predict[i];

        double x_pop_pred = (year - min_year_pop) / scale_pop;
        double x_internet_pred = (year - min_year_internet) / scale_internet;

        double pred_pop = evaluate_polynomial(x_pop_pred, pop_coeffs, MAX_DEGREE);
        double pred_internet = evaluate_polynomial(x_internet_pred, internet_coeffs,
MAX_DEGREE);

        if (pred_internet > 100.0) pred_internet = 100.0;
        if (pred_internet < 0.0) pred_internet = 0.0;

        printf("Tahun %d:\n", year);
        printf("Jumlah Penduduk: %.0f\n", pred_pop);
        printf("Persentase Pengguna Internet: %.4f%%\n\n", pred_internet);
    }

    return 0;
}
```