#include <iostream>

#include <string>

#include <chrono>

using namespace std;

using namespace std::chrono;

//Andrew Virya Victorio\_32200091

struct Patient {

    string name;

    string disease;

};

//Maria Yosephine\_32200027

//PRINT

void print (Patient arrPatients[], int data){

    for ( int i = 0; i < data; i++ ){

        cout << "Nama Pasien Setelah Diurutkan : " << arrPatients[i].name << endl;

    }

}

//Maria Yosephine\_32200027

//SEQUENTIAL SEARCH FUNCTION

Patient seqSearch ( Patient arrPatients[], int data, string name ){

    Patient patient;

    for ( int i = 0; i < data; i++ ){

        if ( arrPatients[i].name == name ){

            patient = arrPatients[i];

            cout << "Pasien ditemukan";

        }

        else if ( i == data - 1 ){

            cout << "Pasien tidak ditemukan, coba lagi!" << endl;

        }

    }

    return patient;

}

//Andrew Virya Victorio\_32200091

//BINARY SEARCH FUNCTION

Patient binSearch ( Patient arrPatients[], int data, string name ){

    Patient patient;

    int left = 0;

    int right = data - 1;

    int mid;

    bool found = false;

    while ( left <= right ){

        mid = (left + right) / 2;

        if ( name == arrPatients[mid].name ){

            found = true;

            patient = arrPatients[mid];

            break;

        }

        else if ( name < arrPatients[mid].name ){

            right = mid - 1;

        }

        else {

            left = mid + 1;

        }

    }

    if ( found == true ){

        cout << "Data Pasien Ditemukan" << endl;

    }

    else {

        cout << "Pasien tidak ditemukan, coba lagi!" << endl;

    }

    return patient;

}

//Maria Yosephine\_32200027

//PATIENT DATA

Patient patientData (){

    Patient patient;

    cout << "Masukkan nama pasien : ";

    getline ( cin, patient.name );

    cout << "Masukkan penyakit pasien : ";

    getline ( cin, patient.disease );

    return patient;

}

//Andrew Virya Victorio\_32200091

//MAIN PROGRAM

int main (){

    int data;

    int i;

    string name;

    Patient arrPatients[10] = {};

    cout << "Searching Program!!\n";

    cout << "===================================\n";

    cout << "Masukkan banyak data : ";

    cin >> data;

    cout << "\n";

    cin.ignore();

    for ( int i = 0; i < data; i++ ){

        Patient patient;

        cout << "Data Pasien Ke-" << i+1 << endl;

        patient = patientData();

        arrPatients[i] = patient;

        cout << "===================================\n";

    }

    //SORTING

    for ( int i = 0; i < data; i++ ){

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

            if ( arrPatients[j].name > arrPatients[i].name ){

                Patient temp = arrPatients[j];

                arrPatients[j] = arrPatients[i];

                arrPatients[i] = temp;

            }

        }

    }

    print(arrPatients, data);

    cout << "Masukkan nama yang ingin dicari : ";

    getline (cin, name);

    auto begin = steady\_clock::now();

    //NB: Kami menggunakan fungsi searching

    //    sehingga tidak diperlukan 2 code

    //    berbeda. Cukup matikan atau nyalakan

    //    commentnya saja untuk menggunakan

    //    fungsinya.

    //Patient patient = seqSearch(arrPatients, data, name);

    Patient patient = binSearch(arrPatients, data, name);

    auto stop = steady\_clock::now();

    auto time = begin - stop;

    double elapsedTime = double( duration\_cast <microseconds> (time).count() );

    cout << "Elapsed time in Microseconds: " << elapsedTime << endl;

    return 0;

}

//ALGORITHM

/\*

#include <iostream>

#include <string>

#include <chrono>

using namespace std;

using namespace std::chrono;

//Andrew Virya Victorio\_32200091

Patient : struct {

    name : string

    disease : string

}

//Maria Yosephine\_32200027

//PRINT

print : void (arrPatients[] : Patient, data : integer){

    for <-- integer i = 0; i to data; do

        write (Nama Pasien Setelah Diurutkan : )

}

//Maria Yosephine\_32200027

//SEQUENTIAL SEARCH FUNCTION

seqSearch : Patient ( arrPatients[] : Patient, data : int, name : string ){

    patient : Patient

    for <-- int i = 0; i to data; do

        if ( arrPatients[i].name == name ){

            patient = arrPatients[i];

            write (Pasien ditemukan)

        }

        else if ( i == data - 1 ){

            write (Pasien tidak ditemukan, coba lagi!)

        }

    return patient

}

//Andrew Virya Victorio\_32200091

//BINARY SEARCH FUNCTION

binSearch : Patient ( arrPatients[] : Patient, data : int, name : string ){

    patient : Patient

    left = 0 : integer

    right = data - 1 : integer

    mid : integer

    found = false : bool

    while ( left <= right ){

        mid = (left + right) / 2;

        if ( name == arrPatients[mid].name ){

            found = true;

            patient = arrPatients[mid];

            break;

        }

        else if ( name < arrPatients[mid].name ){

            right = mid - 1;

        }

        else {

            left = mid + 1;

        }

    }

    if ( found == true ){

        write (Data Pasien Ditemukan)

    }

    else {

        write (Pasien tidak ditemukan, coba lagi!)

    }

    return patient

}

//Maria Yosephine\_32200027

//PATIENT DATA

patientData : Patient (){

    patient : Patient

    write (Masukkan nama pasien : )

    getline ( read, patient.name )

    write (Masukkan penyakit pasien : )

    getline ( read, patient.disease )

    return patient

}

//Andrew Virya Victorio\_32200091

//MAIN PROGRAM

int main (){

    data : integer

    i: integer

    name : string

    arrPatients[10] = {} : Patient

    write (Searching Program!!)

    write (===================================)

    write (Masukkan banyak data : )

    read : data

    read.ignore()

    for <-- int i = 0; i to data; do

        patient : Patient

        write (Data Pasien Ke-)

        patient = patientData();

        arrPatients[i] = patient;

        write (===================================)

    //SORTING

    for <-- int i = 0; i to data; do

        for <-- int j = 0; j to i; do

            if ( arrPatients[j].name > arrPatients[i].name ){

                Patient temp = arrPatients[j];

                arrPatients[j] = arrPatients[i];

                arrPatients[i] = temp;

            }

    print(arrPatients, data);

    write (Masukkan nama yang ingin dicari : )

    getline (read, name)

    auto begin = steady\_clock::now();

    //NB: Kami menggunakan fungsi searching

    //    sehingga tidak diperlukan 2 code

    //    berbeda. Cukup matikan atau nyalakan

    //    commentnya saja untuk menggunakan

    //    fungsinya.

    //Patient patient = seqSearch(arrPatients, data, name)

    Patient patient = binSearch(arrPatients, data, name)

    auto stop = steady\_clock::now()

    auto time = begin - stop

    double elapsedTime = double( duration\_cast <microseconds> (time).count() )

    write (Elapsed time in Microseconds: )

    return 0;

}

\*/

/\*

- HIPOTESA :

sequential search secara data, lebih cepat dibandingkan binary search. alasannya

dikarenakan, sequential search mencari data secara berurut/linier. sedangkan,

binary search, mencari data secara bagi dua, yang mana berdampak pada kecepatan

pencarian data. Kalau, data yang diinput sangat banyak, misalkan > 100 data, menggunakan

binary search bisa lebih cepat. Alasannya dikarenakan dia membagi dua, kalau data berada

di sebelah kanan, sequential search harus mencari data dari kiri ke kanan, yang mana mem-

butuhkan waktu lebih lama. Kalau data yang digunakan sedikit, menggunakan sequential search

lebih cepat.

- variable test =

1. penginputan data sebanyak 3 data

2. percobaan dilakukan sebanyak 4x

3. nama dan penyakit diinput sama semuanya selama 4x berturut-turut

4. percobaan dilakukan di sequential maupun binary dengan variable percobaan yang sama

5. percobaan dilakukan dengan menggunakan library chrono untuk mengukur waktu

- hasil percobaan :

seqSearch = elapsed time nya ada di kisaran rata-rata 1000 microseconds

binSearch = elapsed time nya ada di kisaran rata-rata 1000-2000 microseconds

- Kesimpulan percobaan :

Di dalam percobaan kali ini, teknik sequential search lebih cepat dibandingkan

binary search. Dikarenakan data yang dicari sedikit.

\*/

/\*

masukkan banyak data = done

user input banyak data = done

looping nama pasien = done

looping penyakit pasien = done

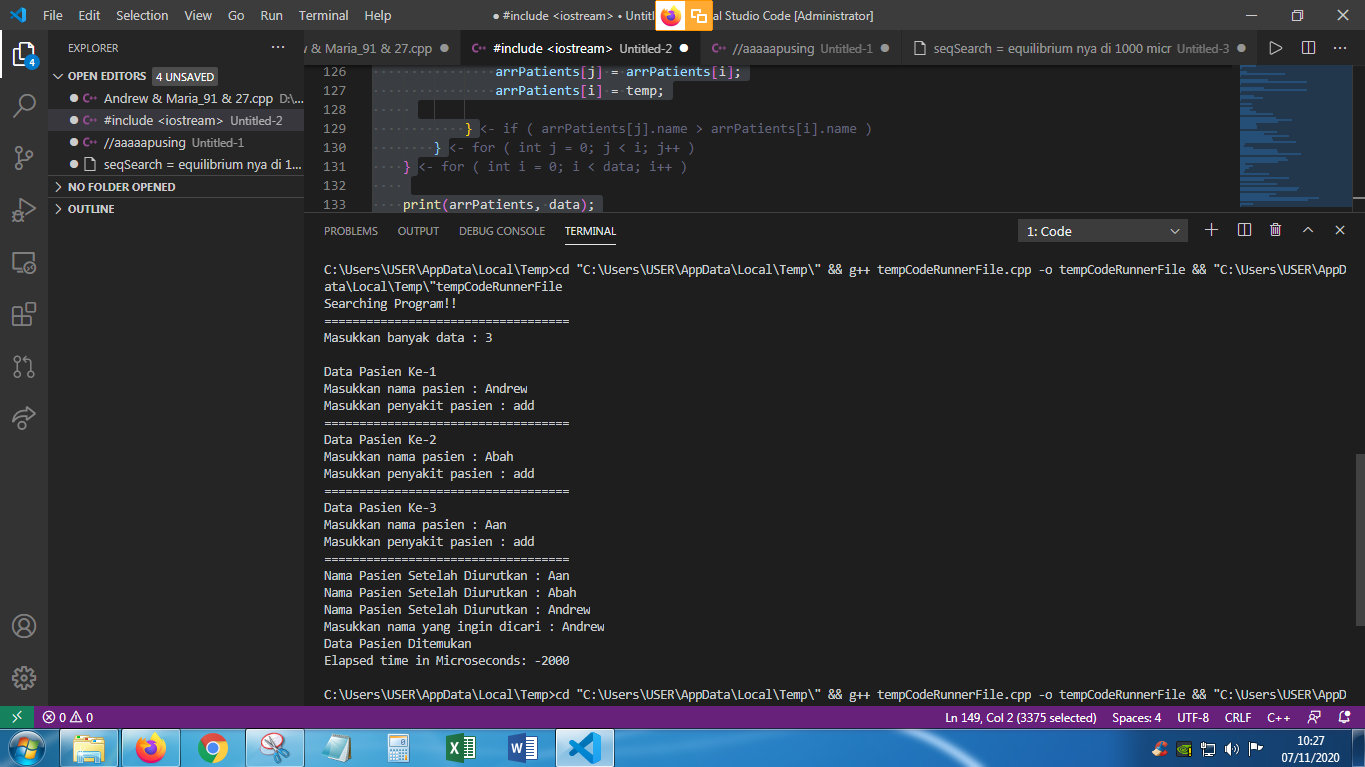
sorting nama pasien = done

searching nama Pasien = done

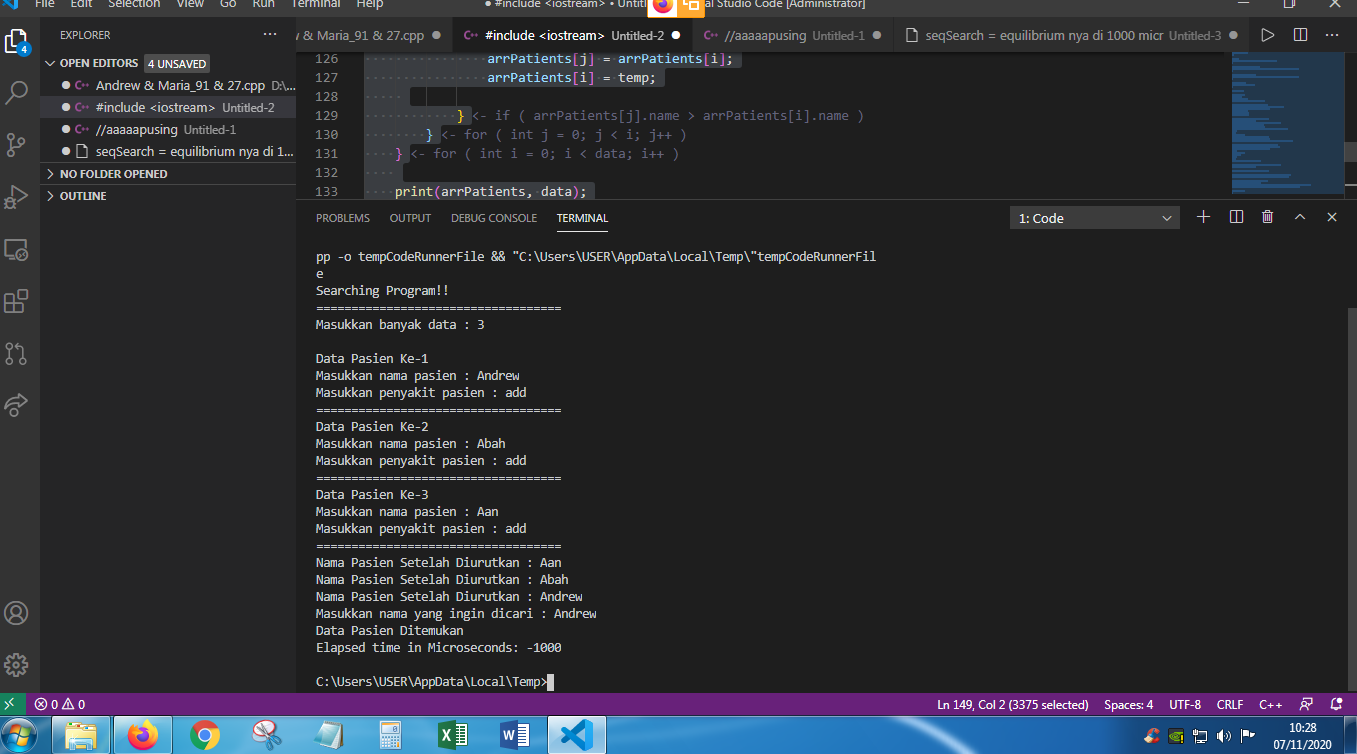
tampilin penyakit pasien yang di search ("nama" terkena penyakit "penyakit", semoga lekas sembuh)

tampilin index ke berapa

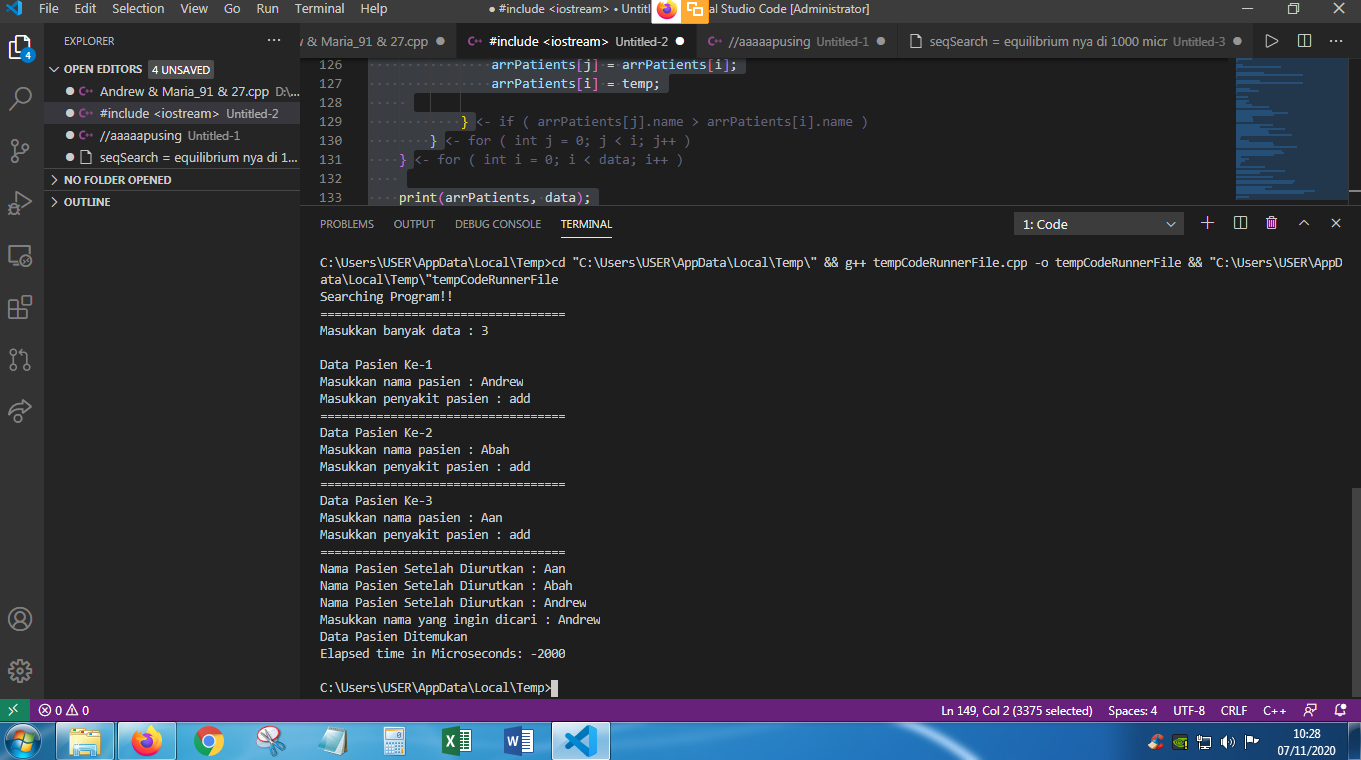
\*/



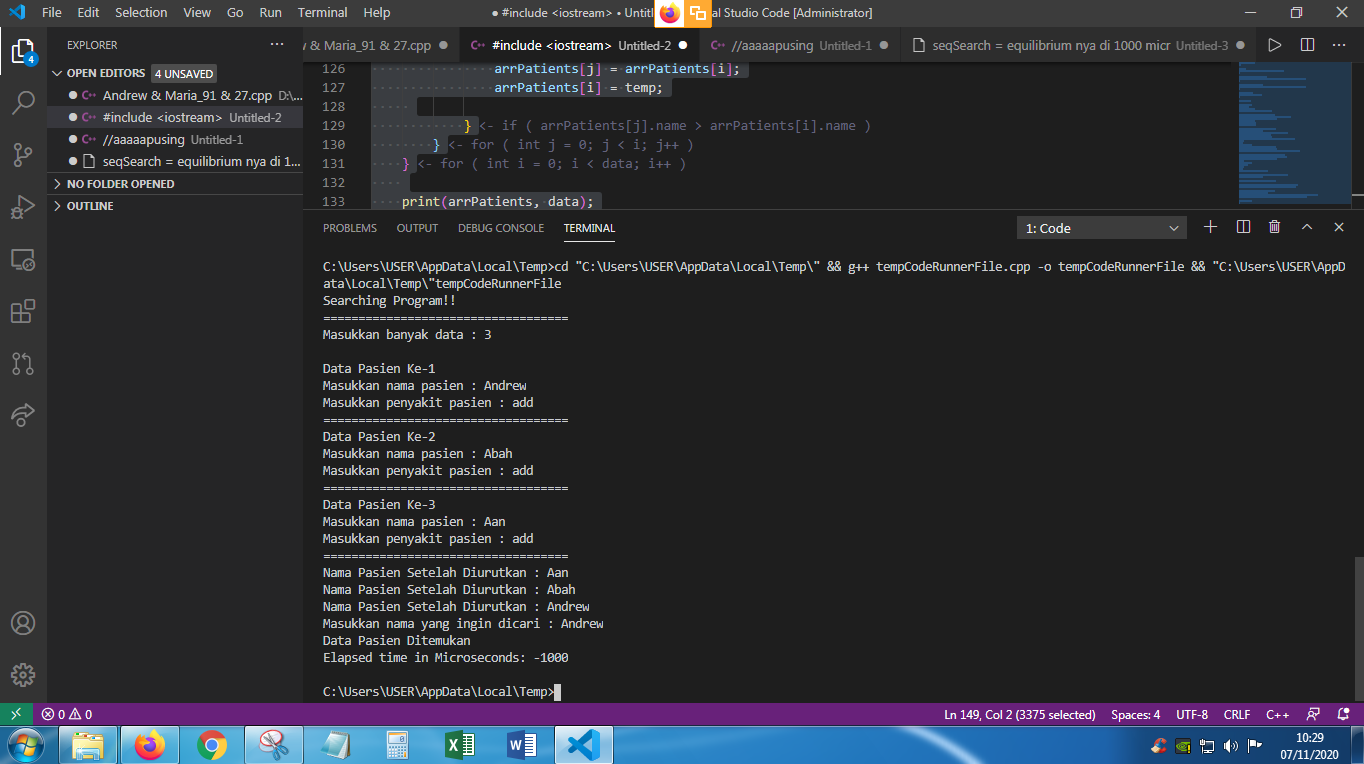
Gambar .1 : Test 1 Binary



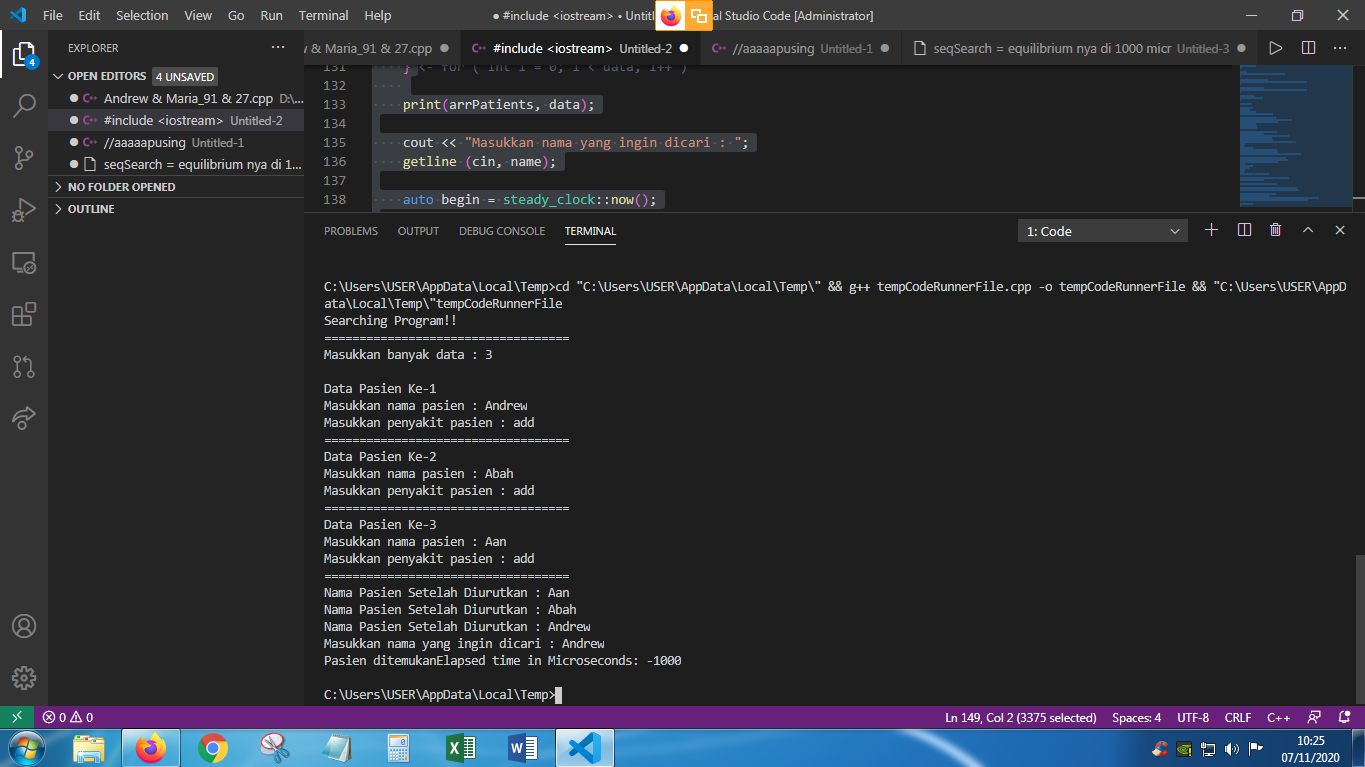
Gambar . Test 2 Binary



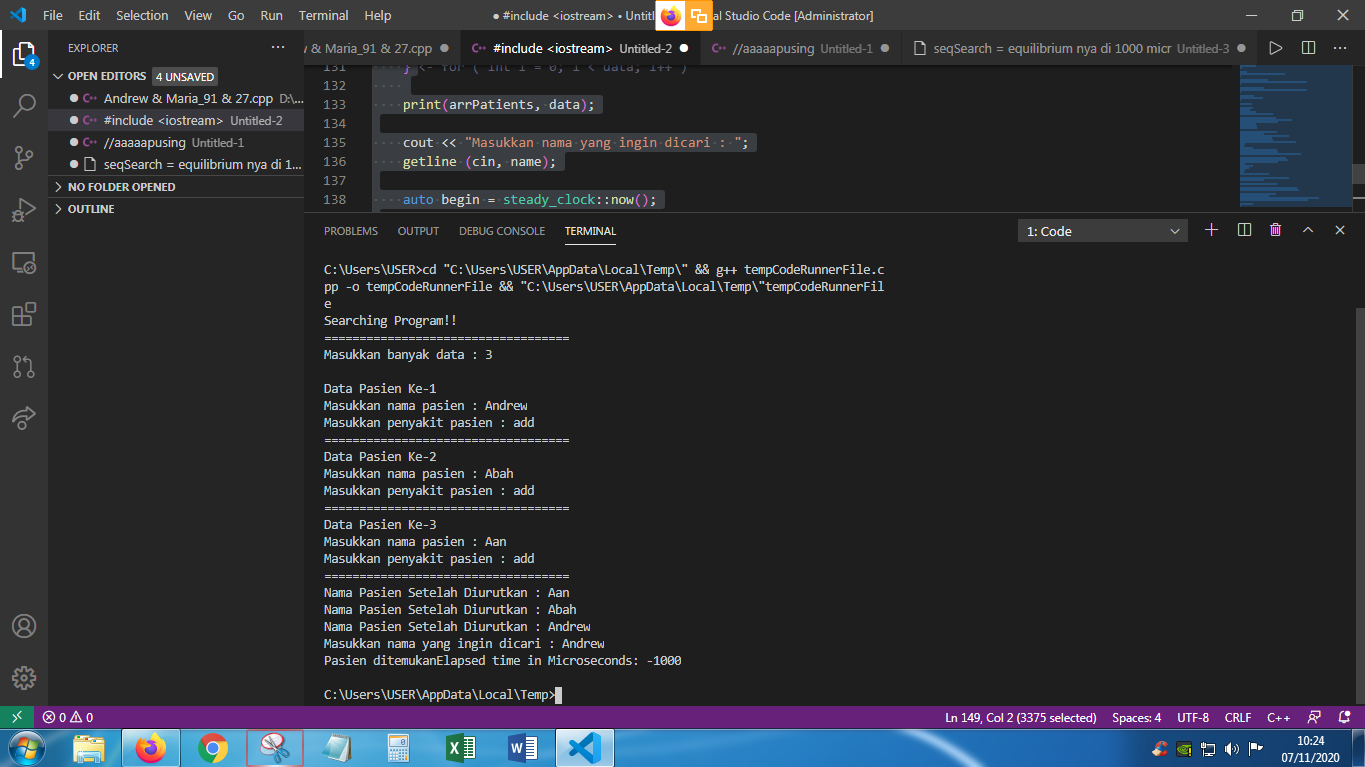
Gambar . Test 3 Binary



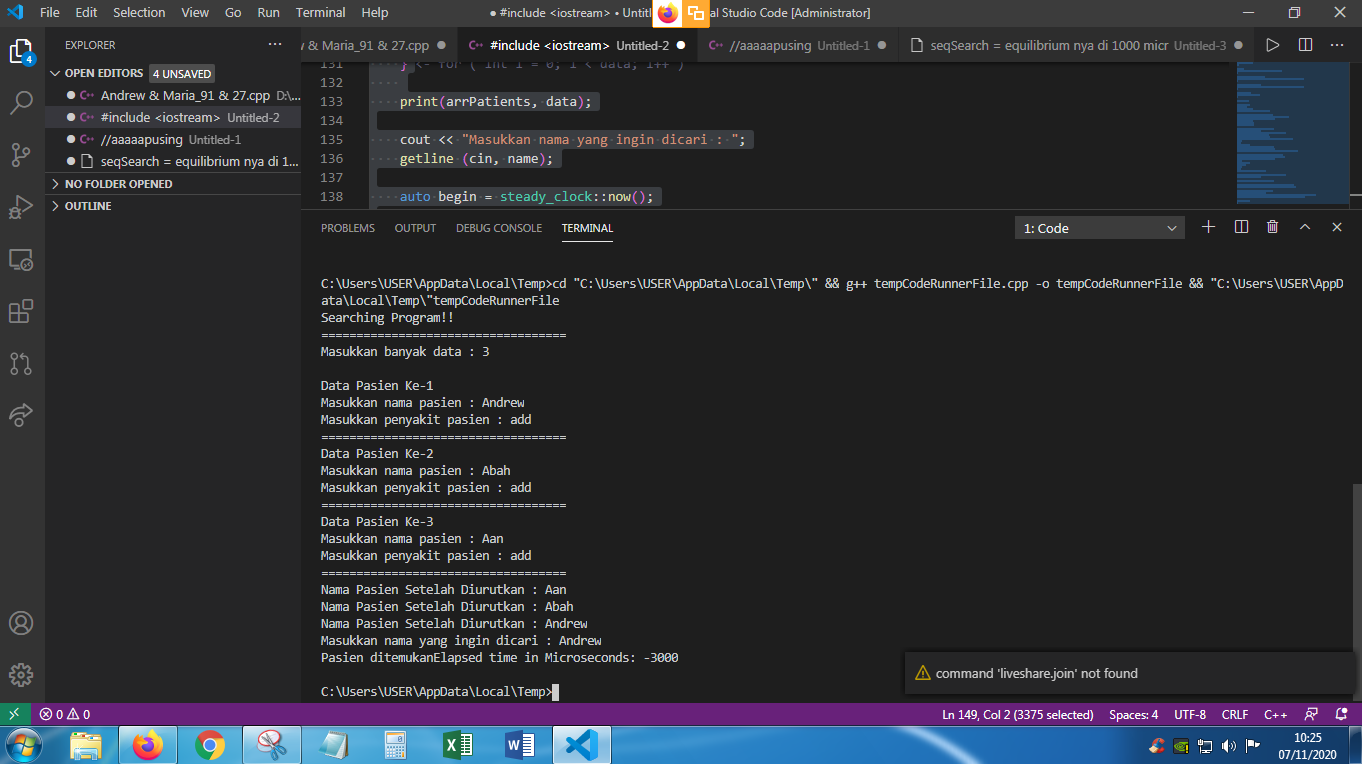
Gambar . Test 4 Binary



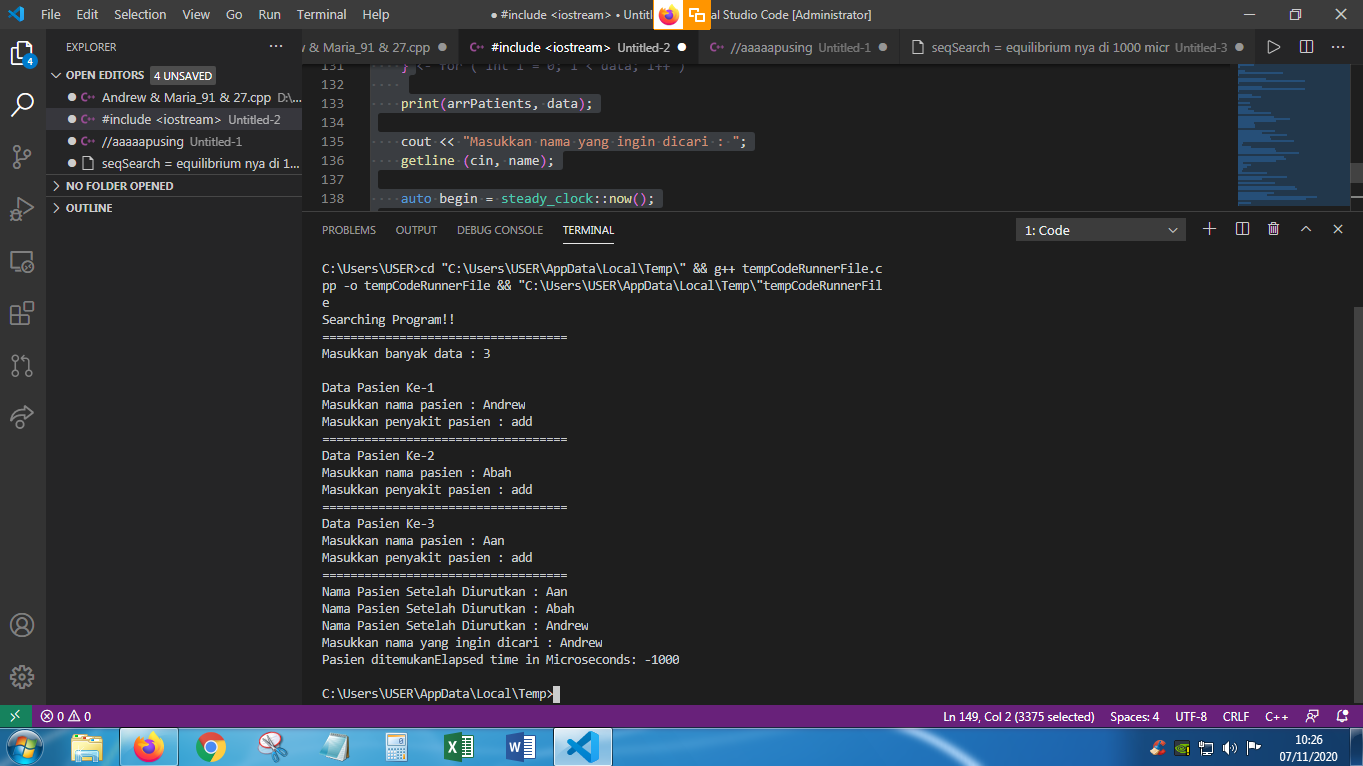
Gambar . Test 1 Sequential



Gambar . Test 2 Sequential



Gambar . Test 3 Sequential



Gambar.. Test 4 Sequential

