Assignment – 8

1. Membalik Bilangan

#include <iostream>

using namespace std;

// deklarasi fungsi prototype reverse

*string* reverse(*string* *n*);

// definisi fungsi main

int main() {

    // deklarasi variabel

*string* n;

    // input bilangan yang akan ditukar

    cin >> n;

    // pemanggilan fungsi reverse

    // print hasil penukaran bilangan

    cout << reverse(n);

    return 0;

}

// definisi fungsi reverse

*string* reverse(*string* *n*) {

    // batas rekursi, jika panjang string kurang dari / sama dengan 1, kembalikan nilai string tersebut ke fungsi

    if(*n*.length() <= 1) return *n*;

    // batas rekursi, jika panjang string lebih dari 1, kembalikan nilai karakter terakhir dari string input ditambah dengan rekursi fungsi reverse dari nilai string pada index pertama hingga index terakhir dikurangi satu

    // perulangan akan terus dijalankan hingga panjang string = 1

    // substr(x, y) adalah fungsi untuk mengambil sebagian elemen dari sebuah string, dimulai dari index ke-x sampai index ke-y

    else return *n*.substr(*n*.length() - 1) + reverse(*n*.substr(0, *n*.length() - 1));

}

2. Koefisien Binomial

#include <iostream>

using namespace std;

// deklarasi fungsi prototype binomial

int binomial(int *n*, int *k*);

// definisi fungsi main

int main() {

    // deklarasi variabel

    int n, k;

    // input variabel indeks

    cin >> n >> k;

    // pemanggilan fungsi binomial

    // print nilai indeks

    cout << binomial(n, k);

    return 0;

}

// definisi fungsi binomial

int binomial(int *n*, int *k*) {

    // batas rekursi, jika nilai n = k, n = 1, atau k = 0, kembalikan nilai 1 ke fungsi

    // karena setiap indeks dengan nilai n yang sama dengan k, n = 1 atau k = 0 pasti memiliki nilai 1

    if(n == k || n == 1 || k == 0) return 1;

    /\* rekursi fungsi binomial

     - nilai dari indeks yang dimasukkan user adalah jumlah dari dua angka yang berada di indeks atasnya pada segitiga pascal

     - pola tersebut dilanjutkan terus menerus hingga mencapai batasnya, yaitu di puncak segitiga pascal seperti kondisi pada if di atas

     - sehingga rumus yang digunakan untuk mencari nilai dari indeks adalah rekursi dari fungsi binomial dengan pengurangan nilai n dan k sebagai berikut \*/

    else return binomial(n - 1, k - 1) + binomial(n - 1, k);

}

3. Fungsi Ackermann

#include <iostream>

using namespace std;

// deklarasi fungsi prototype a (ackermann)

int a(int *m*, int *n*);

// definisi fungsi main

int main() {

    // deklarasi variabel

    int m, n;

    // input bilangan parameter

    cin >> m >> n;

    // pemanggilan fungsi a

    // print nilai fungsi

    cout << a(m, n);

    return 0;

}

// deklarasi fungsi prototype a (ackermann)

int a(int *m*, int *n*) {

    // batas rekursi, jika nilai m = 0, kembalikan nilai n + 1 ke fungsi (sesuai dengan aturan dari fungsi ackermann)

    if(*m* == 0) return *n* + 1;

    // batas rekursi, jika nilai n = 0, kembalikan nilai rekursi fungsi a di mana m dikurangi 1

    else if(*n* == 0) return a(*m* - 1, 1);

    // jika m dan n keduanya bukan 0, fungsi rekursi yang digunakan adalah sesuai dengan rumus fungsi ackermann, yaitu sebagai berikut

    else return a(*m* - 1, a(*m*, *n* - 1));

}

4. Palindrom

#include <iostream>

using namespace std;

// deklarasi fungsi prototype palindrome

*string* palindrome(*string* *a*);

// definisi fungsi main

int main() {

    // deklarasi variabel

*string* s;

    // input string

    cin >> s;

    // pemanggilan fungsi palindrome

    // print jenis string (palindrome/bukan)

    cout << palindrome(s);

    return 0;

}

// definisi fungsi palindrome

*string* palindrome(*string* *s*) {

    // jika string hanya terdiri dari 1 karakter, maka string tersebut termasuk palindrom

    if(*s*.length() <= 1) return "Palindrom";

    // jika string terdiri dari 2 karakter atau lebih

    else {

        // jika huruf pertama dari string sama dengan huruf terakhir dari string

        // kembalikan nilai string baru namun dimulai dari huruf kedua dari awal string dan diakhiri pada huruf kedua dari akhir string

        if(*s*[0] == *s*[*s*.length() - 1]) return palindrome(*s*.substr(1, *s*.length() - 2));

        // jika huruf pertama dan huruf terakhir tidak sama, maka string otomatis bukan palindrom

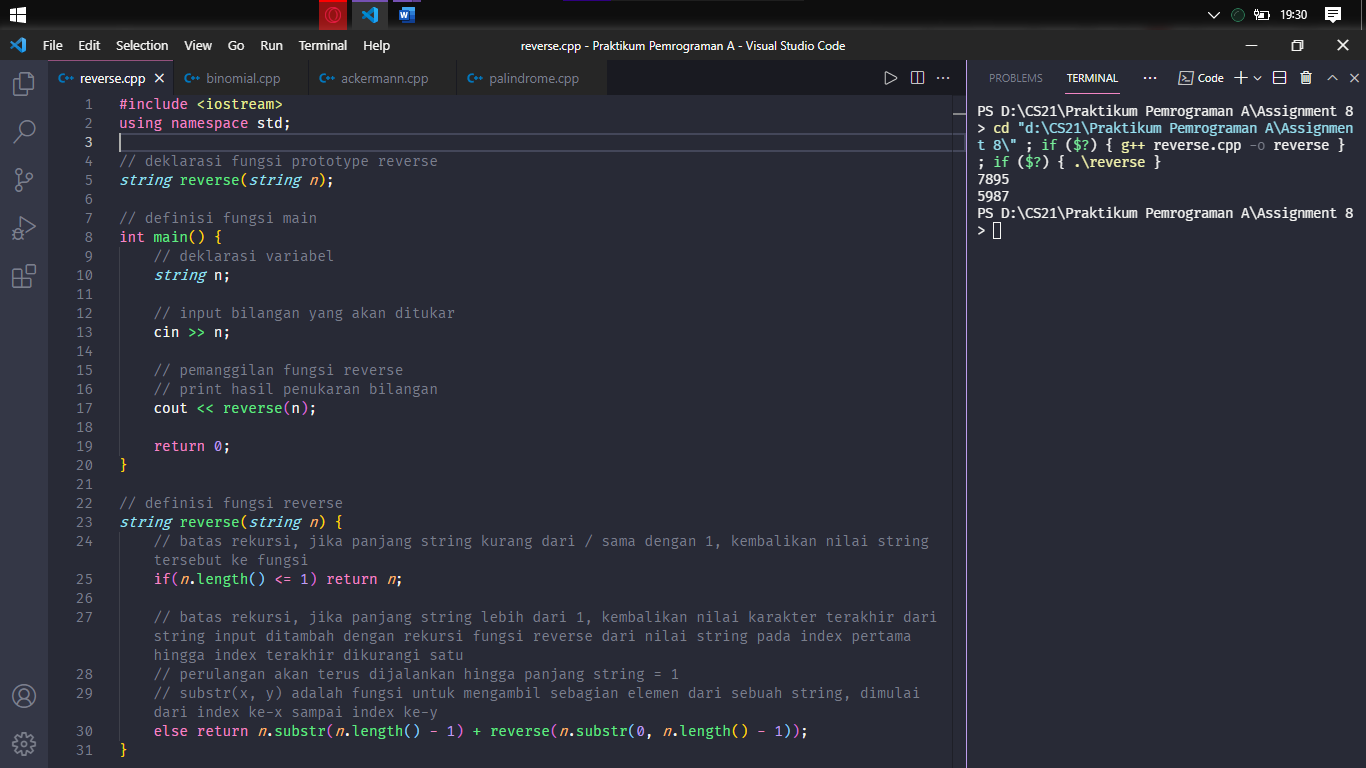
        else return "Bukan Palindrom";

    }

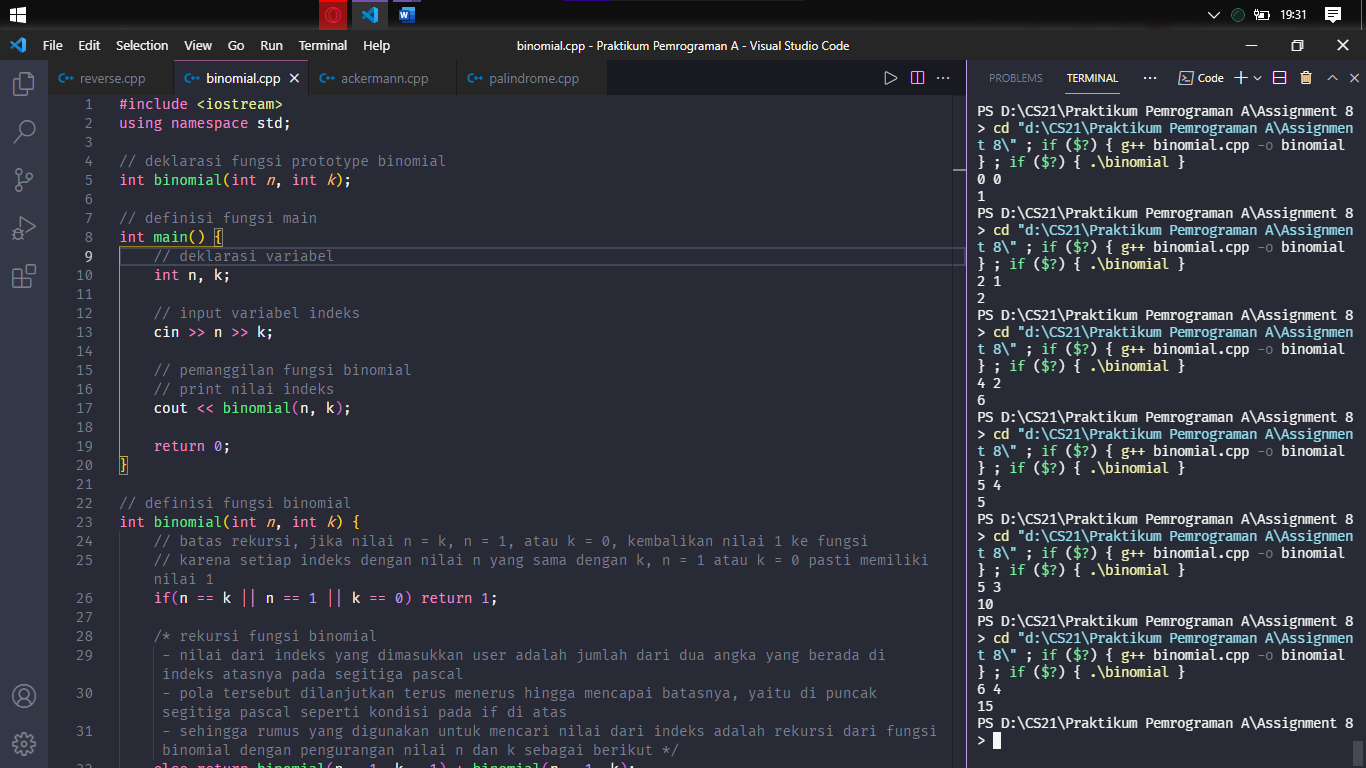
}

Screenshot

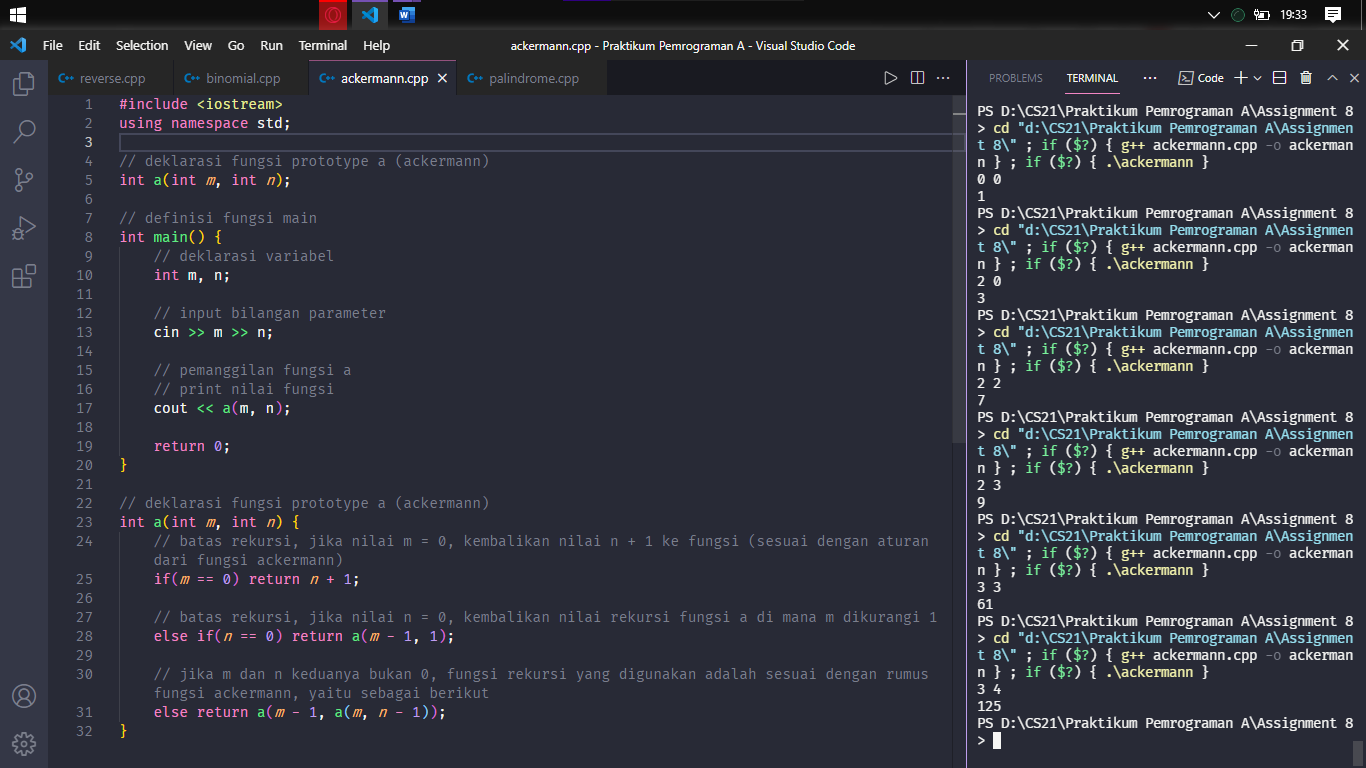
1. Membalik Bilangan



2. Koefisien Binomial



3. Fungsi Ackermann



4. Palindrom

