**LAPORAN TUGAS KECIL 2**

**MEMBUAT PUSTAKA UNTUK PERKALIAN POLINOM DENGAN ALGORITMA DIVIDE AND CONQUER**

**IF2211 Strategi Algoritma**

**­­**

Oleh:

**Daffa Pratama Putra 13518033**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2020**

# **BAB I STRATEGI ALGORITMA**

1. **Algoritma *Brute Force***

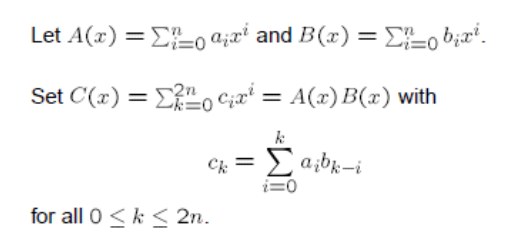
Algoritma *brute force* adalah salah satu algoritma yang digunakan untuk menyelesaikan persoalan komputasi. Pendekatan yang dilakukan oleh algoritma *brute force* adalah secara *straight forward* atau secara langsung. Biasanya algoritma ini digunakan untuk menyelesaikan persoalan yang sederhana, langsung, dan jelas persoalannya. Contoh penggunaan algoritma *brute force* adalah untuk mencari elemen terbesar/terkecil pada senarai, mencari elemen yang ada pada senarai, menghitung faktorial, dan lain sebagainya, termasuk untuk menyelesaikan persoalan perkalian polinomial.

Perkalian polinomial dapat diselesaikan dengan algoritma *brute force*. Algoritma *brute force* ini sama seperti mengalikan polinomial di kehidupan nyata, yaitu mengalikan semua koefisien polinom pertama dengan koefisien polinom kedua. Misalkan dua polinom berderajat n sebagai berikut :

A(x) = a0 + a1x + a2x2 + … + an-1xn-1 + anxn

B(x) = b0 + b1x + b2x2 + … + bn-1xn-1 + bnxn

maka secara umum perkalian kedua polinom tersebut adalah sebagai berikut:



sebagai contoh seperti di bawah ini :



Dengan menggunakan algoritma *brute force*, persoalan mengalikan dua polinom memiliki kompleksitas waktu sebesar O(n2). Kompleksitas waktu tersebut didapatkan karena pada algoritma *brute force* akan mengalikan sebanyak n derajat polinom terhadap n derajat polinom. Sehingga akan ada perkalian sebanyak n2, maka didapatkan kompleksitas waktu sebesar O(n2).

1. **Algoritma *Divide and Conquer***

Algoritma *divide and conquer* adalah salah satu strategi algoritma yang menyelesaikan persoalan dengan cara membagi persoalan yang besar menjadi persoalan yang lebih kecil, sehingga lebih mudah diselesaikan. Langkah yang dilakukan dalam algoritma *divide and conquer* ada 3 yaitu :

* *Divide* : membagi persoalan besar menjadi sub-persoalan yang lebih kecil
* *Conquer* : menyelesaikan sub-persoalan dengan konsep rekursif
* *Combine* : menggabungkan solusi masing-masing sub-persoalan, sehingga membentuk kembali solusi semula

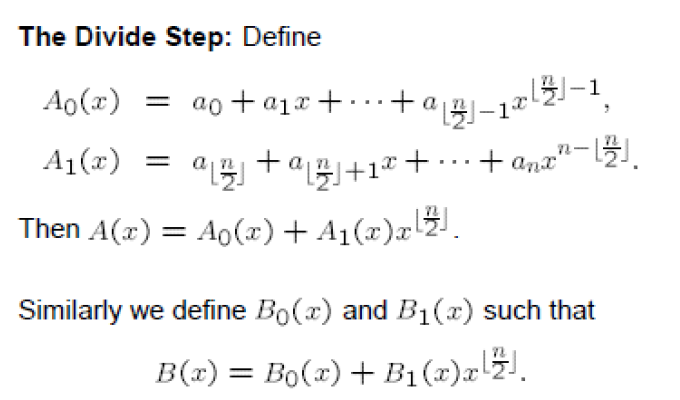
Biasanya objek yang dibagi menjadi sub-persoalan adalah berupa larik (*array*) berukuran n. Tiap sub-persoalan memiliki karakteristik yang sama dengan persoalan awal, sehingga sub-solusi dapat digunakan untuk memecahkan persoalan awal. Maka dari itu solusi dari algoritma *divide and conquer* dapat dicari dengan cara rekursif. Persoalan yang dapat diselesaikan oleh algoritma *divide and conquer* antara lain perpangkatan, mengurutkan larik (*quick sort, merge sort, insertion sort,* dan *selection sort*), dan perkalian polinom.

Penyelesaian persoalan perkalian polinom dengan algoritma *divide and conquer* mengadopsi algoritma Karatsuba. Algoritma Karatsuba dinilai lebih cepat dan efektif dalam menyelesaikan persoalan mengalikan dua polinom, yakni dengan kompleksitas waktu sebesar O() yang lebih cepat dibanding algoritma *brute force*. Pada algoritma *divide and conquer*, diasumsikan kedua polinom memiliki derajat polinom yang sama. Misalkan kedua polinom A dan polinom B sebagai berikut :

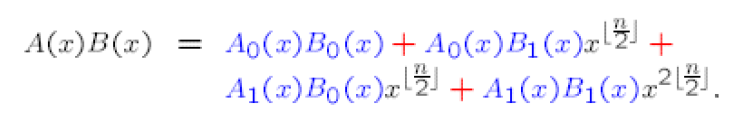
A(x) = a0 + a1x + a2x2 + … + an-1xn-1 + anxn

B(x) = b0 + b1x + b2x2 + … + bn-1xn-1 + bnxn

Maka bagi kedua polinom tersebut menjadi 4 bagian, misalkan A0­, A1, B0, dan B1. Pembagian polinom dilakukan tepat di tengah suku ke-n/2. Lebih jelasnya sebagai berikut :



Dengan begitu, kita dapat menulis perkalian polinom A terhadap polinom B sebagai



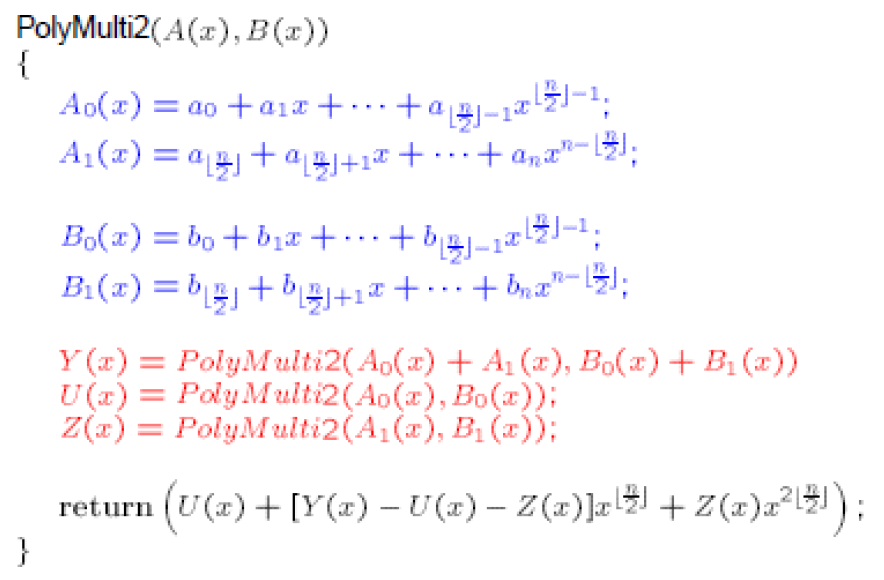
Dapat dilihat dari suku ke-2 dan ke-3 dapat digabung menjadi (A0B1 + A1B0), sehingga dengan melakukan manipulasi didapatkan

(A0B1 + A1B0) = (A0+A1)(B0+B1) – A0B0 – A1B1

sehingga dapat disusun keempat sub-polinom tersebut menjadi polinom U = A0B0, polinom Y = (A0+A1)(B0+B1), dan polinom Z = A1B1. Maka solusi perkalian polinom tersebut menjadi



Jika ditulis secara keseluruhan menjadi



# **BAB II KODE PROGRAM**

Kode program yang dibuat menggunakan bahasa C++ dengan memanfaatkan fitur kelas. Berikut adalah kode program untuk pustaka algoritma *Bruteforce* (dari kode program file header “Bruteforce.hpp”) :

|  |
| --- |
| #ifndef \_\_BRUTE\_\_FORCE\_\_HPP\_\_  #define \_\_BRUTE\_\_FORCE\_\_HPP\_\_  #include <iostream>  #include <time.h>  using namespace std;  class Polinom {      private :          int \*coef;          // Container koefisien polinom          int degree;         // Pangkat tertinggi polinom          static int numOp;   // Counter operator      public  :          // Constructor          Polinom();          Polinom(int degree);          // Copy constructor          Polinom(const Polinom& p);          // Destructor          ~Polinom();          // Operator assignment          Polinom& operator=(const Polinom& p);          // Getter setter          int getCoef(int idx) const;          int getDegree() const;          static int getNumOp();          void setCoef(int idx, int val);          void setDegree(int degree);          // Other operator          Polinom operator\*(const Polinom& A);          // Other method          void print();          void inputCoef();          void inputRandom();  };  #endif |

Berikut adalah kode program untuk pustaka algoritma *Divide and Conquer* (dari kode program file header “DividenConquer.hpp”) :

|  |
| --- |
| #ifndef \_\_DIVIDE\_\_AND\_\_CONQUER\_\_HPP\_\_  #define \_\_DIVIDE\_\_AND\_\_CONQUER\_\_HPP\_\_  #include <iostream>  #include <time.h>  using namespace std;  class Polinom {      private :          int \*coef;          // Container koefisien polinom          int degree;         // Pangkat tertinggi polinom          static int numOp;   // Counter operator      public  :          // Constructor          Polinom();          Polinom(int degree);          // Copy constructor          Polinom(const Polinom& p);          // Destructor          ~Polinom();          // Operator assignment          Polinom& operator=(const Polinom& p);          // Getter setter          int getCoef(int idx) const;          int getDegree() const;          static int getNumOp();          void setCoef(int idx, int val);          void setDegree(int degree);          // Other operator          Polinom operator+(const Polinom& A);          Polinom operator-(const Polinom& A);          Polinom operator\*(const Polinom& A);          // Other method          void print();          void inputCoef();          void inputRandom();          Polinom multiplyX(int d);  };  #endif |

Dari kode program di atas, ada beberapa *method* dan *attribute* yang dimiliki oleh kedua pustaka. *Attribute* yang dimiliki oleh kelas Polinom adalah coef, degree, dan numOp. Coef berfungsi sebagai tempat penampungan koefisien polinom yang berbentuk *array* dinamis, degree merepresentasikan derajat dari polinom atau pangkat tertinggi, dan numOp sebagai variabel menghitung jumlah operator yang digunakan. Lalu pada kedua pustaka memiliki “empat sekawan”, yakni *constructor* yang berfungsi untuk membentuk objek dari kelas polinom, *copy constructor* yang berfungsi untuk membentuk objek dari objek yang sudah ada, *destructor* untuk “menghancurkan” objek ada, dan operator= atau operator assignment untuk menyalin dari objek lain. Selanjutnya ada kelompok *getter* dan *setter* yang berfungsi untuk mengembalikan koefisien dan derajat serta mengeset koeifisen dan derajat.

Pada kelompok *method* lain, ada print, inputCoef, dan inputRandom. Print berfungsi untuk mencetak polinom ke layar, inputCoef berfungsi untuk memasukkan koefisien dari pengguna, dan inputRandom yang berfungsi untuk memasukkan koefisien yang dibangkitkan secara acak. Operator perkalian (operator\*) pada kedua pustaka terlihat sama, yakni untuk mengalikan kedua polinom, namun pada implementasinya menggunakan pendekatan yang berbeda. Pada kode program “Bruteforce.cpp”, implementasi operator\* menggunakan pendekatan secara algoritma *brute force*, sedangkan pada kode program “DividenConquer.cpp”, operator\* diimplementasikan secara algoritma *Divide and Conquer* yang mengadopsi dari algoritma Karatsuba. Pada kode program “DividenConquer.cpp”, ada dua *method* tambahan, yaitu operator+ yang berfungsi untuk menjumlahkan dua polinom dan operator- yang berfungsi untuk mengurangkan dua polinom.

# **BAB III UJI COBA**

1. **Spesifikasi Komputer**

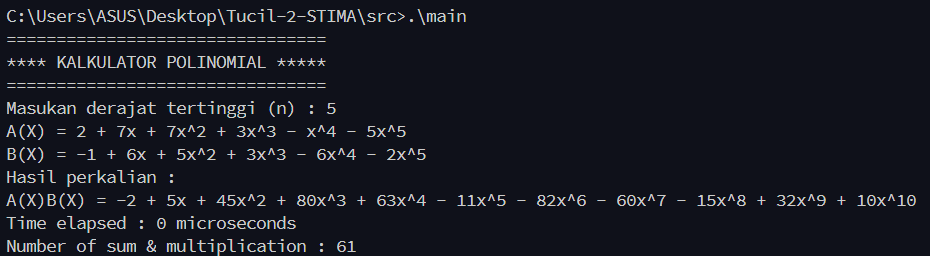
Berikut adalah spesifikasi komputer yang digunakan untuk melakukan uji coba.

* Nama : ASUS X550IK
* OS : Windows 10 64 bit
* CPU : QuadCore AMD Radeon FX-9830P 3.8GHz
* GPU : AMD Radeon R7 & AMD Radeon RX560
* RAM : 8GB

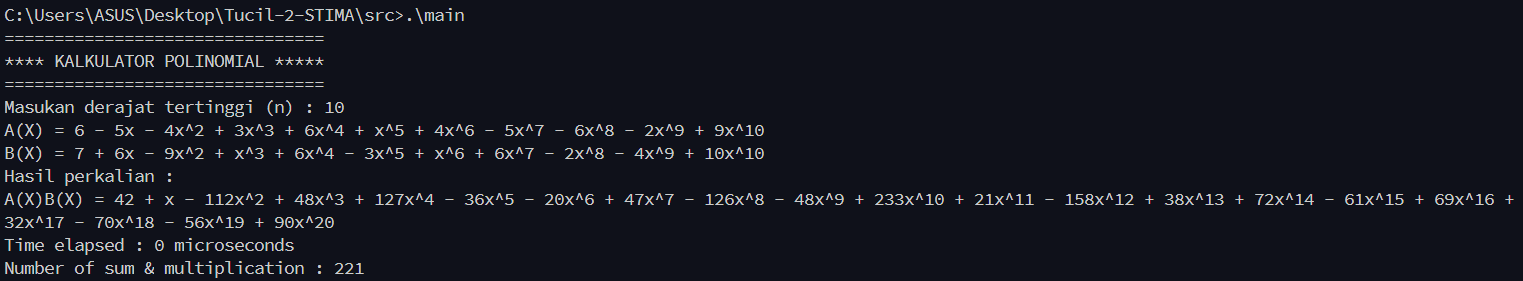
1. **Uji Coba dengan Algoritma *Brute Force***

Berikut adalah hasil uji coba dengan menggunakan algoritma *brute force* dengan membangkitkan koefisien polinom secara acak :

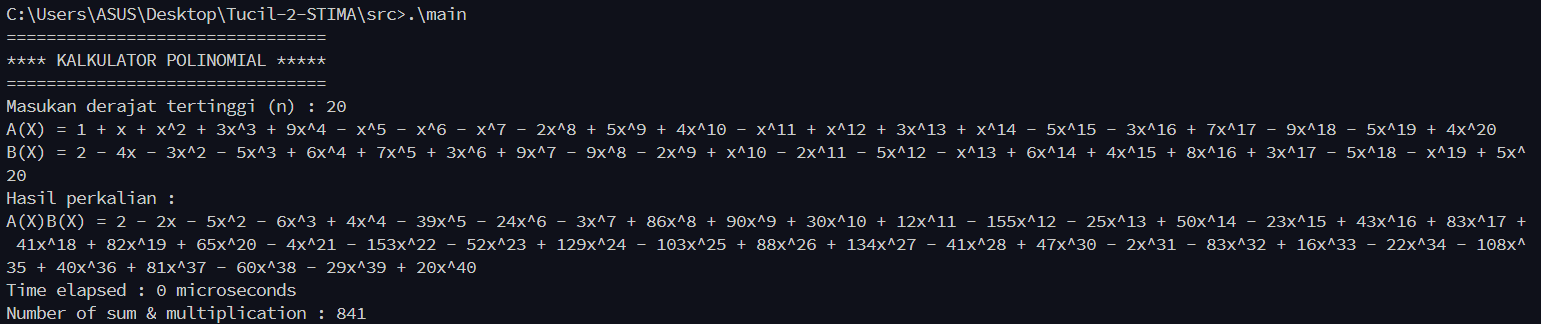
* + - 1. Untuk n = 5



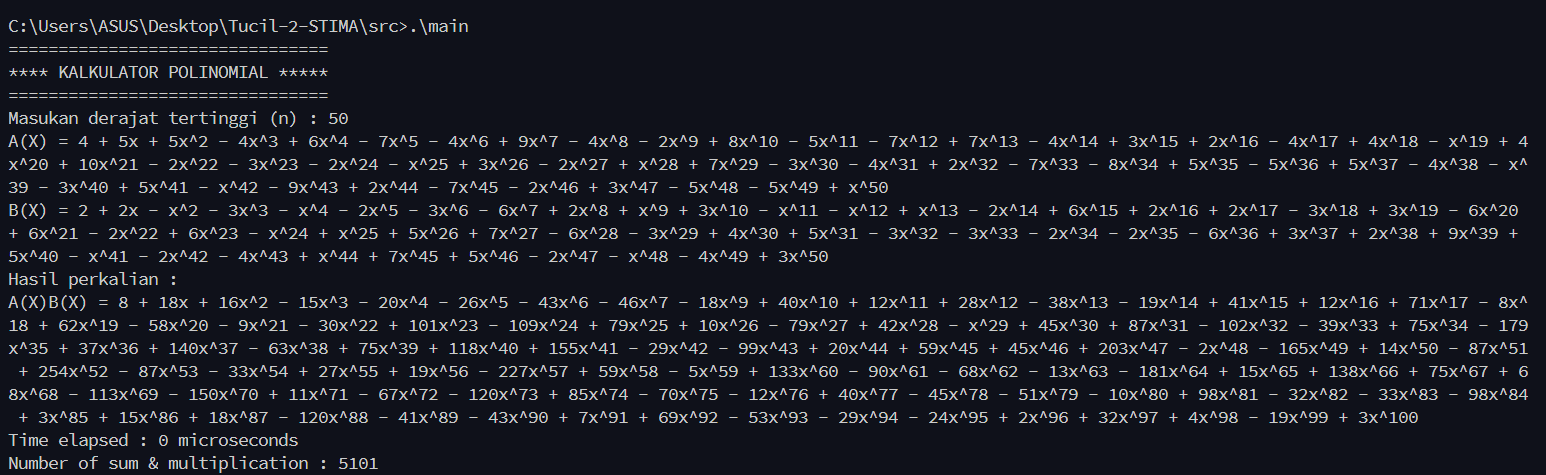
* + - 1. Untuk n = 10



* + - 1. Untuk n = 20



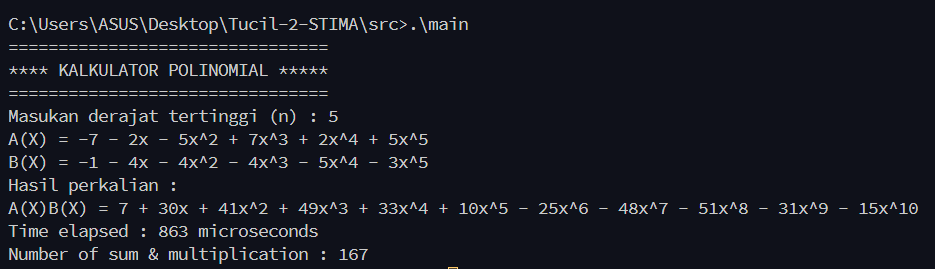
* + - 1. Untuk n = 50



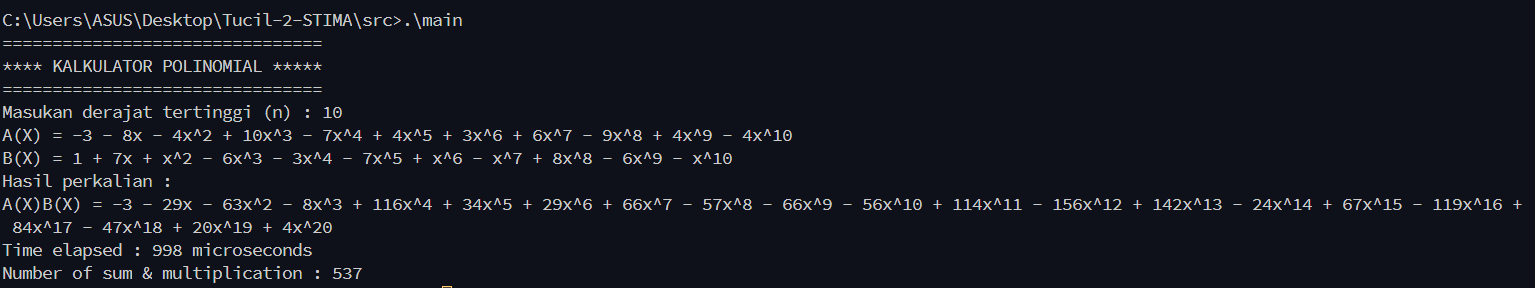
1. **Uji Coba dengan Algoritma *Divide and Conquer***

Berikut adalah hasil uji coba dengan menggunakan algoritma *divide and conquer* dengan membangkitkan koefisien polinom secara acak :

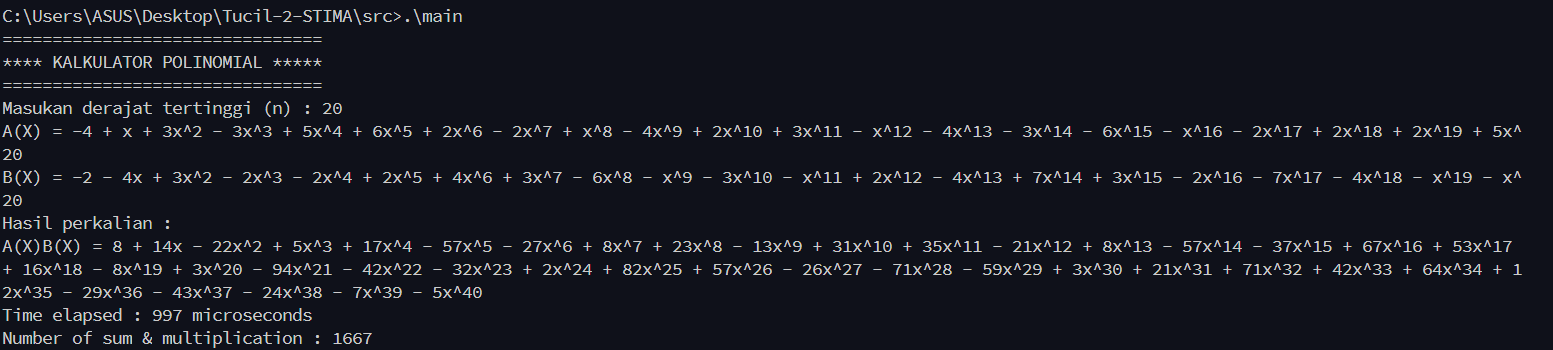
* + - 1. Untuk n = 5



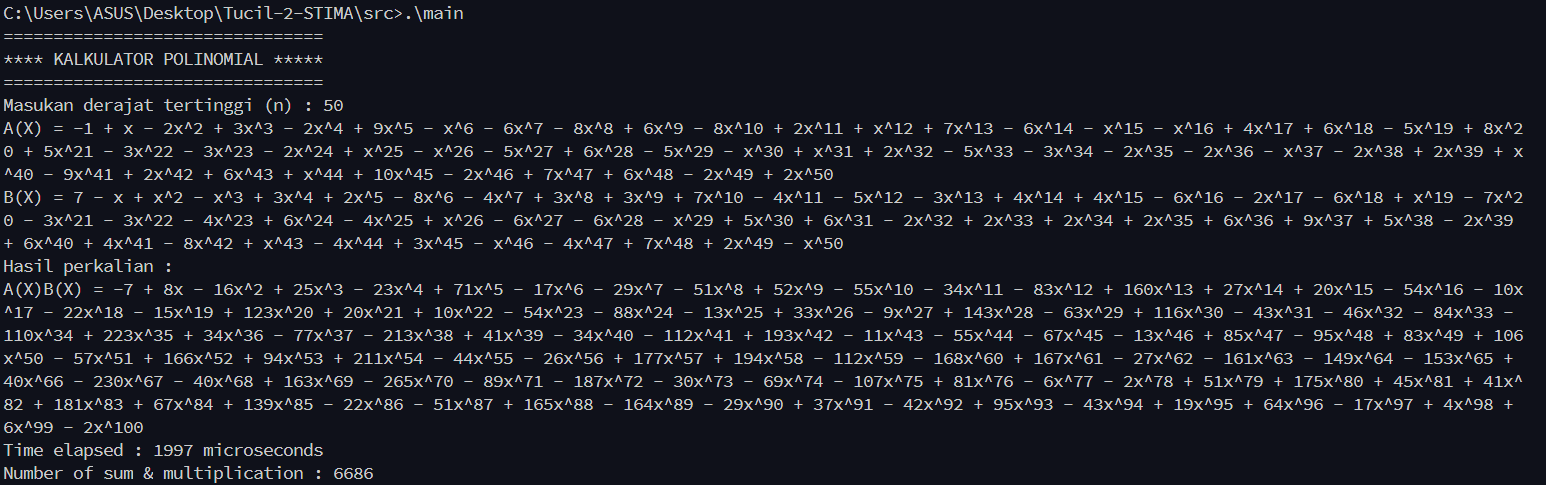
* + - 1. Untuk n = 10



* + - 1. Untuk n = 20



* + - 1. Untuk n = 50



Penilaian Asisten

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Program berhasil dikompilasi | V |  |
| 1. Program berhasil *running* | V |  |
| 1. Program dapat menerima input dan menuliskan output | V |  |
| 1. Luaran sudah benar untuk semua *n* | V |  |