JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

USULAN TUGAS AKHIR

1. IDENTITAS PENGUSUL

NAMA : **Yudha Nugraha**
NRP : **5110 100 062**
DOSEN WALI : **Dr.Ir. Siti Rochimah, MT.**
DOSEN PEMBIMBING : **1. Dwi Sunaryono, S.Kom., M.Kom.**
2. Rizky Januar Akbar, S.Kom., M.Eng.

2. JUDUL TUGAS AKHIR

"AutoGrader: Kakas Penilai Tugas Pemrograman dalam Bahasa C++ Menggunakan Metode *Graph Similarity*"

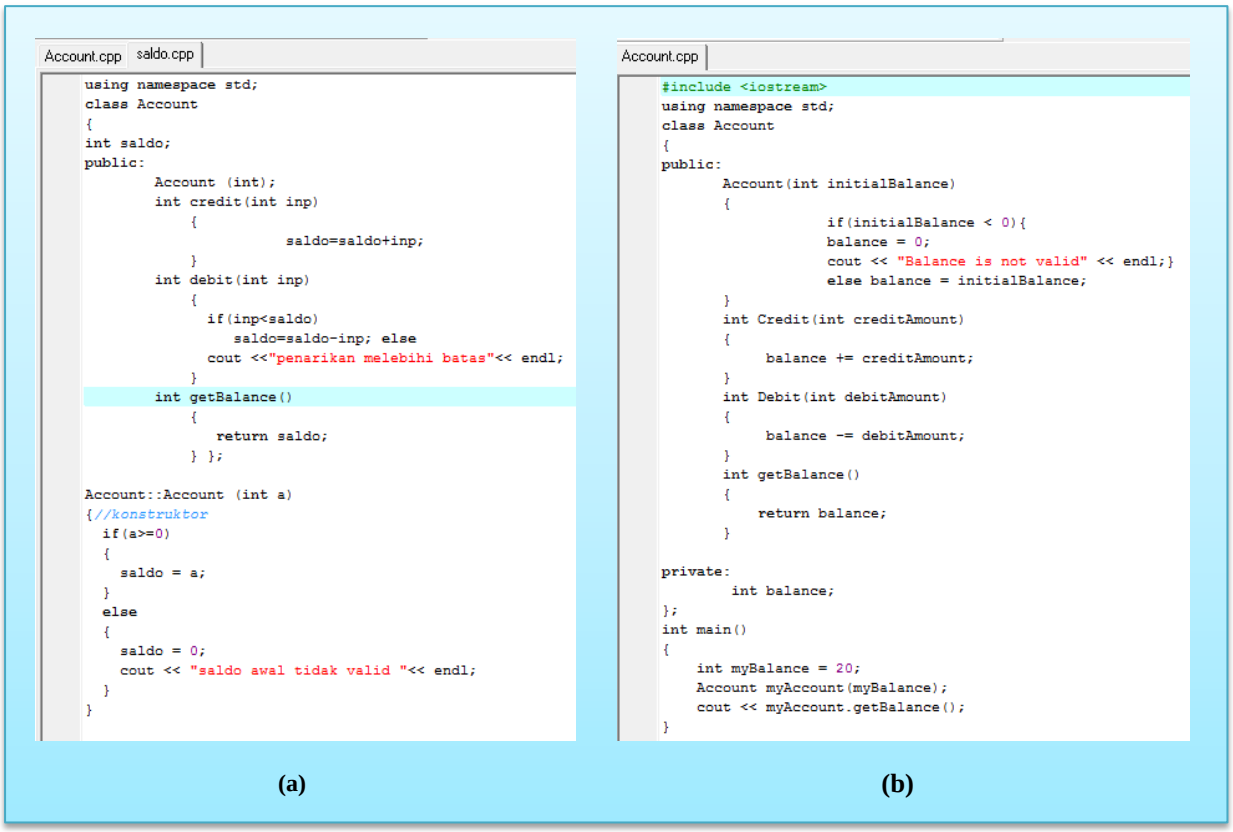
3. LATAR BELAKANG

Pemrograman berorientasi objek adalah salah satu mata kuliah wajib di jurusan Teknik Informatika. Mata kuliah ini mengharuskan banyak latihan untuk membantu mahasiswa menguasai materi. Bentuk latihan dapat berupa pemberian tugas oleh dosen pengajar yang kemudian akan diberikan *feedback* kepada mahasiswa agar dapat mempelajari kembali tugas tersebut jika terdapat kesalahan. Namun terkadang banyaknya jumlah mahasiswa menyebabkan dosen pengajar menghabiskan banyak waktu dalam mengoreksi setiap tugas yang terkumpul. Maka dibutuhkan sebuah kakas yang membantu dosen pengajar dalam mengoreksi tugas pemrograman. Berikut adalah contoh salah satu jenis soal yang biasa ditugaskan dalam mata kuliah pemrograman.

"Buatlah *class* Account untuk merepresentasikan rekening dari nasabah bank. Terdapat *attribute* berjenis *integer* yang merepresentasikan saldo rekening. Berikan sebuah *constructor* untuk saldo awal dan pastikan bernilai lebih besar atau sama

---

dengan dari 0. Jika tidak maka saldo bernilai 0 dan akan muncul pesan bahwa saldo awal tidak valid. Terdapat juga tiga *method*. *Credit* untuk menambah saldo. *Debit* untuk penarikan saldo dengan syarat tidak boleh melebihi saldo di rekening. *getBalance* untuk memberikan jumlah saldo ”[1].



Account.cpp | saldo.cpp

```
using namespace std;
class Account
{
int saldo;
public:
        Account (int);
        int credit(int inp)
            {
                        saldo=saldo+inp;
            }
        int debit(int inp)
            {
              if(inp<saldo)
                 saldo=saldo-inp; else
              cout <<"penarikan melebihi batas"<< endl;
            }
        int getBalance()
            {
               return saldo;
            } };

Account::Account (int a)
{//konstruktor
  if(a>=0)
  {
    saldo = a;
  }
  else
  {
    saldo = 0;
    cout << "saldo awal tidak valid "<< endl;
  }
}
```

(a)

Account.cpp

```
#include <iostream>
using namespace std;
class Account
{
public:
       Account(int initialBalance)
       {
                   if(initialBalance < 0){
                   balance = 0;
                   cout << "Balance is not valid" << endl;}
                   else balance = initialBalance;
       }
       int Credit(int creditAmount)
       {
            balance += creditAmount;
       }
       int Debit(int debitAmount)
       {
            balance -= debitAmount;
       }
       int getBalance()
       {
           return balance;
       }

private:
        int balance;
};
int main()
{
    int myBalance = 20;
    Account myAccount(myBalance);
    cout << myAccount.getBalance();
}
```

(b)

Gambar 1.(a). Kode Solusi Dosen. **(b).** Kode Solusi Mahasiswa.

Gambar 1 adalah dua kode solusi yang memiliki cara penyelesaian yang berbeda. Gambar 1.a adalah solusi dari dosen pengajar dan Gambar 1.b merupakan solusi mahasiswa. Selanjutnya dua kode tersebut akan dibandingkan. Dengan asumsi kode solusi dosen benar, berikut adalah beberapa perbedaan dari dua solusi di atas. Konstruktor pada kode solusi dosen memiliki parameter “a” dengan tipe *int.* Sedangkan pada kode solusi mahasiswa konstruktor memiliki parameter “initialBalance” dengan tipe *int.* Pada bagian ini kedua solusi memiliki nama parameter yang berbeda. Namun jika dilihat dari soal keduanya sama-sama benar. Maka pada kasus ini solusi dari mahasiswa tetap bernilai benar. Dengan kata lain selama tipe sama maka penamaan tidak menjadi masalah.

Sama halnya dengan *variable* “saldo” pada kode solusi dosen dan variable “balance” pada kode solusi mahasiswa. Selama *variable* bertipe *int* kasus ini masih dianggap benar. Sedangkan untuk *method* yang menjadi poin terpenting adalah jumlah yang harus sesuai dan tidak boleh kurang. *Parameter* yang dimiliki juga harus sesuai.

Begitupun dengan implementasi yang terdapat di dalamnya. Jika berupa percabangan maka akan diperiksa hingga ditemukan implementasi yang sesuai. Hal ini untuk menghindari jika pendeklarasian syarat di kasus percabangan terbalik seperti pada konstruktor kedua kode solusi di atas. Selain itu jika implementasi di suatu *method* tidak sesuai dengan soal maka akan mempengaruhi penilaian terhadap kode solusi mahasiswa.

Singkatnya kode solusi tersebut nantinya akan diuraikan menggunakan *Another Tool for Language Recognition* (ANTLR) yang akan menghasilkan *Abstract Syntax Tree* (AST). Menggunakan *graph similarity*, AST dari kode solusi mahasiswa dan kode solusi dari dosen pengajar akan dibandingkan untuk mendapatkan nilai.

4. RUMUSAN MASALAH

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut.

- Bagaimana membandingkan solusi tugas pemrograman dosen pengajar dengan solusi dari mahasiswa berdasarkan struktur program berorientasi objek (*class, property, method*).
- Bagaimana menentukan kriteria penilaian terhadap tugas pemrograman yang berorientasi objek.

5. BATASAN MASALAH

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, diantaranya sebagai berikut.

- *File input* adalah kode program menggunakan bahasa C++ dan berbentuk tunggal.
- Kakas hanya membandingkan struktur *class, property* dan *method*.
- Kakas dibangun menggunakan bahasa Java.

6. TUJUAN PEMBUATAN TUGAS AKHIR

Tujuan dari pembuatan tugas akhir ini adalah sebagai berikut.

- Membangun kakas untuk membandingkan kode solusi dosen dengan kode solusi mahasiswa berdasarkan struktur program berorientasi objek.
- Membuat sebuah kriteria penilaian terhadap tugas pemrograman menggunakan metode *graph similarity.*

7. MANFAAT TUGAS AKHIR

Manfaat dari tugas akhir ini adalah memberikan kemudahan bagi dosen pengajar pemrograman dalam menilai tugas pemrograman yang berorientasi objek.

8. TINJAUAN PUSTAKA

a. ANTLR

ANTLR adalah sebuah *parser generator* yang dapat digunakan untuk membaca, mengolah, menjalankan atau menerjemahkan teks terstruktur atau *file* biner. ANTLR mengambil suatu *input* (selanjutnya disebut *grammar*) sebagai *input* yang mengartikan sebuah bahasa dan menghasilkan *output* untuk mengenali bahasa tersebut. *Output* berupa program *Parser*, *Lexer* dan *Listener*. *Parser,* salah satu program yang merupakan *output* dari ANTLR dapat menghasilkan AST [2].

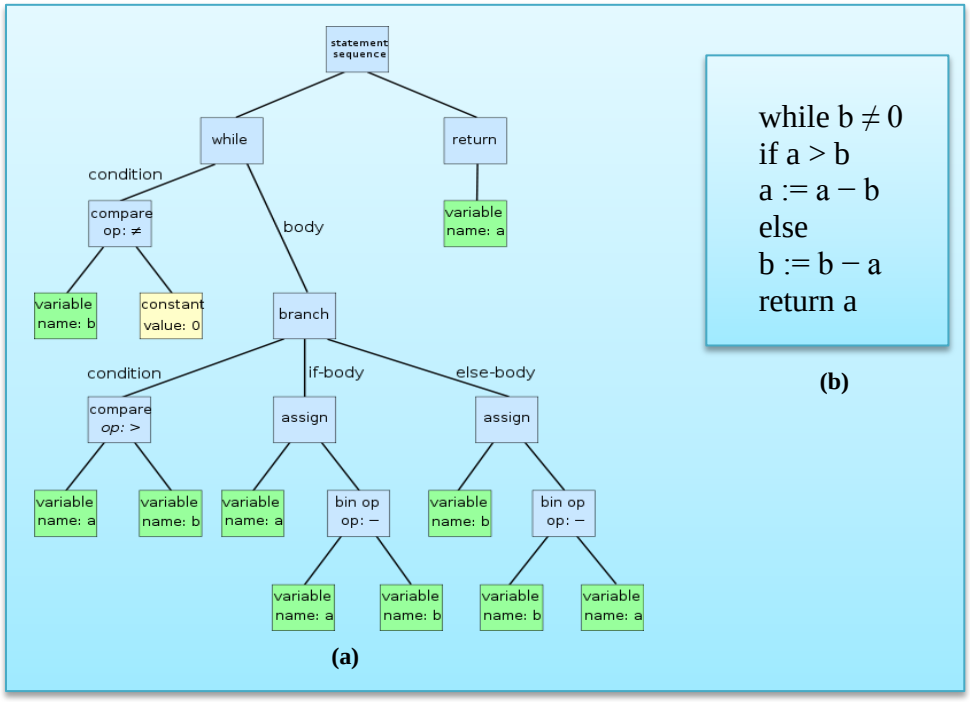
Grammar merupakan sebuah aturan yang digunakan ANTLR untuk mengartikan *input* yang ditulis menggunakan *grammar* tersebut. Secara sederhana *grammar* dapat dibuat, namun telah ada *grammar* untuk beberapa bahasa pemrograman seperti C#, C++ dan Java.

b. AST

Dalam ilmu komputer, AST atau biasa disebut *syntax tree,* adalah *tree* yang merepresentasikan aturan penulisan dari sebuah kode yang ditulis dalam suatu bahasa pemrograman. Setiap *node* dari *tree* mewakili sebuah *constract* yang ada di kode. Sintak disebut "*abstract*" adalah karena *tree* ini tidak merepresentasikan setiap detail yang muncul dalam kode. Misalnya aturan percabangan *if* yang digambarkan dalam sebuah *node* tunggal dengan dua cabang [3].

Hal ini membedakan AST dari *Concrete Syntax Tree* (CST) yang biasanya untuk menggambarkan *parse tree.* Diolah oleh sebuah *parser* bersamaan dengan proses penerjemahan dan kompilasi, CST biasa digunakan untuk analisa yang bersifat kontekstual.

AST merupakan sebuah struktur data yang digunakan dalam *compiler,* karena *tree* ini merepresentasikan struktur kode program. Sebuah AST biasanya merupakan hasil dari fase analisis sintaksis sebuah *compiler*. Gambar 2 adalah contoh representasi kode ke dalam sebuah AST. Gambar 2.b adalah kode *euclidean* dan Gambar 2.a adalah representasi bentuk AST.



Gambar 2.(a). AST. **(b).** Kode *euclidean.*

c. Graph Similarity

Graph similarity telah banyak digunakan dalam berbagai aplikasi seperti jejaring sosial, pengolahan gambar, dan *computer vision.* Terdapat banyak algoritma dan pendekatan kesamaan yang digunakan menerapkan metode ini. Cara yang diusulkan diklasifikasikan menjadi 3 kategori utama: *edit distance/graph isomorphism, feature extraction dan iterative methods*.

Edit Distance/Graph isomorphism

Salah satu pendekatan untuk mengevaluasi *graph similarity* adalah *graph isomorphism*. Dua *graph* dikatakan serupa jika keduanya isomorfik, atau salah satunya isomorfik dengan *subgraph* lainnya, atau keduanya memiliki *subgraph* yang isomorfik. Kelemahan metode ini adalah sifat eksponensial yang sangat tidak cocok diterapkan pada *graph* ukuran besar.

Edit distance adalah generalisasi dari kekurangan *graph isomorphism* yang tujuannya mengubah satu *graph* ke *graph* yang lain dengan melakukan sejumlah operasi (penambahan, pengurangan, penggantian *node* atau *edge*).

Feature Extraction

Kunci utama dari metode ini adalah bahwa *graph* yang mirip berbagi beberapa sifat seperti derajat distribusi dan diameter. Setelah mengekstrak fitur-fitur ini, ukuran kesamaan diterapkan untuk menilai kesamaan secara statistik, yang secara bersamaan juga menilai kesamaan *graph*. Metode ini dapat berjalan dengan baik karena memetakan *graph* ke dalam bentuk

statistik yang secara ukuran jauh lebih kecil dari pada *graph*. Namun masih mungkin ditemukan kekurangan seperti didapatkannya kesamaan yang tinggi antar dua *graph* yang memiliki perbedaan berarti dalam jumlah *node*.

Iterative methods
Prinsip dari *iterative method* adalah "dua *node* adalah mirip jika tetangga mereka juga mirip". Dalam setiap iterasi, *node* saling bertukar nilai kemiripan dan berhenti ketika konvergensi tercapai [4].

9. RINGKASAN ISI TUGAS AKHIR

Banyaknya latihan yang diperlukan untuk mencapai kompetensi dalam mata kuliah pemrograman terkadang membuat dosen pengajar mengalokasikan waktu ekstra untuk mengoreksi dan menilai tugas-tugas yang terkumpul. Maka diperlukan suatu kakas untuk membantu dosen pengajar melakukan penilaian terhadap tugas pemrograman.

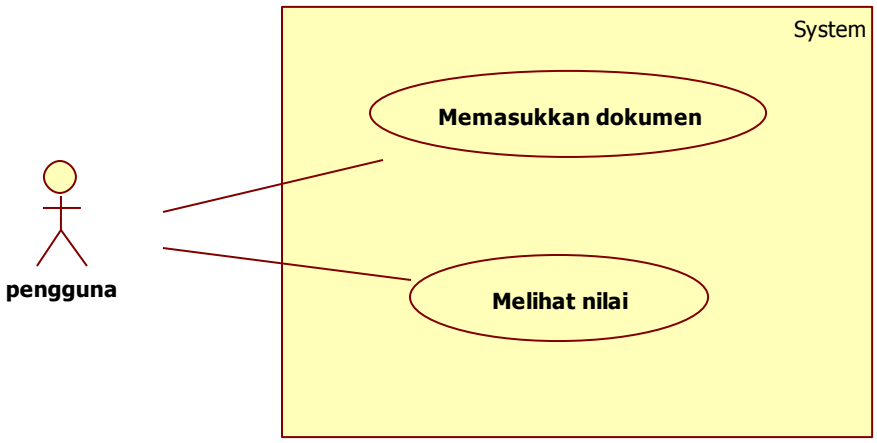
Tugas akhir ini akan membuat suatu kakas yang akan membandingkan kode solusi mahasiswa dan kode solusi dosen. Selain itu tugas akhir ini juga memberikan nilai dari kode solusi yang di-*input*-kan. Metode yang digunakan dalam tugas akhir ini adalah *graph similarity.*

Proses inti dalam kakas ini adalah membandingkan dua kode. Dimulai dengan proses *parsing* kode untuk mendapatkan bentuk *tree.* Bentuk inilah yang kemudian akan dibandingkan menggunakan metode *graph similarity.*

Dalam tugas akhir ini penulis menggunakan ANTLR sebagai *parser.* ANTLR membutuhkan *grammar* untuk mengenali suatu *input.* Singkatnya, ANTLR mengubah sebuah *grammar* menjadi program yang mengenali *input* dalam suatu bahasa yang memenuhi aturan dari *grammar* tersebut.

Hasil dari proses *parsing* adalah graph dalam bentuk AST. Setelah diperoleh AST dari kode solusi mahasiswa dan dosen, selanjutnya akan dilakukan *compare* untuk memperoleh nilai-nilai sebagai *output* akhir dari kakas ini.

Kasus pengguna pada Gambar 3 digunakan untuk mendukung pengerjaan tugas akhir ini. Dari gambar dapat diketahui bahwa dengan kakas yang akan dibangun pengguna dapat memasukkan dokumen yang berupa kode solusi dan melihat nilai hasil perbandingan.



Gambar 3. Kasus Pengguna dari Kakas yang akan Dibangun.

10.METODOLOGI

a. Penyusunan proposal tugas akhir

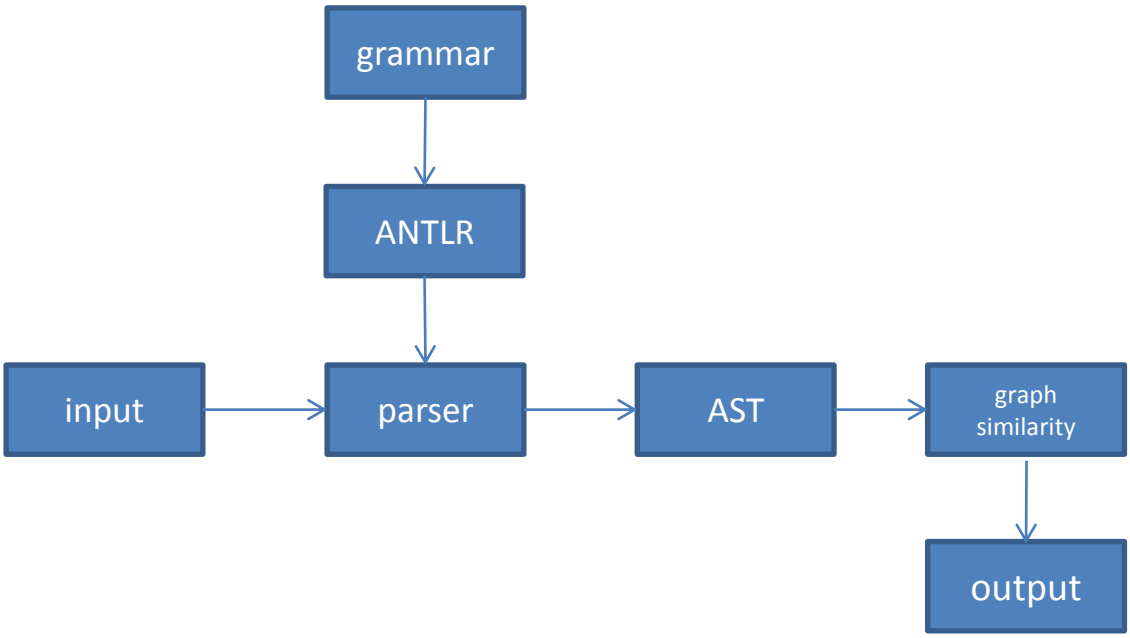
Tahap awal pengerjaan tugas akhir ini adalah penyusunan proposal tugas akhir. Pada proposal ini penulis mengajukan pembuatan aplikasi autograder menggunakan ANTLR dan *graph similarity*.

b. Studi literatur

Tugas akhir ini menggunakan makalah beserta beberapa artikel dari internet [5].

c. Analisis dan desain perangkat lunak

Pada tahap ini dilakukan perancangan sistem menggunakan studi literatur dan mempelajari konsep perangkat lunak yang akan dibangun. Selain itu juga akan dibuat bentuk awal dari perangkat lunak beserta proses-proses di dalamnya. Pada tahap ini juga dilakukan analisis awal dan pendefinisian kebutuhan sistem untuk mengetahui masalah yang sedang dihadapi. Dari proses tersebut selanjutnya dirumuskan rancangan sistem yang dapat memberi pemecahan masalah tersebut. Alur proses pada kakas ditunjukkan pada Gambar 4.



Gambar 4. Proses pada Sistem yang akan Dibangun.

d. Implementasi perangkat lunak

Dalam mengerjakan tugas akhir ini digunakan metode *graph similarity* untuk membandingkan *tree* hasil *parsing* kode solusi dosen dan kode solusi mahasiswa. Implementasi menggunakan bahasa Java dan IDE Eclipse.

e. Pengujian dan evaluasi

Pada tahap ini akan dilakukan pengujian terhadap aplikasi yang telah dibangun menggunakan tugas-tugas dari mata kuliah Pemrograman Berorientasi Objek tahun 2013-2014.

f. Penyusunan Buku Tugas Akhir

Pada tahap ini disusun laporan tugas akhir sebagai dokumentasi pelaksanaan tugas akhir, yang mencakup seluruh konsep, teori, implementasi, serta hasil yang telah dikerjakan. Laporan tugas akhir ini akan dibagi menjadi beberapa bab sebagai berikut.

1. Pendahuluan
    a. Latar Belakang
    b. Rumusan Masalah
    c. Batasan Tugas Akhir

d. Tujuan
e. Metodologi
f. Sistematika Penulisan

2. Tinjauan Pustaka
3. Desain dan Implementasi
4. Pengujian dan Evaluasi
5. Kesimpulan dan Saran
6. Daftar Pustaka

11. JADWAL KEGIATAN

Pengerjaan tugas akhir akan berlangsung sebagaimana ditampilkan pada Tabel 1.

Tabel 1. Jadwal Kegiatan.

Tahapan	Tahun 2014																			
	Februari				Maret				April				Mei				Juni			
Penyusunan proposal		■	■	■	■															
Studi Literatur		■	■	■	■	■	■	■	■											
Perancangan sistem							■	■	■	■	■	■	■	■	■	■				
Implementasi							■	■	■	■	■	■	■	■	■	■	■	■		
Pengujian dan evaluasi												■	■	■	■	■	■	■	■	
Penyusunan buku												■	■	■	■	■	■	■	■	■

12. DAFTAR PUSTAKA

[1] Deitel, P., & Deitel, H. *C++ How To Program*. Boston: Pearson Education, Inc, 2012.

[2] Parr, T. *The Definitife ANTLR 4 Reference*. Dallas: Pragmatic Bookshelf, 2013.

[3] *Abstract Syntax Tree*. (2014, Februari 18). Retrieved Februari 18, 2014, from Wikipedia: http://en.wikipedia.org/wiki/Abstract_syntax_tree

[4] Koutra, D., Parikh, A., Ramdas, A., & Xiang, J. "Algorithm for Graph Similarity and Subgraph Mtching," Desember 2011.

[5] Naude, K. A., Greyling, J. H., & Vogts, D. "Marking Student Programs Using Graph Similarity," *Science Direct*, vol. 1, pp. 545-561, September 2009.