Tugas Besar Teori Bahasa Formal dan Automata

oleh

Ardysatrio Fakhri Haroen	13517062
Anzaldi Sulaiman Oemar	13517098
M Algah Fattah Illahi	13517122

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2018

Daftar Isi

Deskripsi Persoalan

Mahasiswa diminta untuk membuat sebuah program untuk mengenali dan menghitung ekspresi matematika sederhana dengan menggunakan implementasi dari *Context Free Grammar(CFG)* dan/atau *Pushdown Automata(PDA).* Bila program diberi masukan sebuah ekspresi matematika, program harus bisa mengenali apakah ekspresi tersebut valid atau tidak (*syntax error)*. Bila ekspresi tersebut valid, program akan menghitung nilai dari ekspresi tersebut dengan terlebih dahulu setiap simbol terminal (angka) menjadi nilai numerik yang bersesuaian. Program juga harus dapat mengenali apakah ekspresi tersebut mungkin dihitung atau tidak *(math error).*

Contoh ekspresi matematika yang valid adalah (-457.01+1280) * (35.7-11.0233)/(-6.1450) (setelah di- enter akan menampilkan hasil perhitungan ekspresi tersebut yaitu -3304.91). Contoh ekspresi tidak valid adalah 3*+-12/(57) (setelah di- enter akan ditampilkan pesan " SYNTAX ERROR"), atau (-5)^(2/3) (setelah di- enter akan ditampilkan pesan " MATH ERROR").

Batasan Masalah

Simbol terminal hanya terdiri dari simbol aritmatika biasa (+, -, +, /), perpangkatan (^), tanda negatif (-), angka (0 ,1 , 2, 3, 4, 5, 6, 7, 8, 9), desimal (.), dan tanda kurung ((,)). Operator hanya terdiri dari simbol aritmatika biasa, tidak mengandung huruf-huruf (e,pi, dan lain-lain). Tidak ada spasi antar token.

Untuk implementasi fungsi pangkat, perhatikan bahwa terdapat kemungkinan implementasi fungsi pangkat negatif dan fungsi pangkat pecahan (akar). Silakan gunakan notasi 9^(0.5) untuk menuliskan notasi akar pangkat 2 dan notasi 2^(-1) untuk notasi $\frac{1}{2}$ pada command prompt.

Dalam implementasi perhitungan pada kalkulator, gunakan aturan sebagai berikut.

1. Operasi yang berada dalam kurung dikerjakan lebih dahulu.
2. Perhatikan urutan eksekusi operasi.
3. Kerjakan berurutan dari kiri (Contoh: 23 + 12 - 16 = 35 - 16 = 19 atau 8 * 5 : 2 = 40 : 2 = 20 atau 72 : 2 - 11 = 36-11 = 25), kecuali untuk pangkat dari kanan (Contoh: 4^3^2 = 4^9, bukan 64^2).

Operan terdiri dari bilangan riil (baik positif atau negatif), tidak hanya bilangan bulat, dari digit (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). Program merupakan implementasi dari tata bahasa dan PDA yang dibuat terlebih dahulu menggunakan teori yang telah dipelajari.

BONUS: Program dapat menangani masukan dan hasil ekspresi bilangan imajiner dan bilangan kompleks. Bilangan imajiner ditandai dengan huruf i.

Deskripsi Umum Solusi

Jawaban permasalahan ini dapat diselesaikan dengan menggunakan konsep *Pushdown Automata (PDA).* dimana implementasi dari PDA dalam bahasa pemrograman adalah berupa *stack.* Dengan PDA, ekspresi matematika dapat dibaca untuk nanti diketahui apakah sintaks yang dimasukkan sudah sesuai atau belum.

Jika ekspresi yang dimasukkan memenuhi sintaks, selanjutnya kita akan menghitung nilainya. Ada banyak cara untuk menghitung nilai ini, seperti dengan menggunakan *tree, array* atau *stack.* Program ini akan menggunakan *array* untuk menghitung nilai ekspresi matematika yang diterimanya.

Ide awal untuk menghitung nilai ekspresi dengan array :

1. Nilai dari bagian ekspresi yang memiliki prioritas tinggi seperti ekspresi dalam tanda kurung, perkalian, pembagian, dan pemangkatan akan langsung dihitung kemudian hasilnya akan dimasukkan ke dalam array.
2. Operasi dengan prioritas rendah (penjumlahan dan pengurangan), setiap operannya dimasukkan ke dalam *array of operand* mulai dari indeks ke-0. Operator prioritas rendah disimpan ke dalam *array of operator,* dimana indeks ke-n merupakan operasi dari *array of operand* indeks ke-n dengan indeks ke-n+1.

Context Free Grammar/ Pushdown Automata yang digunakan

$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$

States : q_0, q_1, q_2, q_3

Input Symbols : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ., *, /, +, -,^, (,)

Stack Symbol : X, Z_0

Tabel transisi

$\delta(q_0,1,Z_0)$	(q_2,Z_0)		$\delta(q_0,2,Z_0)$	(q_2,Z_0)		$\delta(q_0,3,Z_0)$	(q_2,Z_0)
$\delta(q_0,4,Z_0)$	(q_2,Z_0)		$\delta(q_0,5,Z_0)$	(q_2,Z_0)		$\delta(q_0,6,Z_0)$	(q_2,Z_0)
$\delta(q_0,7,Z_0)$	(q_2,Z_0)		$\delta(q_0,8,Z_0)$	(q_2,Z_0)		$\delta(q_0,9,Z_0)$	(q_2,Z_0)
$\delta(q_0,1,X)$	(q_2,X)		$\delta(q_0,2,X)$	(q_2,X)		$\delta(q_0,3,X)$	(q_2,X)
$\delta(q_0,4,X)$	(q_2,X)		$\delta(q_0,5,X)$	(q_2,X)		$\delta(q_0,6,X)$	(q_2,X)
$\delta(q_0,7,X)$	(q_2,X)		$\delta(q_0,8,X)$	(q_2,X)		$\delta(q_0,9,X)$	(q_2,X)
$\delta(q_1,1,Z_0)$	(q_2,Z_0)		$\delta(q_1,2,Z_0)$	(q_2,Z_0)		$\delta(q_1,3,Z_0)$	(q_2,Z_0)
$\delta(q_1,4,Z_0)$	(q_2,Z_0)		$\delta(q_1,5,Z_0)$	(q_2,Z_0)		$\delta(q_1,6,Z_0)$	(q_2,Z_0)
$\delta(q_1,7,Z_0)$	(q_2,Z_0)		$\delta(q_1,8,Z_0)$	(q_2,Z_0)		$\delta(q_1,9,Z_0)$	(q_2,Z_0)
$\delta(q_1,1,X)$	(q_2,X)		$\delta(q_1,2,X)$	(q_2,X)		$\delta(q_1,3,X)$	(q_2,X)
$\delta(q_1,4,X)$	(q_2,X)		$\delta(q_1,5,X)$	(q_2,X)		$\delta(q_1,6,X)$	(q_2,X)
$\delta(q_1,7,X)$	(q_2,X)		$\delta(q_1,8,X)$	(q_2,X)		$\delta(q_1,9,X)$	(q_2,X)
$\delta(q_2,0,Z_0)$	(q_2,Z_0)		$\delta(q_2,1,Z_0)$	(q_2,Z_0)		$\delta(q_2,2,Z_0)$	(q_2,Z_0)
$\delta(q_2,3,Z_0)$	(q_2,Z_0)		$\delta(q_2,4,Z_0)$	(q_2,Z_0)		$\delta(q_2,5,Z_0)$	(q_2,Z_0)
$\delta(q_2,6,Z_0)$	(q_2,Z_0)		$\delta(q_2,7,Z_0)$	(q_2,Z_0)		$\delta(q_2,8,Z_0)$	(q_2,Z_0)
$\delta(q_2,9,Z_0)$	(q_2,Z_0)		$\delta(q_2,0,X)$	(q_2,X)		$\delta(q_2,1,X)$	(q_2,X)

$\delta(q_2,2,X)$	(q_2,X)		$\delta(q_2,3,X)$	(q_2,X)		$\delta(q_2,4,X)$	(q_2,X)
$\delta(q_2,5,X)$	(q_2,X)		$\delta(q_2,6,X)$	(q_2,X)		$\delta(q_2,7,X)$	(q_2,X)
$\delta(q_2,8,X)$	(q_2,X)		$\delta(q_2,9,X)$	(q_2,X)		$\delta(q_3,1,Z_0)$	(q_2,Z_0)
$\delta(q_3,2,Z_0)$	(q_2,Z_0)		$\delta(q_3,3,Z_0)$	(q_2,Z_0)		$\delta(q_3,4,Z_0)$	(q_2,Z_0)
$\delta(q_3,5,Z_0)$	(q_2,Z_0)		$\delta(q_3,6,Z_0)$	(q_2,Z_0)		$\delta(q_3,7,Z_0)$	(q_2,Z_0)
$\delta(q_3,8,Z_0)$	(q_2,Z_0)		$\delta(q_3,9,Z_0)$	(q_2,Z_0)		$\delta(q_3,1,X)$	(q_2,X)
$\delta(q_3,2,X)$	(q_2,X)		$\delta(q_3,3,X)$	(q_2,X)		$\delta(q_3,4,X)$	(q_2,X)
$\delta(q_3,5,X)$	(q_2,X)		$\delta(q_3,6,X)$	(q_2,X)		$\delta(q_3,7,X)$	(q_2,X)
$\delta(q_3,8,X)$	(q_2,X)		$\delta(q_3,9,X)$	(q_2,X)			
$\delta(q_0,(,Z_0)$	(q_0,XZ_0)		$\delta(q_0,(,X)$	(q_0,XX)		$\delta(q_0,-,Z_0)$	(q_1,Z_0)
$\delta(q_0,-,X)$	(q_1,X)		$\delta(q_1,(,Z_0)$	$q_0,XZ_0)$		$\delta(q_1,(,X)$	(q_0,XX)
$\delta(q_2,.,Z_0)$	(q_2,Z_0)		$\delta(q_2,.,X)$	(q_2,X)		$\delta(q_2,),X)$	(q_2,ε)
$\delta(q_2,+,Z_0)$	(q_3,Z_0)		$\delta(q_2,-,Z_0)$	(q_3,Z_0)		$\delta(q_2,*,Z_0)$	(q_3,Z_0)
$\delta(q_2,/,Z_0)$	(q_3,Z_0)		$\delta(q_2,\wedge,Z_0)$	(q_3,Z_0)		$\delta(q_2,+,Z_0)$	(q_3,X)
$\delta(q_2,-,Z_0)$	(q_3,X)		$\delta(q_2,*,Z_0)$	(q_3,X)		$\delta(q_2,/,Z_0)$	(q_3,X)
$\delta(q_2,\wedge,Z_0)$	(q_3,X)		$\delta(q_3,(,Z_0)$	(q_0,Z_0)		$\delta(q_3,(,X)$	(q_2,X)

Source Code

boolean.h

```
#ifndef _BOOLEAN_h
#define _BOOLEAN_h
#define boolean unsigned char
#define true 1
#define false 0
#endif
```

stack.h

```
#ifndef stack_H
#define stack_H

#include "boolean.h"
#include <stdio.h>

#define Nil 0
#define MaxEl 100
/* Nil adalah stack dengan elemen kosong . */
/* Karena indeks dalam bhs C dimulai 0 maka tabel dg indeks 0 tidak dipakai
*/

typedef char infotype;
typedef int address;   /* indeks tabel */

/* Contoh deklarasi variabel bertype stack dengan ciri TOP : */
/* Versi I : dengan menyimpan tabel dan alamat top secara eksplisit*/
typedef struct {
    infotype T[MaxEl+1]; /* tabel penyimpan elemen */
    address TOP;  /* alamat TOP: elemen puncak */
} Stack;
/* Definisi stack S kosong : S.TOP = Nil */
/* Elemen yang dipakai menyimpan nilai Stack T[1]..T[MaxEl] */
/* Jika S adalah Stack maka akses elemen : */
   /* S.T[(S.TOP)] untuk mengakses elemen TOP */
   /* S.TOP adalah alamat elemen TOP */

/* Definisi akses dengan Selektor : Set dan Get */
#define Top(S) (S).TOP
```

```
#define InfoTop(S) (S).T[(S).TOP]

/* ************ Prototype ************ */
/* *** Konstruktor/Kreator *** */
void CreateEmpty (Stack *S);
/* I.S. sembarang; */
/* F.S. Membuat sebuah stack S yang kosong berkapasitas MaxEl */
/* jadi indeksnya antara 1.. MaxEl+1 karena 0 tidak dipakai */
/* Ciri stack kosong : TOP bernilai Nil */

/* ************ Predikat Untuk test keadaan KOLEKSI ************ */
boolean IsEmpty (Stack S);
/* Mengirim true jika Stack kosong: lihat definisi di atas */
boolean IsFull (Stack S);
/* Mengirim true jika tabel penampung nilai elemen stack penuh */

/* ************ Menambahkan sebuah elemen ke Stack ************ */
void Push (Stack * S, infotype X);
/* Menambahkan X sebagai elemen Stack S. */
/* I.S. S mungkin kosong, tabel penampung elemen stack TIDAK penuh */
/* F.S. X menjadi TOP yang baru,TOP bertambah 1 */

/* ************ Menghapus sebuah elemen Stack ************ */
void Pop (Stack * S, infotype* X);
/* Menghapus X dari Stack S. */
/* I.S. S  tidak mungkin kosong */
/* F.S. X adalah nilai elemen TOP yang lama, TOP berkurang 1 */

#endif
```

stack.c

```
#include "stack.h"


/* *** Konstruktor/Kreator *** */
void CreateEmpty (Stack *S){
    Top(*S) = Nil;
    InfoTop(*S) = Nil;
}
/* I.S. sembarang; */
/* F.S. Membuat sebuah stack S yang kosong berkapasitas MaxEl */
```

```
/* jadi indeksnya antara 1.. MaxEl+1 karena 0 tidak dipakai */
/* Ciri stack kosong : TOP bernilai Nil */

/* ************ Predikat Untuk test keadaan KOLEKSI ************ */
boolean IsEmpty (Stack S){
    return Top(S) == Nil;
}
/* Mengirim true jika Stack kosong: lihat definisi di atas */
boolean IsFull (Stack S){
    return Top(S) == MaxEl;
}
/* Mengirim true jika tabel penampung nilai elemen stack penuh */

/* ************ Menambahkan sebuah elemen ke Stack ************ */
void Push (Stack * S, infotype X){
    Top(*S)++;
    InfoTop(*S) = X;
}
/* Menambahkan X sebagai elemen Stack S. */
/* I.S. S mungkin kosong, tabel penampung elemen stack TIDAK penuh */
/* F.S. X menjadi TOP yang baru,TOP bertambah 1 */

/* ************ Menghapus sebuah elemen Stack ************ */
void Pop (Stack * S, infotype* X){
    *X = InfoTop(*S);
    Top(*S)--;

}
/* Menghapus X dari Stack S. */
/* I.S. S  tidak mungkin kosong */
/* F.S. X adalah nilai elemen TOP yang lama, TOP berkurang 1 */
```

pda.h

```
#ifndef pdacalculator_H
#define pdacalculator_H

#include "boolean.h"
#include "string.h"
#include "stack.h"
#include <stdio.h>
```

```
float StringTofloat(char *s);
/* Mengubah string dari sebuah bilangan ke bentuk float
I.S : s terdefinisi sebagai sebuah string berbentuk bilangan dan tidak
kosong,
minimal adalah 2 karakter untuk bilangan negatif dan 1 karakter untuk bil.
positif
F.S : mengembalikan hasil konversi dari string s
*/
void trans_state(int *State, char Symbol, Stack *S,boolean *stuck);
/* Prosedur transisi PDA
State terdefinisi 0,1,2,3. Symbol merupakan karakter yang sedang dibaca. S
merupakan Stack yang digunakan. stuck bernilai true jika Symbol tidak valid
*/

boolean PDA(char *s);
/* Validasi apakah string merupakan ekspresi matematika yang valid?
I.S : str terdefinisi sebagai sebuah string
F.S : jika str merupakan ekspresi matematika, output true. Jika tidak,
output false
*/

void proc_in_parentheses(char **s, boolean *error, float temp[], int idx);

void proc_string(char *s, boolean *error, float *res);
/* Menghitung operasi matematika dari string yang diinputkan
Prekondisi  : Sintaks string harus benar
F.S         : Mengeluarkan hasil dari operasi matematika string tersebut
Operasi yang ada (+,-,*,/,())
*/

float proc_parentheses(char *s, boolean *error, float *res);
/* Algoritma khusus menghitung operasi dalam kurung
Prekondisi  : Sintaks merupakan operasi dalam kurung
F.S         : Menghasilkan hasil operasi dalam kurung
*/

#endif
```

pda.c

```
#include "pda.h"
#include "boolean.h"
#include <math.h>
#include <stdio.h>
```

```
float StringTofloat(char *s){ //mengubah array of char menjadi float
    float result = 0;
    float SignDec = 1;

    if(*s == '-'){//menangani bilangan negatif
       s++;
       SignDec = -1;
    }

    boolean point = false;

    for(; *s; s++){

       if(*s == '.'){//menandai mulainya koma
             point = true;
       }

       int    temp = *s - '0'; //menampung hasil konversi karaker ke
bilangan

       if(temp >= 0 && temp <= 9){
             if(point){
                   SignDec = SignDec *10; //menghitung jumlah digit di
belakang koma
                                                          //dimana jumlah digit
di belakang koma = log(abs(SignDec)
             }
             result = result * 10 + (float) temp;
       }

    }
    return result / SignDec;
}

void trans_state(int *State, char Symbol, Stack *S,boolean *stuck){
//menangani state transition untuk PDA
    infotype temp;

    switch (*State){
       case 0: //State 0, menangani angka, buka kurung, dan tanda negatif
             switch (Symbol) {
                   case '(':
                         if(InfoTop(*S) == 'Z' || InfoTop(*S) == 'X'){
                               Push(S,'X');
                               break;
                         }
```

```
                  case '0':
                  case '1':
                  case '2':
                  case '3':
                  case '4':
                  case '5':
                  case '6':
                  case '7':
                  case '8':
                  case '9':
                        if(InfoTop(*S) == 'X' || InfoTop(*S) == 'Z'){
                              *State = 2;
                              break;
                        }
                  case '-':
                        if(InfoTop(*S) == 'X' || InfoTop(*S) == 'Z'){
                              *State = 1;
                              break;
                        }
                  default : //selain kondisi diatas maka stuck
                        *stuck = true;

            }
            break;

      case 1: //State 1, menangani ekspresi setelah tanda '-', bisa berupa
angka atau '('
            switch (Symbol){
                  case '(':
                        if(InfoTop(*S) == 'X' || InfoTop(*S) == 'Z'){
                              Push(S,'X');
                              *State = 0;
                              break;
                        }
                  case '0':
                  case '1':
                  case '2':
                  case '3':
                  case '4':
                  case '5':
                  case '6':
                  case '7':
                  case '8':
                  case '9':
                        if(InfoTop(*S) == 'X' || InfoTop(*S) == 'Z'){
                              *State = 2;
                              break;
```

```
                        }
                  default :
                        *stuck = true;
            }
            break;

      case 2: //menangani state 2, penambahan digit angka atau adanya
operator atau ada ')'
            switch (Symbol){
                  case '0':
                  case '1':
                  case '2':
                  case '3':
                  case '4':
                  case '5':
                  case '6':
                  case '7':
                  case '8':
                  case '9':
                  case '.':
                        if(InfoTop(*S) == 'X' || InfoTop(*S) == 'Z'){
                              *State = 2;
                              break;
                        }
                  case '^':
                  case '*':
                  case '/':
                  case '+':
                  case '-':
                        if(InfoTop(*S) == 'X' || InfoTop(*S) == 'Z'){
                              *State = 3;
                              break;
                        }
                  case ')':
                        if(InfoTop(*S) == 'X'){
                              Pop(S,&temp);
                              *State = 2;
                              break;
                        }
                  default :
                        *stuck = true;
            }
            break;

      case 3: //menangani ekspresi setelah ada ')' atau operator lain
            switch (Symbol){
                  case '0':
```

```
                    case '1':
                    case '2':
                    case '3':
                    case '4':
                    case '5':
                    case '6':
                    case '7':
                    case '8':
                    case '9':
                          if(InfoTop(*S) == 'X' || InfoTop(*S) == 'Z'){
                                *State = 2;
                                break;
                          }
                    case '(':
                          if(InfoTop(*S) == 'X' || InfoTop(*S) == 'Z'){
                                Push(S,'X');
                                *State = 0;
                                break;
                          }
                    default :
                          *stuck = true;
              }
              break;


    }

}


boolean PDA(char *s){//menangani transisi antar state dan manajemen stack
    Stack S;
    infotype temp;
    boolean stuck = false;
    int state = 0;
    CreateEmpty(&S);

    Push(&S,'Z');

    while(*s){
       if(*s != '\n' && !stuck){
             trans_state(&state, *s, &S, &stuck);
       }
       s++;
    }

    if(InfoTop(S) == 'Z' && state == 2 && !stuck){
```

```
       Pop(&S,&temp);
    }

    return IsEmpty(S);
}

void proc_in_parentheses(char **s, boolean *error, float temp[], int idx){
    float tempres;
    int count;

    proc_parentheses(*s, error, &tempres);
    if(*error){
    }
    else {
       temp[idx] = tempres;
       //printf("parenth res in %f\n", temp[idx]);
       count = 1;
    } //mengembalikan angka/ekspresi yang ada di parentheses

}


void proc_string(char *s, boolean *error, float *res){
    float result;
    float temp[50];
    float f;
    float tempres;
    char flt[50];
    char *temps;
    char operand[50];
    char opt;
    int idxa = 0;
    int idxc = 0;
    int idxpow = 0;
    int idx = 0;
    int count;
    boolean OneElmt = true;
    boolean preopt;
    boolean Minus = false;
    boolean firstopt = true;
    boolean multiplied = false;
    *error = false;

    while((*s != '\0') && !*error){
       if(*s == '('){

             temps = s;
```

```
                  proc_in_parentheses(&temps, error, temp, idx);
            if(*error) break;
            else{
                  s = temps;
                  count = 1;
            s++;
            while(count != 0){
                  //melanjutkan pembacaan sampai akhir kurung
                  if(*s == ')'){
                        count--;
                  } else  if(*s == '('){
                        count++;
                  }
                  s++;
            }
            }
      }
      else {
            //menangani bila yang muncul langsung angka
            if(*s == '-'){
                  flt[idxc] = *s;
                  idxc++;
                  s++;
                  Minus = true;
            }
            if(idxa != 0){
                  //mengosongkan array flt yang akan diisi angka
                  idxa--;
                  while(idxa >= 0){
                        flt[idxa] = '\0';
                        idxa--;
                  }
                  idxa = 0;
            }
            while ((*s != '\0') && (*s != '+') && (*s != '-') && (*s !=
'*') && (*s != '/') && (*s != '^')){
                  //memasukkan digit-digit angka ke array flt
                  flt[idxa] = *s;
                  idxa++;
                  s++;
            }
            temp[idx] = StringTofloat(flt);//mengubah isi array flt
menjadi float
            if(Minus) temp[idx] *= -1;
            //printf("temp[%d] %f\n", idx, temp[idx]);
      }
```

```
	if(*s != '\0'){
		//menangani munculnya operator
		if (firstopt){
			preopt = *s;
			firstopt = false;
		}  //mengeset firstopt (operator pertama)
		else preopt = opt;

		opt = *s;
		while(opt == '*' || opt == '/' || opt == '^'){
			multiplied = true;
			if (idxc == 0) OneElmt = true;

			s++;
			if(*s == '('){
				//menangani tanda kurung
				proc_parentheses(s, error, &tempres);
				if(*error){
					break;
				}
				else {
					f= tempres; //printf("parenth res %f\n",
temp[idx]);}
					count = 1;
					s++;
					while(count != 0){
						if(*s == '('){
							count++;
						} else if(*s == ')'){
							count--;
						}
						s++;
					}
				}
			}
			else {
				//menangani angka
				if(idxa != 0){
					//loop buat mengosongkan array flt yang
dipake buat menyimpan angka
					idxa--;
					while(idxa >= 0){
						flt[idxa] = '\0';
						idxa--;
					}
					idxa = 0;
				}
```

```
                        while ((*s != '\0') && (*s != '+') && (*s != '-')
&& (*s != '*') && (*s != '/') && (*s != '^')){
                                    //loop buat menangkap semua angka dan
mengkonversi jadi float
                                    flt[idxa] = *s;
                                    idxa++;
                                    s++;
                        }
                        f = StringTofloat(flt);
                  }

                  if(opt == '^'){//menangani perpangkatana
                        if((temp[idx] < 0) && ((f < 1) && (f > 0))){
//bilangan negatif dipangkat dengan bilangan kurang dari 1
                              *error = true;
                              break;
                        }
                        else{
                              idx++;
                              temp[idx] = f;
                              idxpow++;
                        }
                  }
                  else if(opt == '*'){//menangani perkalian
                        temp[idx] *= f;
                  }
                  else if(opt == '/'){//menangani pembagian
                        if(f == 0){//bila bilangan yang baru dibaca berupa
0, maka error karena division by zero
                              *error = true;
                              break;
                        }
                        else temp[idx] /= f;
                  }

                  preopt = opt;
                  opt = *s;

                  //printf("preopt %c\n", preopt);
                  //printf("opt %c\n", opt);
                  //printf("temp[%d] %f\n", idx, temp[idx]);

                  if((preopt == '^') && (opt != '^') && (idxpow != 0)){
                        //menangani perpangkatan yang lebih dari satu
berturut turut, dan setelahnya ada perkalian
                        //atau pembagian
                        //printf("inloop 1\n");
```

```
                        //printf("idxpow %d\n", idxpow);
                        for (int k = 0; k < idxpow; k++){
                              idx--;
                              temp[idx] = pow(temp[idx], temp[idx+1]);
                              //printf("temp[%d] %f\n", idx, temp[idx]);
                        }
                        idxpow = 0;
                  }
            }

            if((preopt == '^') && (opt != '^') && (idxpow != 0)){
                  //menangani perpangkatan lebih dari satu kali berturut
turut, dan setelahnya ada
                  //karakter break atau tambah atau
                  //printf("inloop 2\n");
                  //printf("idxpow %d\n", idxpow);
                  for (int k = 0; k < idxpow; k++){
                        idx--;
                        temp[idx] = pow(temp[idx], temp[idx+1]);
                        //printf("temp[%d] %f\n", idx, temp[idx]);
                  }
                  idxpow = 0;
            }

            //printf("outofloop\n");
            //printf("temp[%d] %f\n", idx, temp[idx]);

            if(((opt == '+') || (opt == '-')) && (*s != '\0')){
                  //menandai berapa banyak tanda + dan - yang ada, dan
dimana letaknya
                  operand[idxc] = opt;
                  //printf("operand[%d] %c\n", idxc, operand[idxc]);
                  OneElmt = false;
                  idxc++;
                  s++;
            }
            idx++;
            //printf("loopidx %d\n", idx);
      }
   }

   if(OneElmt){
      //kondisi isi array bilangan hanya 1
      //printf("case 1 idx%d\n", idx);
      result = temp[0];
   }
   else {
```

```
        /* if(multiplied) idx--; */
        //printf("case 2 idx %d\n", idx);
        result = 0;
        for(int i = 0; i < idx; i++){
              //printf("sumloop\n");
              //printf("temp[%d] %f\n", i, temp[i]);
              //printf("temp[%d] %f\n", i+1, temp[i+1]);
              if(i == 0){
                    if(operand[i] == '+'){
                          result = result + temp[i] + temp[i+1];
                    } else if(operand[i] == '-'){
                          result = result + temp[i] - temp[i+1];
                    }
              } else {
                    if(operand[i] == '+'){
                          result += temp[i+1];
                    } else if(operand[i] == '-'){
                          result -= temp[i+1];
                    }
              }
        }
    }
    *res = result;
}


float proc_parentheses(char *s, boolean *error, float *res){
    float tabfl[50];
    float result = 0;
    float temp;
    float tempres;
    char flt[50];
    char operand[50];
    char opt;
    char preopt;
    int count;
    int idx = 0;
    int idxc = 0;
    int idxa = 0;
    int idxpow = 0;

    boolean OneElmt = true;
    boolean Minus = false;
    boolean FirstElmt = true;
    boolean firstopt = false;
    boolean multiplied = false;
    *error = false;
```

```
    s++;
    if(*s == '-'){
       flt[idxc] = *s;
       idxc++;
       s++;
       Minus = true;
    }
    while(*s != ')'){
       if((!Minus) || (!FirstElmt)){
             //kalo  !(minus dan first element), maka kosongkan array flt
yang buat menyimpan angka
             if(idxc != 0){
                   idxc--;
                   while(idxc >= 0){
                         flt[idxc] = '\0';
                         idxc--;
                   }
                   idxc = 0;
             }
       }

       if(*s == '('){
             proc_parentheses(s, error, &tempres);
             if(*error){
                   break;
             }
             else tabfl[idx] = tempres;
             ////printf("tabfl[%d] %f\n", idx, tabfl[idx]);

             if(Minus){
                   tabfl[idx] *= -1;
             }
             count = 1; //merupakan jumlah ')' yang dibutuhkan
             s++;
             while (count != 0){
                   if(*s == ')'){
                         count--;
                   } else if(*s == '('){
                         count++;
                   }
                   s++;

             }
       }
       else {
             while((*s != ')') && (*s != '*') && (*s != '/') && (*s != '+')
```

```
&& (*s != '-') && (*s != '^')){
                    flt[idxc] = *s;
                    idxc++;
                    s++;
                    }
                    tabfl[idx] = StringTofloat(flt);
              //printf("tabfl[%d] %f\n", idx, tabfl[idx]);

       }

       FirstElmt = false;

       if(*s != ')'){
              if (firstopt){
                    preopt = *s;
                    firstopt = false;
              }  //mengeset firstopt (operator pertama)
              else preopt = opt;

              opt = *s;
              while(opt == '*' || opt == '/' || (opt == '^')){
                    if(idxa == 0) OneElmt = true;
                    multiplied = true;
                    s++;
                    if(*s == '('){
                          proc_parentheses(s, error, &tempres);
                          if(*error){
                                break;
                          }
                          else temp = tempres;

                          count = 1;
                          s++;
                          while(count != 0){
                                if(*s == ')'){
                                      count--;
                                } else if(*s == '('){
                                      count++;
                                }
                                s++;
                          }
                    } else {
                          if (idxc != 0){
                                idxc--;
                                while(idxc >= 0){
                                      flt[idxc] = '\0';
                                      idxc--;
```

```
                              }
                              idxc = 0;
                        }
                        while((*s != ')') && (*s != '*') && (*s != '/') &&
(*s != '+') && (*s != '-') && (*s != '^')){
                              flt[idxc] = *s;
                              idxc++;
                              s++;
                        }
                        temp = StringTofloat(flt);
                  }

                  if(opt == '^'){
                        if((tabfl[idx] < 0) && (temp < 1)){
                              *error = true;
                              break;
                        }
                        else{
                              idx++;
                              tabfl[idx] = temp;
                              idxpow++;
                        }
                  }
                  else if(opt == '*'){
                        tabfl[idx] *= temp;
                  }
                  else if(opt == '/'){
                        if(temp == 0){
                              *error = true;
                              break;
                        }
                        else{
                              tabfl[idx] /= temp;
                        }
                  }
                  preopt = opt;
                  opt = *s;

                  if((preopt == '^') && (opt != '^') && (idxpow != 0)){
                        //printf("inloop 1\n");
                        //printf("idxpow %d\n", idxpow);
                        for (int k = 0; k < idxpow; k++){
                              idx--;
                              tabfl[idx] = pow(tabfl[idx], tabfl[idx+1]);
                              //printf("tabfl[%d] %f\n", idx, tabfl[idx]);
                        }
                        idxpow = 0;
```

```
                }
            }

            //pemrosesan pangkat(diulang)
            if((preopt == '^') && (opt != '^') && (idxpow != 0)){
                        //printf("inloop 2\n");
                        //printf("idxpow %d\n", idxpow);
                        for (int k = 0; k < idxpow; k++){
                              idx--;
                              tabfl[idx] = pow(tabfl[idx], tabfl[idx+1]);
                              //printf("tabfl[%d] %f\n", idx, tabfl[idx]);
                        }
                        idxpow = 0;
                }

            if (((opt == '+') || (opt == '-')) && (*s != ')')){
                  operand[idxa] = opt;
                  OneElmt = false;
                  idxa++;
                  s++;
            }
            idx++;
      }
   }


   if(OneElmt){
      result = tabfl[0];
      //printf("case 1 idx%d\n", idx);
      //printf("result %f\n", result);

   } else {
      /* if(multiplied) idx--; */
      //pemrosesan tab(diulang)
      //printf("case 2 idx%d\n", idx);
      result = 0;
      for(int i = 0; i < idx; i++){
            if(i == 0){
                  //printf("sumloop\n");
                  //printf("tabfl[%d] %f\n", i, tabfl[i]);
                  //printf("tabfl[%d] %f\n", i+1, tabfl[i+1]);
                  if(operand[i] == '+'){
                        result = result + tabfl[i] + tabfl[i+1];
                  } else if (operand[i] == '-'){
                        result = result + tabfl[i] - tabfl[i+1];
                  }
```

```
            } else {
                  if(operand[i] == '+'){
                        result = result + tabfl[i+1];
                  } else if(operand[i] == '-'){
                        result = result - tabfl[i+1];
                  }
            }
            //printf("result %d %f\n", i, result);
      }
   }

   *res = result;
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "lib/pda.c"
#include "lib/stack.c"

int main()
{
   // Stack S;
   char input[1000];

   // fgets(input, 1000, stdin);
   printf("Ketik EXIT untuk keluar program\n\n");
   do
   {
      scanf("%s",input);
      if (PDA(input))
      {
            float result = proc_string(input);
            if(result != result)
            {
                  printf("Math Error\n");
            }
            else
            {
                  printf("-> ");
                  printf("%.2f\n\n", result);
            }
      }
      else if (strcmp(input,"EXIT") != 0)
```

```
        {
                printf("Syntax Error\n\n");
        }
    }
    while (strcmp(input,"EXIT") != 0);
}
```

Contoh Masukan dan Keluaran