Aufgabe 27

Unter Moodle finden Sie die Eingabefiles ‚uebung27.l' für den Scannergenerator lex und ‚uebung27.y' für den Parsergenerator bison, die noch verändert werden sollen. Achten Sie darauf, dass das im Vorspann mit uebungs26_tab.h bezeichnete Header-file vom Parsergenerator bison geschrieben wird. Die erforderlichen binaries für die Anwendung des Parsergenerators bison können Sie für Windows unter https://sourceforge.net/projects/winflexbison/
finden.
Das Generieren des Scanners aus uebung26.l funktioniert so wie in den Aufgaben 7 und 8. Für das Generieren des Parsers mit win_bison rufen Sie einfach

```
win_bison -dtv uebung26.y
```

auf.
Anschließend bitte sowohl die entstandenen c-Dateien:
uebung26.tab.c
lex.yy.c
als auch das von bison generierte header-file:
uebung26.tab.h
in das Projekt einbinden und übersetzen.
In der von win_bison erstellten Datei uebung26.output sehen Sie die Parsertabelle für den LR-Parser nebst einigen anderen Informationen.
Nun zur Aufgabe:
Folgende Grammatik ist gegeben:

```
S: A B C;
A: a;
B: b B;
B: b;
C: c;
```

Die entstehende Sprache is $L(G)=\{ab^nc|n>=1\}$. Ihr Parser soll nach der Eingabe eines Worts, die Anzahl der auftretenden b's bestimmen.
Geben Sie die Grammatik an der geeigneten Stelle in uebung26.y ein und führen Sie Aktionen aus, die die Anzahl der b's im Eingabewort bestimmen.

```
%{                          // flex-Eingabe fuer Uebung 27
#include <stdio.h>
#include <stdlib.h>
#include "uebung26.tab.h" // von yacc erstelltes header-file
%}
%%
"s" {return s;}
"r" {return r;}
"l" {return l;}
"v" {return v;}
"z" {return z;}
"e" {return e;}
"\n" {return nl;}
%%
yywrap()
{
 return 0;
};
```

Aufgabe 28

Ein Roboter kann mit Hilfe folgender Befehle gesteuert werden:
s – Start
v – 1 Schritt voraus
z – 1 Schritt zurück
r – Drehung um 90° nach rechts
l – Drehung um 90° nach links
e – Ende der Eingabe

Schreiben Sie mit flex und bison ein Programm, das eine beliebige Befehlsequenz einliest und die Koordinaten ausgibt, die der Roboter nach dieser Schrittfolge einnimmt. Der Roboter startet im Koordinatenursprung (0,0) und ist in x-Richtung ausgerichtet.

Beispieleingaben:
`s v v r v l v e` ergibt die Endkoordinaten (3/-1)
`s l v v r v l v e` ergibt die Endkoordinaten (1/3)

```
/* grammar.y - yacc -Eingabe für Übung 28*/
%{
#include <stdio.h>
#include <ctype.h>
#include <malloc.h>
int x=0, y=0,vx=1,vy=0,tmp;
%}
%token s r l v z e nl
%%
S: E S ;
S: ;
E: A e nl {printf("Roboter steht an Position %i,%i \n",x,y);};
A: A r {tmp=vx;vx=vy;vy=-tmp;};
A: A l {tmp=vx;vx=-vy;vy=tmp;};
A: A v {x+=vx;y+=vy;};
A: A z {x-=vx;y-=vy;};
A: s {/*x=0; y=0;vx=1;vy=0;*/};
%%
int yyerror(char*mesg){
printf("Syntaxfehler");
return 0;
}

main(){
yyparse();
}
```

Aufgabe 29

Erstellen Sie mittels der tools flex und bison, die auch in Aufgabenblatt 11 zum Einsatz kamen, einen „Taschenrechner“, der geklammerte Ausdrücke mit +,-,* und / verarbeiten kann und das Ergebnis eines eingegebenen Ausdruckes auf dem Bildschirm ausgibt.
Eingabebeispiel: 3.4+5*7
Ausgabe:38.4

Aufgabe 30

Erweitern Sie den Taschenrechner aus Aufgabe so, dass Variablenzuweisungen verarbeitet werden können. Hierzu müssen Sie eine Symboltabelle erstellen (z.B. in Form eines arrays), die Namen und Wert der Variablen enthält.
Flex-Eingabe:

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <string.h>
#include "uebung27.tab.h"

char csymbole[100][16];
int max_ID=0;
int lookup(char* );
%}
ziffer [0-9]
Buch [a-zA-Z]
Zahl ({ziffer}+".")?{ziffer}+
%%
"+" {return ADD;}
"-" {return SUB;}
"*" {return MUL;}
"/" {return DIV;}
"(" {return LKL;}
")" {return RKL;}
"\n" {return NL;}
"=" {return ASSIGN;}
{Zahl} {yylval.dval = atof(yytext); return (ZAHL);}
{Buch}({Buch}|{ziffer})*  {yylval.ival=lookup(yytext); return (ID);}
%%
int lookup(char* s)
{
       int index=-1,i;
       for (i=0;i<max_ID;i++)
              if (strcmp(s,csymbole[i])==0) index=i;
       if (index==-1)
       {
              index=max_ID;
              strcpy(csymbole[index],s);
              max_ID++;
       };
       return index;
}

yywrap()
{
 return 0;
};
```

Bison-Eingabe:

```
%{
#include <stdio.h>
#include <ctype.h>
#include <malloc.h>
#include <math.h>
extern char *yytext;
double dsymbole[100];
%}
%union { double dval; int ival; }
%token <dval> ZAHL
%token <ival> ADD SUB
%token <ival> MUL DIV
%token <ival> LKL RKL NL ASSIGN ID
%type <dval> zeile expr statm
%left ADD
%left SUB
%left DIV
%left MUL
%right ASSIGN
%%
zeile  :      zeile NL                    {printf("leere Eingabe\n");}
              |      zeile statm NL              {printf("Ergebnis: %f\n", $2);}
              |      error {printf("Fehler beim Lesen von: %s\n", yytext);}
              |      statm NL                    {printf("Ergebnis: %f\n", $1);};
statm  :      expr   { $$ = $1; }
              |      ID ASSIGN expr { $$ = $3; dsymbole[$1] = $3; } ;
expr   :      expr ADD expr          { $$ = $1 + $3; }
              |      expr SUB expr   { $$ = $1 - $3; }
              |      expr MUL expr   { $$ = $1 * $3; }
              |      expr DIV expr   { $$ = $1 / $3; }
              |      LKL expr RKL    { $$ = $2; }
              |      ZAHL            { $$ = $1; }
              |      ID              { $$ = dsymbole[$1]; } ;
%%
int yyerror(char*mesg){
printf("Syntaxfehler beim Lesen von %s\n",yytext);
return 0;
}
main(){
yyparse();}
```

Aufgabe 31

Wandeln Sie den Taschenrechner so ab, dass der eingegebene Ausdruck in Postfix-Notation ausgegeben wird. Dies kann beispielsweise bereits als Zwischencode für eine Stack-Maschine verwendet werden.
Eingabebeispiel: 3.4+5*7
Ausgabebeispiel Zwischencode Stackmaschine:

```
PUSH 3.4
PUSH 5
PUSH 7
MULT
ADD
```

```
%%
zeile  :      zeile NL {printf("leere Eingabe\n");}
              |      zeile statm NL       {printf("Ergebnis: %f\n", $2);}
              |      error {printf("Fehler beim Lesen von: %s\n", yytext);}
              |      statm NL             {printf("Ergebnis: %f\n", $1);};
statm  :      expr                        { $$ = $1; printf("\n");}
              ;
expr   :      expr ADD expr               { $$ = $1 + $3; printf("ADD \n");}
              |      expr SUB expr        { $$ = $1 - $3; printf("SUB \n");}
              |      expr MUL expr        { $$ = $1 * $3; printf("MULT \n");}
              |      expr DIV expr        { $$ = $1 / $3; printf("DIV \n");}
              |      LKL expr RKL         { $$ = $2; }
              |      ZAHL                 { $$ = $1; printf("PUSH %f \n",$1);}
              ;
%%
```

Aufgabe 32

Schreiben Sie einen Taschenrechner mit flex+bison, der unär codierte ganze Zahlen erkennt, in Dezimalzahlen übersetzt und die Summe dieser Zahlen als Dezimalzahl ausgibt:
Eingabe: 11111+111
Ausgabe: 5+3=8

Lex-Datei:

```
ziffer [1]
Zahl {ziffer}+
%%
"+" {return ADD;}

"\n" {return NL;}
{Zahl} {yylval.ival = unaer(yytext); return (ZAHL);} /* Übergabe des Wertes in
yylval.ival*/

%%
int unaer(char* s)
{
       return strlen(s); /*Wert der unaeren Zahl entspricht seiner Laenge*/
}
```

Yacc-Datei:

```
%%
zeile  :      zeile NL             {printf("leere Eingabe\n");}
              |      zeile statm NL       {printf("\n");}
              |      error {printf("Fehler beim Lesen von: %s\n", yytext);}
              |      statm NL {printf("\n");}
              ;
statm  :      expr          { $$ = $1;}
              ;
expr   :      expr ADD expr { $$ = $1 + $3;printf("%i + %i = %i",$1,$3,$$);}
              |      ZAHL   { $$ = $1;}

%%
```