

UE20CS353-CD ASSIGNMENT-2

NAME	SRN	CLASS & SECTION
VIJAY J	PES2UG20CS815	6 - J

ABSTRACT SYNTAX TREE(AST):

lexer.l

```
lexer.l
~/PESU/6th Sem/CD/Assignment/Assignment-2/AST
Save

1 %f
2 //NAME: Vijay J
3 //SRN: PES2UG20CS815
4 //SECTION: J
5 #define YYSTYPE char*
6 #include <unistd.h>
7 #include "y.tab.h"
8 #include <stdio.h>
9 extern void yyerror(const char *); // declare the error handling function
10 %}
11
12 /* Regular definitions */
13 digit    [0-9]
14 letter   [a-zA-Z]
15 id       {letter}{(letter){digit}}*
16 digits   {digit}+
17 opFraction    (\.{digits})?
18 opExponent    ([Ee][+-]?{digits})?
19 number        {digits}{opFraction}{opExponent}
20 %option yylineno
21
22 %%
23 \/\(.*) ; // ignore comments
24 [\t\n] ; // ignore whitespaces
25 "("      {return *yytext;}
26 ")"      {return *yytext;}
27 "."      {return *yytext;}
28 ","      {return *yytext;}
29 "*"      {return *yytext;}
30 "+"      {return *yytext;}
31 ";"      {return *yytext;}
32 "-"      {return *yytext;}
33 "/"      {return *yytext;}
34 "="      {return *yytext;}
35 ">"      {return *yytext;}
36 "<"      {return *yytext;}
37 "if"      {return IF;}
38 "else"     {return ELSE;}
39 "{"        {return *yytext;}
40 "}"        {return *yytext;}
41 {number}   {
42             yylval = strdup(yytext); //stores the value of the number to
be used later for symbol table insertion
43             return T_NUM;
44         }
45 {id}       {
46             yylval = strdup(yytext); //stores the
identifier to be used later for symbol table insertion
47             return T_ID;
48         }
49 .         {} // anything else => ignore
50 %%
```

parser.y

```
Open [icon] *parsery ~/PESU/6th Sem/CD/Assignment/Assignment-2/AST Save [menu] - [max] [close]
1 %{
2 //NAME: Vijay J
3 //SRN: PES2UG20CS815
4 //SECTION: J
5     #include "abstract_syntax_tree.c"
6     #include <stdio.h>
7     #include <stdlib.h>
8     #include <string.h>
9     void yyerror(char*
10 s);
11 error handling function
12 int
13 yylex();
14 declare the function performing lexical analysis
15 extern int
16 yylineno;
17 track the line number
18 %}
19
20
21
22 %token <text> T_ID T_NUM IF ELSE
23
24 %type <text> RELOP
```

```
Open [icon] *parsery ~/PESU/6th Sem/CD/Assignment/Assignment-2/AST Save [menu] - [max] [close]
25 %type <exp_node> E T F START ASSGN S C SEQ
26
27 /* specify start symbol */
28 %start START
29
30
31
32 %%
33 START : SEQ {
34     display_exp_tree($1);
35     printf("\nValid syntax\n");
36     YYACCEPT;
37 };
38
39 SEQ : S SEQ { $$ = init_exp_node("seq", $1, $2, NULL); }
40     | S { $$ = $1; }
41 ;
42
43 S : IF '(' C ')' '{' SEQ '}' {
44     $$ = init_exp_node(strdup("if"), $3, $6, NULL);
45 }
46     | IF '(' C ')' '{' SEQ '}' ELSE '{' SEQ '}' {
47     $$ = init_exp_node(strdup("if-else"), $3, $6, $10);
48 }
49     | ASSGN { $$ = $1; }
50 ;
51
52 C : F RELOP F {
53     $$=init_exp_node(strdup($2), $1, $3, NULL);
54 }
55 ;
56
```

```
Open  [icon] *parsery
~/PESU/6th Sem/CD/Assignment/Assignment-2/AST Save [icon] [icon] [icon]

57 RELOP : '<' { $$ = "<"; }
58         | '>' { $$ = ">"; }
59         | '>=' { $$ = ">="; }
60         | '<=' { $$ = "<="; }
61         | '=' { $$ = "="; }
62         | '!=' { $$ = "!="; }
63 ;
64
65
66
67 /* Grammar for assignment */
68 ASSGN : T_ID '=' E ';' {
69     $$ =
70     init_exp_node(strdup("=") ,init_exp_node(strdup($1) ,NULL ,NULL ,NULL) ,$3 ,NULL);
71     ;
72
73 /* Expression Grammar */
74 E : E '+' T {
75     $$ = init_exp_node(strdup("+") ,$1 ,
76     $3 ,NULL);
77     | E '-' T {
78     $$ = init_exp_node(strdup("-") ,$1 ,
79     $3 ,NULL);
80     | T { $$ = $1; }
81     ;
82
83
84 T : T '*' F {
```

```
Open  [icon] *parsery
~/PESU/6th Sem/CD/Assignment/Assignment-2/AST Save [icon] [icon] [icon]

85     $$ = init_exp_node(strdup("*") ,$1 ,
86     $3 ,NULL);
87     | T '/' F {
88     $$ = init_exp_node(strdup("/") ,$1 ,
89     $3 ,NULL);
90     | F { $$ = $1; }
91     ;
92
93 F : '(' E ')' { $$ = $2; }
94     | T_ID {
95     $$ = init_exp_node(strdup($1) ,NULL ,NULL ,NULL);
96     }
97     | T_NUM {
98     $$ =
99     init_exp_node(strdup($1) ,NULL ,NULL ,NULL);
100    }
101    ;
102 %%
103
104 /* error handling function */
105 void yyerror(char* s)
106 {
107     printf("Error :%s at %d \n",s,yylineno);
108 }
109
110
111 int yywrap() {
112     return 1;
113 }
```

```

113 }
114
115 /* main function - calls the yyparse() function which will in turn drive yylex() as
well */
116 int main(int argc, char* argv[])
117 {
118     printf("Preorder:\n");
119     yyparse();
120     return 0;
121 }

```

abstract_syntax_tree.h

```

Open  abstract_syntax_tree.h  Save  -  x
~/PESU/6th Sem/CD/Assignment/Assignment-2/AST

1 typedef struct expression_node
2 {
3     /*      needs 3 members
4             i) pointer to the left child
5             ii) pointer to the right child
6             iii) string to store the value of the node
7     */
8     struct expression_node* left;
9     struct expression_node* middle;
10    struct expression_node* right;
11    char* val;
12 }
13 expression_node;
14
15 expression_node* init_exp_node(char* val, expression_node* left, expression_node*
middle, expression_node* right);
16 void display_exp_tree(expression_node* exp_node);
17

```

abstract_syntax_tree.c

```

Open  abstract_syntax_tree.c  Save  -  x
~/PESU/6th Sem/CD/Assignment/Assignment-2/AST

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include "abstract_syntax_tree.h"
5
6 expression_node* init_exp_node(char* val, expression_node* left, expression_node*
middle, expression_node* right)
7 {
8     // function to allocate memory for an AST node and set the left and right
children of the nodes
9     expression_node* node = (expression_node*)malloc(sizeof(expression_node));
10    node->left = left;
11    node->middle = middle;
12    node->right = right;
13    node->val = val;
14 }
15
16 void helper(expression_node* exp_node, int first)
17 {
18     if(exp_node != NULL) {
19         if(first) {
20             printf("%s", exp_node->val);
21             first = 0;
22         }
23         else {
24             printf(", %s", exp_node->val);
25         }
26         helper(exp_node->left, first);
27         helper(exp_node->middle, first);
28         helper(exp_node->right, first);
29     }
30 }

```

```

32 void display_exp_tree(expression_node* exp_node)
33 {
34     // traversing the AST in preorder and displaying the nodes
35     int first = 1;
36     helper(exp_node, first);
37 }

```

run.sh

```

Open  run.sh  Save  -  x
~/PESU/6th Sem/CD/Assignment/Assignment-2/AST

1 #!/bin/bash
2
3 lex lexer.l
4 yacc -d parser.y
5 gcc -g y.tab.c lex.yy.c
6
7 rm lex.yy.c
8 rm y.tab.c
9 rm y.tab.h
10

```

input code and output screenshot test_input1.c

```

Open  test_input1.c  Save  -  x
~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815

1 if(a > b)
2 {
3     a = a + 1;
4     b = b - 1;
5 }

```

Output

```

~/P/6/C/A/A/P/AST

yoyo@zaemon in ~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815/AST via C v12.2.1-gcc took 2ms
λ ./run.sh

yoyo@zaemon in ~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815/AST via C v12.2.1-gcc took 384ms
λ ./a.out < test_input1.c

Preorder:
if,>,a,b,seq,=,a,+,a,1,=,b,-,b,1
Valid syntax

```

test_input2.c

```
test_input2.c
~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815
Save

1 if(a > b)
2 {
3     a = a + 1;
4     b = b - 1;
5 }
6 else
7 {
8     a = a - 1;
9     b = b - 1;
10 }
```

Output

```
~/P/6/C/A/A/P/AST
yoyo@zaemon in ~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815/AST via C v12.2.1-gcc took 4ms
λ ./run.sh

yoyo@zaemon in ~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815/AST via C v12.2.1-gcc took 376ms
λ ./a.out < test_input2.c
Preorder:
if-else,>,a,b,seq,=,a,+,a,1,=,b,-,b,1,seq,=,a,-,a,1,=,b,-,b,1
Valid syntax
```

test_input3.c

```
test_input3.c
~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815
Save

1 if(a > b)
2 {
3     a = a + 1;
4     b = b - 1;
5 }
6 else
7 {
8     a = a - 1;
9     b = b - 1;
10 if (b < 0)
11 {
12     b = b + 1;
13 }
14 else
15 {
16     b = 0;
17 }
18 }
```

Output

```
~/P/6/C/A/A/P/AST
yoyo@zaemon in ~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815/AST via C v12.2.1-gcc took 2ms
λ ./run.sh

yoyo@zaemon in ~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815/AST via C v12.2.1-gcc took 362ms
λ ./a.out < test_input3.c
Preorder:
if-else,>,a,b,seq,=,a,+,a,1,=,b,-,b,1,seq,=,a,-,a,1,seq,=,b,-,b,1,if-else,<,b,0,=,b,+,b,1,=,b,0
Valid syntax
```

INTERMEDIATE CODE GENERATION(ICG):

lexer.l

```
lexer.l
~/PESU/6th Sem/CD/Assignment/Assignment-2/ICG
Save

1 %{
2 //NAME: Vijay J
3 //SRN: PES2UG20CS815
4 //SECTION: J
5 #define YYSTYPE char*
6 #include <unistd.h>
7 #include "y.tab.h"
8 #include <stdio.h>
9 extern void yyerror(const char *); // declare the error handling function
10 %}
11
12 /* Regular definitions */
13 digit [0-9]
14 letter [a-zA-Z]
15 id {letter}({letter}|{digit})*
16 digits {digit}+
17 opFraction (\.{digits})?
18 opExponent ([Ee][+-]?{digits})?
19 number {digits}{opFraction}{opExponent}
20 %option yylineno
21
22 %%
23 \/\/(.*) ; // ignore comments
24 [\t\n] ; // ignore whitespaces
25 "<=" {return LTEQ;}
26 ">=" {return GTEQ;}
27 "==" {return EQQ;}
28 "!=" {return NEQ;}
29 "{" {return OC;}
30 "}" {return CC;}
31 "(" {return *yytext;}
32 ")" {return *yytext;}
33 "." {return *yytext;}
34 "," {return *yytext;}
35 "*" {return *yytext;}
36 "+" {return *yytext;}
37 ";" {return *yytext;}
38 "-" {return *yytext;}
39 "/" {return *yytext;}
40 "=" {return *yytext;}
41 ">" {return GT;}
42 "<" {return LT;}
43 {number} {
44     yylval = strdup(yytext); //stores the value of the number to
be used later for symbol table insertion
45     return T_NUM;
46 }
47 "if" {return T_IF;}
48 "else" {return T_ELSE;}
49 {id} {
50     yylval = strdup(yytext); //stores the
identifier to be used later for symbol table insertion
51     return T_ID;
52 }
53 . {} // anything else => ignore
54 %%
```

parser.y

```
1 %{
2 //NAME: Vijay J
3 //SRN: PES2UG20CS815
4 //SECTION: J
5     #include "quad_generation.c"
6     #include <stdio.h>
7     #include <stdlib.h>
8     #include <string.h>
9
10     #define YYSTYPE char*
11
12     void yyerror(char*
13 s);
14 error handling function
15
16 int
17 yylex();
18 declare the function performing lexical analysis
19
20 extern int
21 yylineno;
22 track the line number
23
24 FILE* icg_quad_file;
25 int temp_no = 1;
26 int label_no=1;
27 %}
28
29 %token T_ID T_NUM T_IF T_ELSE GTEQ LTEQ EQQ NEQ GT LT OC CC
30
31 /* specify start symbol */
32 %start START
```

```
33
34
35 %nonassoc T_IF
36 %nonassoc T_ELSE
37
38 %%
39
40 START : S {
41     printf("_____\\n");
42     printf("Valid syntax\\n");
43     YYACCEPT;
44 }
45
46
47 /* Grammar for assignment */
48 ASSGN : T_ID '=' E {
49     quad_code_gen($1, $3, "=", " ");
50 }
51
52 ;
53
54 /* Expression Grammar */
55 E : E '+' T {
56     $$ = new_temp();
57     quad_code_gen($$, $1, "+", $3);
58 }
59
60 | E '-' T {
61     $$ = new_temp();
62     quad_code_gen($$, $1, "-", $3);
63 }
64
65 | T {
66     $$ = $1;
67 }
```



```

58 | T
59 ;
60
61
62 T : T '*' F {
63     $$ = new_temp();
64     quad_code_gen($$, $1, "*", $3);
65 }
66 | T '/' F {
67
68
69     $$ = new_temp();
70     quad_code_gen($$, $1, "/", $3);
71 }
72 | F
73 ;
74
75 F : '(' E ')' {
76
77     $$=strdup($2);
78 }
79 | T_ID {
80     $$=strdup($1);
81 }
82 | T_NUM {
83     $$=strdup($1);
84 }
85 ;
86
87 S : T_IF '(' C ')' OC S CC {quad_code_gen($3,"","Label","");} S
88 | T_IF '(' C ')' OC S CC {

```

```

89     $2 = new_label();
90     quad_code_gen($2,"","goto","");
91     quad_code_gen($3,"","Label","");} T_ELSE OC S CC
92 {quad_code_gen($2,"","Label","");}S
93 | ASSIGN ';' S
94 | '{' S '}'
95 |
96 ;
97 C : E rel E { $$ = new_temp();
98     quad_code_gen($$, $1, $2, $3);
99     $1 = new_label();
100     quad_code_gen($1,$$,"if","");
101     $$ = new_label();
102     quad_code_gen($$, "", "goto", "");
103     quad_code_gen($1, "", "Label", "");
104
105 };
106
107 rel : GT {strcpy($$, ">");}
108 | LT {strcpy($$, "<");}
109 | LTEQ {strcpy($$, "<=");}
110 | GTEQ {strcpy($$, ">=");}
111 | EQQ {strcpy($$, "=");}
112 | NEQ {strcpy($$, "!=");}
113 ;
114
115 %%
116
117
118 /* error handling function */

```

```

119 void yyerror(char* s)
120 {
121     printf("Error :%s at %d \n",s,yylineno);
122 }
123
124 int yywrap() {
125     return 1;
126 }
127
128 /* main function - calls the yyparse() function which will in turn drive yylex() as
well */
129 int main(int argc, char* argv[])
130 {
131
132     printf("Generated Intermediate Code \n");
133     printf("_____ \n");
134     printf("| %-10s | %-10s | %-10s | %-10s | \n", "op", "arg1", "arg2",
"result");
135     printf("_____ \n");
136     yyparse();
137     return 0;
138 }

```

quad_generation.h

```

quad_generation.h
~/PESU/6th Sem/CD/Assignment/Assignment-2/ICG
1 extern FILE* icg_quad_file; //pointer to the output file
2 extern int temp_no; //variable to keep track of current
temporary count
3 extern int label_no;
4
5 void quad_code_gen(char* a, char* b, char* op, char* c);
6 char* new_temp();

```

quad_generation.c

```

quad_generation.c
~/PESU/6th Sem/CD/Assignment/Assignment-2/ICG
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include "quad_generation.h"
5
6
7 void quad_code_gen(char* a, char* b, char* op, char* c)
8 {
9     printf("| %-10s | %-10s | %-10s | %-10s | \n", op, b, c, a);
10 }
11
12 char* new_temp()
13 {
14     char* tempNew = (char*)malloc(sizeof(char)*4);
15     sprintf(tempNew, "t%d", temp_no);
16     temp_no++;
17     return tempNew;
18 }
19
20 char* new_label()
21 {
22     char* label = (char*)malloc(sizeof(char)*4);
23     sprintf(label, "L%d", label_no);
24     label_no++;
25     return label;
26 }

```

run.sh

```
run.sh
~/PESU/6th Sem/CD/Assignment/Assignment-2/ICG
1 lex lexer.l
2 yacc -d parser.y
3 gcc -g y.tab.c lex.yy.c
4
5 rm lex.yy.c
6 rm y.tab.c
7 rm y.tab.h
8
```

input code and output screenshot test_input1.c

```
test_input1.c
~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815/ICG
1 if(a>b)
2 {
3     a=a+1;
4     b=b-1;
5 }
6 a=a+b*a;
```

Output

```
~/P/6/C/A/A/P/ICG
...yoyyo@zaemon in ~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815/ICG via C v12.2.1-gcc took 3ms
└─ ./run.sh

...yo@zaemon in ~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815/ICG via C v12.2.1-gcc took 388ms
└─ ./a.out < test_input1.c
Generated Intermediate Code
-----
| op      | arg1  | arg2  | result |
-----
| >       | a     | b     | t1      |
| if      | t1    |       | L1      |
| goto    |       |       | L2      |
| Label   |       |       | L1      |
| +       | a     | 1     | t2      |
| =       | t2    |       | a       |
| -       | b     | 1     | t3      |
| =       | t3    |       | b       |
| Label   |       |       | L2      |
| *       | b     | a     | t4      |
| +       | a     | t4    | t5      |
| =       | t5    |       | a       |
-----
Valid syntax
```

test_input2.c

```
test_input2.c
~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815/ICG
Save

1 if(a > b)
2 {
3     a = a + 1;
4     b = b - 1;
5 }
6 else
7 {
8     a = a - 1;
9     b = b - 1;
10 }
```

Output

```
~/P/6/C/A/A/P/ICG
yoyo@zaemon in ~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815/ICG via C v12.2.1-gcc took 51ms
λ ./run.sh

yoyo@zaemon in ~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815/ICG via C v12.2.1-gcc took 1s
λ ./a.out < test_input2.c
Generated Intermediate Code
-----
| op      | arg1  | arg2  | result |
-----
| >       | a     | b     | t1      |
| if      | t1    |       | L1      |
| goto    |       |       | L2      |
| Label   |       |       | L1      |
| +       | a     | 1     | t2      |
| =       | t2    |       | a       |
| -       | b     | 1     | t3      |
| =       | t3    |       | b       |
| goto    |       |       | L3      |
| Label   |       |       | L2      |
| -       | a     | 1     | t4      |
| =       | t4    |       | a       |
| -       | b     | 1     | t5      |
| =       | t5    |       | b       |
| Label   |       |       | L3      |
-----
Valid syntax
```