VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

B.Tech. Winter Semester 2023-24
School Of Computer Science and Engineering
(SCOPE)

Digital Assignment - VI

Compiler Design Lab

Apurva Mishra, 22BCE2791

12 Novemeber 2024

Note: This is mirror of Assessment 6 submitted on 12 November, 2024 on Moodle

1. Question

Problem 1.1 .
To write a YACC program to recognize strings of { b^naa | n>=5}.

main.l

```
%{
#include <stdlib.h>
#include <stdio.h>
void yyerror(char *);
#include "y.tab.h"
%}
%%
((bbbbb)(b*)(aa)) { return PATTERN; }
[ \t] ;
. yyerror("invalid character");
%%
int yywrap(void) {
return 1;
}
```

main.y

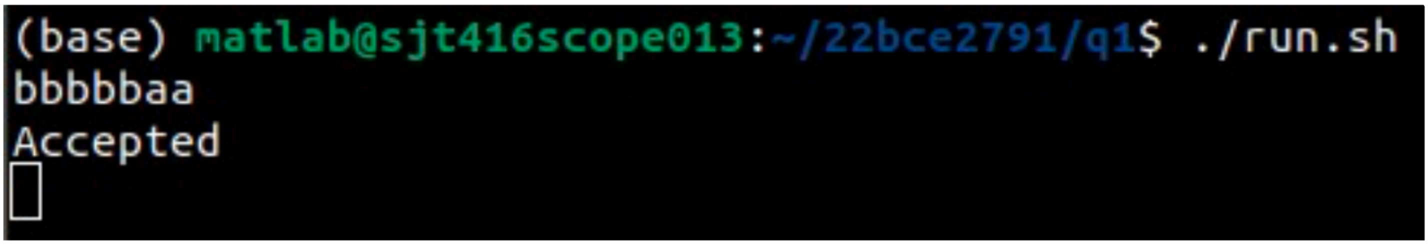
```
%token PATTERN
%{
#include <stdio.h>
void yyerror(char *);
int yylex(void);
%}
%%
program:
program expr '\n'
|
;
expr:
PATTERN {$$ = 1; printf("Accepted");}
|
;
%%
void yyerror(char *s) {
fprintf(stderr, "%s\n", s);
}
int main(void) {
```

```
yyparse();
return 0;
}
```

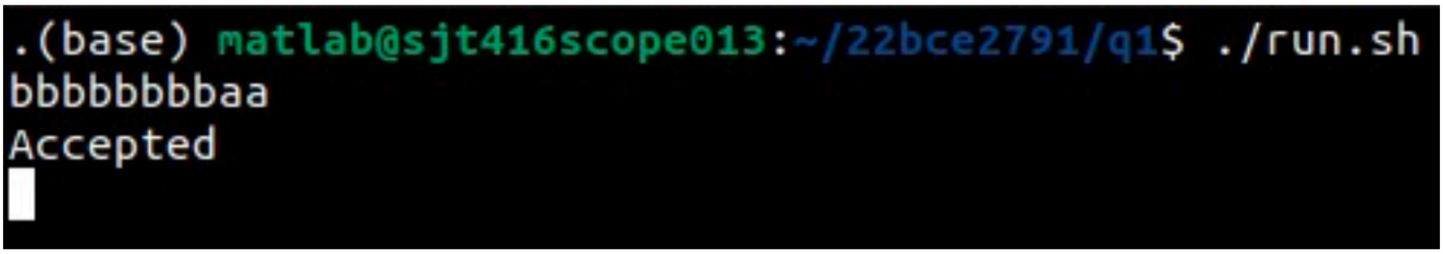
run.sh

```bash
#!/bin/bash

lex main.l
yacc -d main.y
gcc lex.yy.c y.tab.c -o main
./main
```

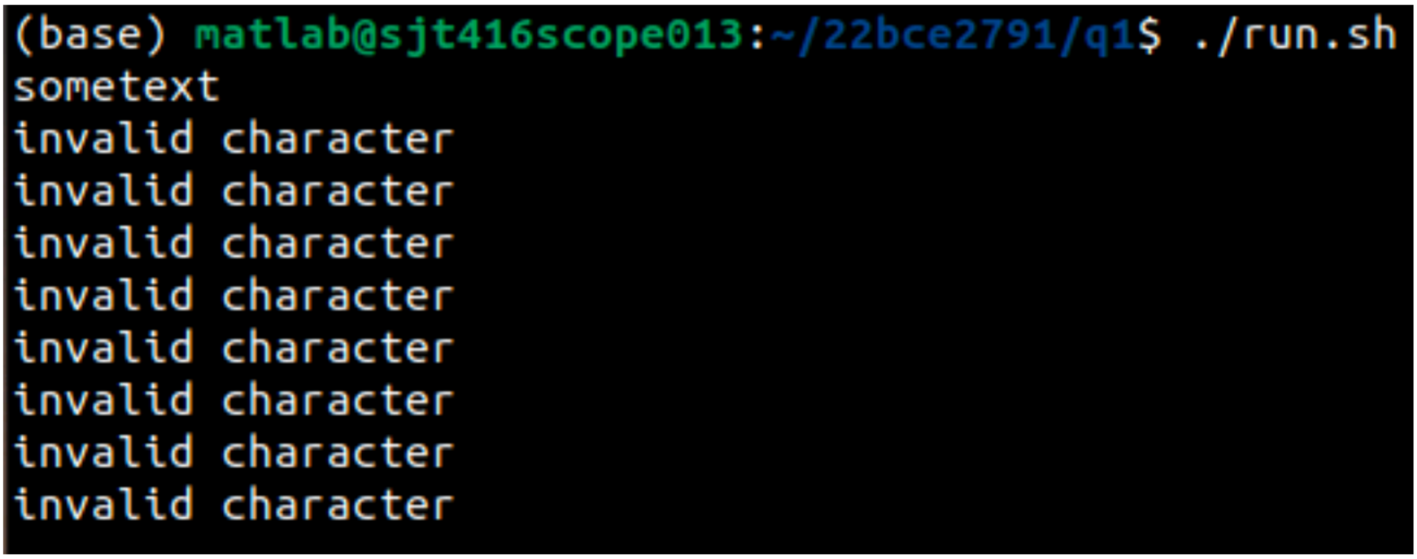
Output



```
(base) matlab@sjt416scope013:~/22bce2791/q1$ ./run.sh
bbbbbaa
Accepted
```



```
.(base) matlab@sjt416scope013:~/22bce2791/q1$ ./run.sh
bbbbbbbbaa
Accepted
```



```
(base) matlab@sjt416scope013:~/22bce2791/q1$ ./run.sh
sometext
invalid character
invalid character
invalid character
invalid character
invalid character
invalid character
invalid character
invalid character
```

2. Question

Problem 2.1 .
Write a C program to implement Code optimization technique.

main.l

```
%{
#include <stdlib.h>
#include <stdio.h>
void yyerror(char *);
#include "y.tab.h"
%}
%%
[0-9]+ {
yylval = atoi(yytext);
return INTEGER;
}
[a-z] {
yylval = *yytext;
return VARIABLE;
}
("jmp") { return JMP; }
[-+/*;=] { return *yytext; }
[ \t\n]+ ;
. yyerror("invalid character");
%%
int yywrap(void) {
return 1;
}
```

main.y

```
%token INTEGER VARIABLE JMP
%left '+' '-'
%left '*' '/'
%{
#include "core.h"
void yyerror(char *);
int yylex(void);
int sym[26];
%}
%%
program:
function { ; }
;
```

```
function:
function expr { ;}
| /* NULL */
;
expr:
| VARIABLE '=' INTEGER '+' INTEGER ';' { new_node($1, $3, $5); print_line();}
| VARIABLE '=' VARIABLE '+' INTEGER ';' { new_node($1, $3, $5); print_line();}
| VARIABLE '=' INTEGER '+' VARIABLE ';' { new_node($1, $3, $5); print_line();}
| VARIABLE '=' VARIABLE '+' VARIABLE ';' { new_node($1, $3, $5); print_line();}
| jump ';' { ; }
;
jump:
JMP INTEGER { ;}
;
%%
void yyerror(char *s) {
fprintf(stderr,"%s\n", s);
}
int main(void) {
yyparse();
return 0;
}
```

core.h

```
#include <stdio.h>

struct eval {
    int var;
    int var1;
    int var2;
};

struct eval global[10];
int i = 0;

int skip_count = 0;

void new_node(int var, int var1, int var2) {
    global[i].var = var;
    global[i].var1 = var1;
    global[i].var2 = var2;

    i += 1;
}

void print_line() {
    if (skip_count > 0) {
```

```
    skip_count -= 1;
    return;
  }

  int k = i - 1;

  int prev = -1;
  for (int j = 0; j < k; j++) {
    if (global[j].var1 == global[k].var1 && global[j].var2 ==
global[k].var2) {
      prev = global[j].var;
      break;
    }
  }

  if (prev != -1) {
    printf("%c = %c; | Common Subexpressions Elimination\n", global[k].var,
prev);
    return;
  }

  if (global[k].var1 >= 'a' && global[k].var2 >= 'a') {
    printf("%c = %c + %c; | No optimization\n", global[k].var,
global[k].var1, global[k].var2);
  } else if (global[k].var1 >= 'a' && global[k].var2 < 'a') {
    printf("%c = %c + %d; | No optimization\n", global[k].var,
global[k].var1, global[k].var2);
  } else if (global[k].var1 < 'a' && global[k].var2 >= 'a') {
    printf("%c = %d + %c; | No optimization\n", global[k].var,
global[k].var1, global[k].var2);
  } else {
    printf("%c = %d; | Constant Folding\n", global[k].var, global[k].var1
+ global[k].var2);
  }
}
```

run.sh

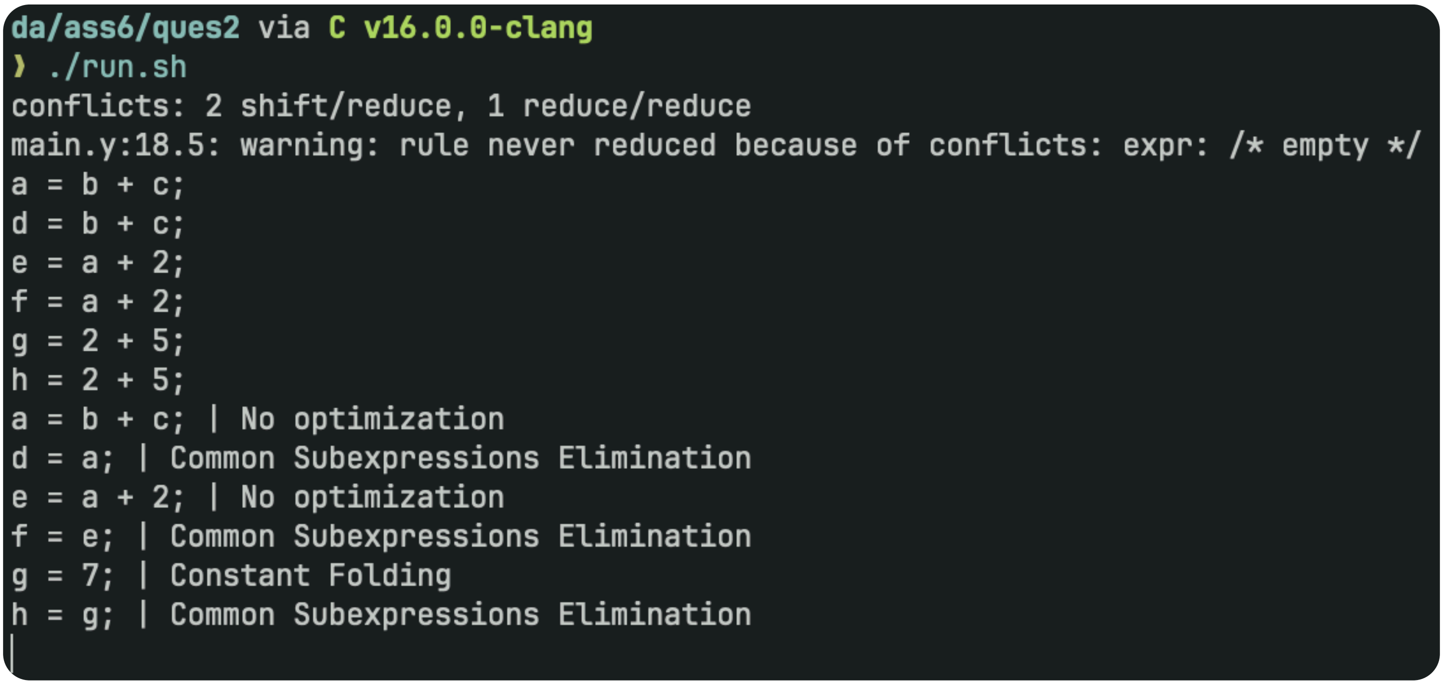
```
#!/bin/bash

lex main.l
yacc -d main.y
gcc lex.yy.c y.tab.c -o main
./main
```

Output

The given code applies following two optimisations:

- Common Subexpressions Elimination

- Constant Folding



```
da/ass6/ques2 via C v16.0.0-clang
❯ ./run.sh
conflicts: 2 shift/reduce, 1 reduce/reduce
main.y:18.5: warning: rule never reduced because of conflicts: expr: /* empty */
a = b + c;
d = b + c;
e = a + 2;
f = a + 2;
g = 2 + 5;
h = 2 + 5;
a = b + c; | No optimization
d = a; | Common Subexpressions Elimination
e = a + 2; | No optimization
f = e; | Common Subexpressions Elimination
g = 7; | Constant Folding
h = g; | Common Subexpressions Elimination
```