EXPERIMENT NO. 1

PROGRAM

```
%{
#include <stdio.h>
%}
DIGIT    [0-9]
LETTER          [a-zA-Z]
ID       {LETTER}({LETTER}|{DIGIT})*
NUM      {DIGIT}+(\.{DIGIT}+)?
WS       [ \t\n]
%%
#.*      {printf("PREPROCESSOR DIRECTIVE : %s\n",yytext);}
(if|else|while|int|return) {printf("KEYWORD: %s\n", yytext);}
{ID}     { printf("ID: %s\n", yytext); }
{NUM}           { printf("NUM: %s\n", yytext); }
[+\-*/=]        { printf("OPERATOR: %s\n", yytext); }
[(){},"] { printf("CHARACTER: %s\n", yytext); }
\/\/[^\n]* ;
{WS}    ;
.       { printf("INVALID CHARACTER: %s\n", yytext); }
%%
int main() {
        yylex();
        return 0;
}
```

OUTPUT

```
exam1@student-Veriton-S2680G:~/Experiment 1$ lex exp1.l
exam1@student-Veriton-S2680G:~/Experiment 1$ gcc lex.yy.c -o exp1 -ll
exam1@student-Veriton-S2680G:~/Experiment 1$ ./exp1 < cprog.c
PREPROCESSOR DIRECTIVE : #include <stdio.h>
KEYWORD: int
ID: main
CHARACTER: (
CHARACTER: )
CHARACTER: {
KEYWORD: int
ID: sum
CHARACTER: ,
ID: i
CHARACTER: ,
ID: j
INVALID CHARACTER: ;
ID: i
OPERATOR: =
NUM: 9
INVALID CHARACTER: ;
ID: j
OPERATOR: =
NUM: 1
INVALID CHARACTER: ;
ID: sum
OPERATOR: =
ID: i
OPERATOR: +
ID: j
INVALID CHARACTER: ;
ID: printf
CHARACTER: (
```

EXPERIMENT NO. 2

PROGRAM

```
%{
#include <stdio.h>
int line_count = 0;
int word_count = 0;
int char_count = 0;
%}

%%
\n      { line_count++; }
[a-zA-Z]+   { word_count++; char_count += yyleng; }
.       { char_count++; }
%%

int main() {
        yylex();
        printf("Lines: %d\nWords: %d\nCharacters: %d\n", line_count, word_count,
char_count);
        return 0;
}
```

OUTPUT

```
exam1@student-Veriton-S2680G:~/Experiment 2$ lex exp2.l
exam1@student-Veriton-S2680G:~/Experiment 2$ gcc lex.yy.c -o exp2 -ll
exam1@student-Veriton-S2680G:~/Experiment 2$ ./exp2 < Text.txt
Lines: 4
Words: 3
Characters: 12
```

EXPERIMENT NO. 3

PROGRAM

```
%{
#include <stdio.h>
%}

%%
abc     { printf("ABC"); }
.       { putchar(yytext[0]); }
%%

int main() {
        yylex();
        return 0;
}
```

OUTPUT

```
exam1@student-Veriton-S2680G:~/Experiment 3$ ./exp3 < Text.txt
Hi ABC
exam1@student-Veriton-S2680G:~/Experiment 3$
```

EXPERIMENT NO. 4

PROGRAM

```
%{
#include <stdio.h>
int vowel_count = 0;
int consonant_count = 0;
%}

%%
[aAeEiIoOuU]  { vowel_count++; }
[a-zA-Z]      { consonant_count++; }
.             ; // Ignore other characters
%%

int main() {
        printf("Enter a String : ");
        yylex();
        printf("Vowels: %d\nConsonants: %d\n", vowel_count, consonant_count);
        return 0;
}
```

OUTPUT

```
exam1@student-Veriton-S2680G:~/Experiment 4$ lex exp4.l
exam1@student-Veriton-S2680G:~/Experiment 4$ gcc -lfl lex.yy.c
exam1@student-Veriton-S2680G:~/Experiment 4$ ./a.out
Enter a String : johnsisgood

Vowels: 4
Consonants: 7
```

EXPERIMENT NO. 5

PROGRAM

YACC PROGRAM

```
%{
#include <stdio.h>
int valid = 1;
%}
%token num id op
%%
start : id '=' s ';'
s       : id x
        | num x
        | '-' num x
        | '(' s ')' x
        ;

x       : op s
        | '-' s
        |
        ;

%%
int yyerror() {
        valid = 0;
        printf("\nInvalid expression!\n");
        return 0;
}
int main() {
        printf("\nEnter the expression:\n");
        yyparse();
        if (valid) {
        printf("\nValid expression!\n");
        }
}
```

LEX PROGRAM

```
%{
        #include "y.tab.h"
%}
%%
[a-zA-Z_][a-zA-Z_0-9]* return id;
[0-9]+(\.[0-9]*)?       return num;
[+/*]           return op;
.               return yytext[0];
\n              return 0;
%%
int yywrap()
{
return 1;
}
```

OUTPUT

```
johns@johns-Inspiron-3593:~/Desktop$ yacc -d e.y
johns@johns-Inspiron-3593:~/Desktop$ flex e.l
johns@johns-Inspiron-3593:~/Desktop$ gcc lex.yy.c y.tab.c -w
johns@johns-Inspiron-3593:~/Desktop$ ./a.out

Enter the expression:
b=a+c*(e+f);

Valid expression!
```

EXPERIMENT NO. 6

PROGRAM

YACC PROGRAM

```
%{
        #include<stdio.h>
        int valid=1;
%}
%token num id op
%%
start :id '=' s ; | s ;
s :     id x
        | num x
        | '-' num x
        | '(' s ')' x
        ;
x :     op s
        | '-' s
        |
        ;
%%
int yyerror()
{
        valid=0;
        printf("\nInvalid expression!\n");
        return 0;
}
int main()
{
        printf("\nEnter the expression:\n");
        yyparse();
        if(valid)
        {
        printf("\nValid expression!\n");
        }
}
```

LEX PROGRAM

```
%{
        #include "y.tab.h"
%}
%%
[a-zA-Z_][a-zA-Z_0-9]* return id;
[0-9]+(\.[0-9]*)?       return num;
[+/*]           return op;
.               return yytext[0];
\n              return 0;
%%
int yywrap()
{
return 1;
}
```

OUTPUT

```
johns@johns-Inspiron-3593:~/Desktop/Compiler $ yacc -d exp6.y
johns@johns-Inspiron-3593:~/Desktop/Compiler $ flex exp6.l
johns@johns-Inspiron-3593:~/Desktop/Compiler $ gcc lex.yy.c y.tab.c -w
johns@johns-Inspiron-3593:~/Desktop/Compiler $ ./a.out

Enter the expression:
myIdentifier123

Valid expression!
johns@johns-Inspiron-3593:~/Desktop/Compiler $
```

EXPERIMENT NO. 7

PROGRAM

YACC PROGRAM

```
 %{
        #include<stdio.h>
        int flag=0;
%}
%token NUMBER
%left '+' '-'
%left '*' '/' '%'
%left '(' ')'
%%
ArithmeticExpression: E{
        printf("\nResult=%d\n",$$);
        return 0;
        };
E:E'+'E {$$=$1+$3;}
 |E'-'E {$$=$1-$3;}
 |E'*'E {$$=$1*$3;}
 |E'/'E {$$=$1/$3;}
 |E'%'E {$$=$1%$3;}
 |'('E')' {$$=$2;}
 | NUMBER {$$=$1;}
;
%%
void main(){
  printf("\nEnter Any Arithmetic Expression:\n");
  yyparse();
 if(flag==0)
  printf("\n\n");
}
void yyerror(){
  printf("\nInvalid\n\n");
  flag=1;
}
```

LEX PROGRAM

```
%{
/* Definition section */
#include<stdio.h>
#include "y.tab.h"
extern int yylval;
%}
/* Rule Section */
%%
[0-9]+ {
        yylval=atoi(yytext);
        return NUMBER;
   }
[\t] ;
[\n] return 0;
. return yytext[0];
%%
int yywrap()
{
return 1;
}
```

OUTPUT

```
johns@johns-Inspiron-3593:~/Desktop/Compiler /Experiment 7$ yacc -d exp7.y
johns@johns-Inspiron-3593:~/Desktop/Compiler /Experiment 7$ flex exp7.l
johns@johns-Inspiron-3593:~/Desktop/Compiler /Experiment 7$ gcc lex.yy.c y.tab.c -w
johns@johns-Inspiron-3593:~/Desktop/Compiler /Experiment 7$ ./a.out

Enter Any Arithmetic Expression:
4*2+3

Result=11
```

EXPERIMENT NO. 8

PROGRAM

```
#include <stdio.h>
#include <string.h>
char result[20][20], copy[3], states[20][20];
void add_state(char a[3], int i) {
        strcpy(result[i], a);
}
void display(int n) {
        int k = 0;
        printf("\nEpsilon closure of %s = { ", copy);
        while (k < n) {
        printf("%s ", result[k]);
        k++;
        }
        printf("}\n");
}
int main() {
        FILE *INPUT;
        INPUT = fopen("input.dat", "r");
        if (INPUT == NULL) {
        fprintf(stderr, "Unable to open the file.\n");
        return 1;
        }
        char state[3];
        int end, i = 0, n, k = 0;
        char state1[3], input[3], state2[3];
        printf("Enter the no of states: ");
        scanf("%d", &n);
        printf("Enter the states\n");
        for (k = 0; k < n; k++) {
        scanf("%s", states[k]);
        }
        for (k = 0; k < n; k++) {
        i = 0;
```

```
        strcpy(state, states[k]);
        strcpy(copy, state);
        add_state(state, i++);
        while (1) {
        end = fscanf(INPUT, "%s%s%s", state1, input, state2);
        if (end == EOF) {
        break;
        }
        if (strcmp(state, state1) == 0) {
        if (strcmp(input, "e") == 0) {
                add_state(state2, i++);
                strcpy(state, state2);
        }
        }
        }
        display(i);
        rewind(INPUT);
        }

        fclose(INPUT);

        return 0;
}
```

Input.dat

```
q0 e q1
q0 e q2
q1 a q3
q2 b q3
q3 c q4
```

OUTPUT

```
Enter the no of states: 4
Enter the states
q0
q1
q2
q3

Epsilon closure of q0 = { q0 q1 }

Epsilon closure of q1 = { q1 }

Epsilon closure of q2 = { q2 }

Epsilon closure of q3 = { q3 }
```

EXPERIMENT NO. 9

PROGRAM

```
#include <stdio.h>
#include <string.h>
#define SUCCESS 1
#define FAILED 0
int E(), Edash(), T(), Tdash(), F();
const char *cursor;
char string[64];
int main() {
        puts("Enter the string");
        // scanf("%s", string);
        sscanf(" ----- ", "%s", string);  // Read input from the user: e.g., i+(i+i)*i
        cursor = string;
        puts("");
        puts("Input          Action");
        puts("--------------------------------");
        if (E() && *cursor == '\0') {
        puts("--------------------------------");
        puts("String is successfully parsed");
        return 0;
        } else {
        puts("--------------------------------");
        puts("Error in parsing String");
        return 1;
        }
}
int E() {
        printf("%-16s E -> T E'\n", cursor);
        if (T()) {
        if (Edash())
        return SUCCESS;
        else
        return FAILED;
        } else
        return FAILED;
}
int Edash() {
        if (*cursor == '+') {
        printf("%-16s E' -> + T E'\n", cursor);
        cursor++;
        if (T()) {
        if (Edash())
```

```
		return SUCCESS;
		else
		return FAILED;
		} else
		return FAILED;
		} else {
		printf("%-16s E' -> $\n", cursor);
		return SUCCESS;
		}
}
int T() {
		printf("%-16s T -> F T'\n", cursor);
		if (F()) {
		if (Tdash())
		return SUCCESS;
		else
		return FAILED;
		} else
		return FAILED;
}
int Tdash() {
		if (*cursor == '*') {
		printf("%-16s T' -> * F T'\n", cursor);
		cursor++;
		if (F()) {
		if (Tdash())
		return SUCCESS;
		else
		return FAILED;
		} else
		return FAILED;
		} else {
		printf("%-16s T' -> $\n", cursor);
		return SUCCESS;
		}
}
int F() {
		if (*cursor == '(') {
		printf("%-16s F -> ( E )\n", cursor);
		cursor++;
		if (E()) {
		if (*cursor == ')') {
		cursor++;
		return SUCCESS;
		} else
		return FAILED;
```

```
        } else
        return FAILED;
        } else if (*cursor == 'i') {
        cursor++;
        printf("%-16s F -> i\n", cursor);
        return SUCCESS;
        } else
        return FAILED;
}
```

OUTPUT

```
Enter the string
(i*i)+i

Input          Action
--------------------------------
(i*i)+i          E -> T E'
(i*i)+i          T -> F T'
(i*i)+i          F -> ( E )
i*i)+i           E -> T E'
i*i)+i           T -> F T'
*i)+i            F -> i
*i)+i            T' -> * F T'
)+i              F -> i
)+i              T' -> $
)+i              E' -> $
+i               T' -> $
+i               E' -> + T E'
i                T -> F T'
                 F -> i
                 T' -> $
                 E' -> $
--------------------------------
String is successfully parsed
```

EXPERIMENT NO. 10

PROGRAM

```
#include<stdio.h>
#include<string.h>
void main(){
char stack[20],ip[20],opt[10][10][1],ter[10];
int i,j,k,n,top=0,col,row;

for(i=0;i<10;i++)
{
 stack[i]=NULL;
 ip[i]=NULL;
 for(j=0;j<10;j++)
 {
 opt[i][j][1]=NULL;
 }
}
printf("Enter the no.of terminals :\n");
scanf("%d",&n);
printf("\nEnter the terminals :\n");
scanf("%s",&ter);
printf("\nEnter the table values :\n");
for(i=0;i<n;i++)
{
 for(j=0;j<n;j++)
 {
 printf("Enter the value for %c %c:",ter[i],ter[j]);
 scanf("%s",opt[i][j]);
 }
}
printf("\n**** OPERATOR PRECEDENCE TABLE ****\n");
for(i=0;i<n;i++)
{
printf("\t%c",ter[i]);
}
```

```
printf("\n");
for(i=0;i<n;i++){printf("\n%c",ter[i]);
for(j=0;j<n;j++){printf("\t%c",opt[i][j][0]);}}
stack[top]='$';
printf("\nEnter the input string:");
scanf("%s",ip);
i=0;
printf("\nSTACK\t\t\tINPUT STRING\t\t\tACTION\n");
printf("\n%s\t\t\t%s\t\t\t",stack,ip);
while(i<=strlen(ip))
{
for(k=0;k<n;k++)
{
if(stack[top]==ter[k])
col=k;
if(ip[i]==ter[k])
row=k;
}
if((stack[top]=='$')&&(ip[i]=='$')){
printf("String is accepted\n");
break;}
else if((opt[col][row][0]=='<') ||(opt[col][row][0]=='='))
{ stack[++top]=opt[col][row][0];
stack[++top]=ip[i];
 printf("Shift %c",ip[i]);
 i++;
  }
else{
 if(opt[col][row][0]=='>')
{
while(stack[top]!='<'){--top;}
top=top-1;
printf("Reduce");
}
else
{
printf("\nString is not accepted");
```

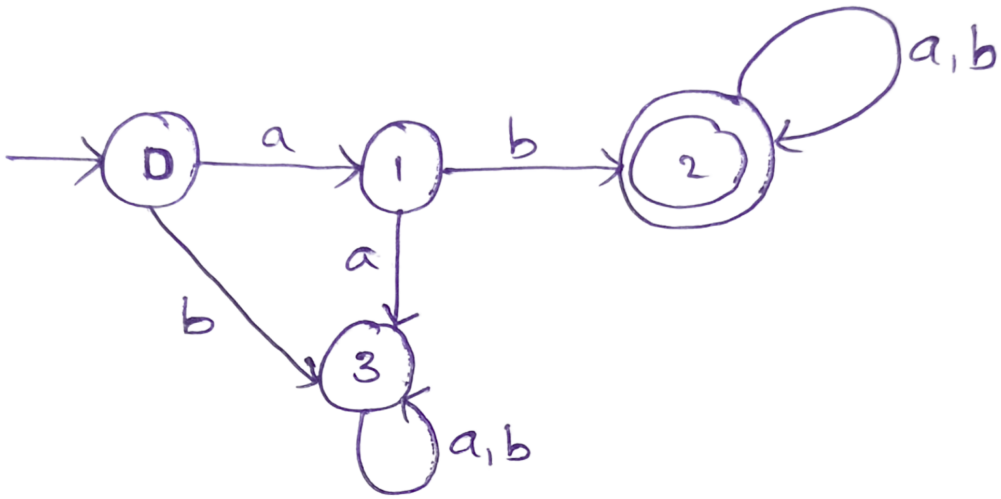
```
break;
}
}
printf("\n");
for(k=0;k<=top;k++)
{
printf("%c",stack[k]);
}
printf("\t\t\t");
for(k=i;k<strlen(ip);k++){
printf("%c",ip[k]);
}
printf("\t\t\t");
}
}
```

OUTPUT

```
Enter the no.of terminals :
4

Enter the terminals :
+*i$

Enter the table values :
Enter the value for + +:>
Enter the value for + *:<
Enter the value for + i:<
Enter the value for + $:>
Enter the value for * +:>
Enter the value for * *:>
Enter the value for * i:<
Enter the value for * $:>
Enter the value for i +:>
Enter the value for i *:>
Enter the value for i i:_
Enter the value for i $:>
Enter the value for $ +:<
Enter the value for $ *:<
Enter the value for $ i:<
Enter the value for $ $:-

**** OPERATOR PRECEDENCE TABLE ****
        +       *       i       $

+       >       <       <       >
*       >       >       <       >
i       >       >       _       >
$       <       <       <       -
Enter the input string:i+i*i$

STACK                   INPUT STRING                    ACTION

$                       i+i*i$                  Shift i
$<i                     +i*i$                   Reduce
$                       +i*i$                   Shift +
$<+                     i*i$                    Shift i
$<+<i                   *i$                     Reduce
$<+                     *i$                     Shift *
$<+<*                   i$                      Shift i
$<+<*<i                 $                       Reduce
$<+<*                   $                       Reduce
$<+                     $                       Reduce
$                       $                       String is accepted
```

EXPERIMENT NO. 11 A

DFA



PROGRAM

```
#include <stdio.h>
#include <string.h>

int main() {
        char input[100];

        printf("Enter a string: ");
        scanf("%s", input);

        int currentState = 0;
        int len = strlen(input);

        for (int i = 0; i < len; i++) {
        switch (currentState) {
        case 0:
        if (input[i] == 'a') {
                currentState = 1;
        } else {
```

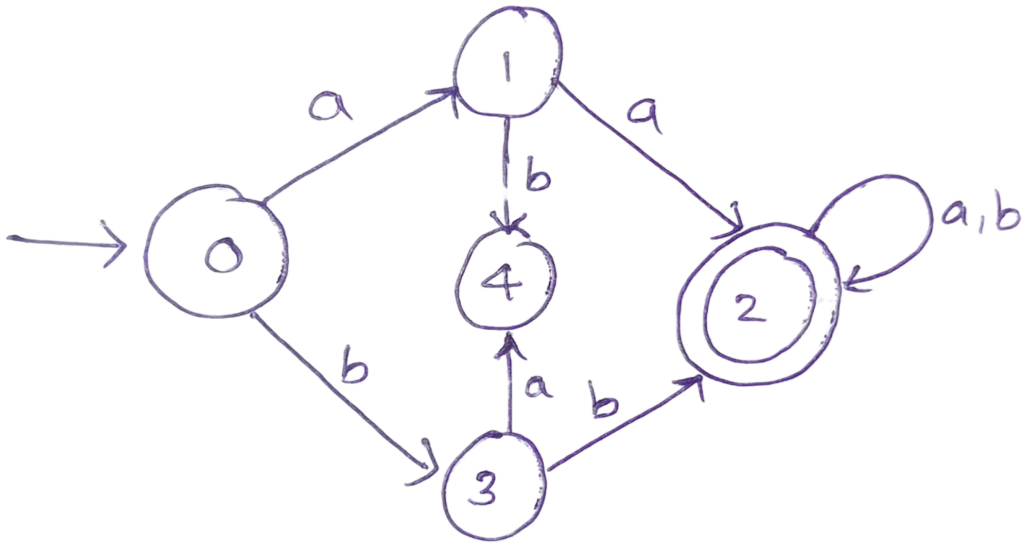
```
            currentState = 0;
        }
        break;
        case 1:
        if (input[i] == 'b') {
            currentState = 2;
        } else {
            currentState = 0;
        }
        break;
        case 2:
        currentState = 2; // Stay in the accepting state for any input
        break;
        default:
        currentState = 0; // Invalid state
        }
        }

        if (currentState == 2) {
        printf("Accepted: The string starts with 'ab'\n");
        } else {
        printf("Rejected: The string does not start with 'ab'\n");
        }

        return 0;
}
```

OUTPUT

```
Enter a string: abaa
Accepted: The string starts with 'ab'
johns@johns-Inspiron-3593:~/Desktop/Compiler /Experiment 11/11 A$ ./a.out
Enter a string: aaba
Rejected: The string does not start with 'ab'
```

EXPERIMENT NO. 11 B

DFA



PROGRAM

```
#include <stdio.h>
#include <string.h>
int main() {
        char input[100];
        printf("Enter a string: ");
        scanf("%s", input);
        int currentState = 0;
        int len = strlen(input);
        for (int i = 0; i < len; i++) {
        switch (currentState) {
        case 0:
        if (input[i] == 'a') {
                currentState = 1;
        } else if (input[i] == 'b') {
                currentState = 2;
        } else {
                currentState = 4;
        }
```

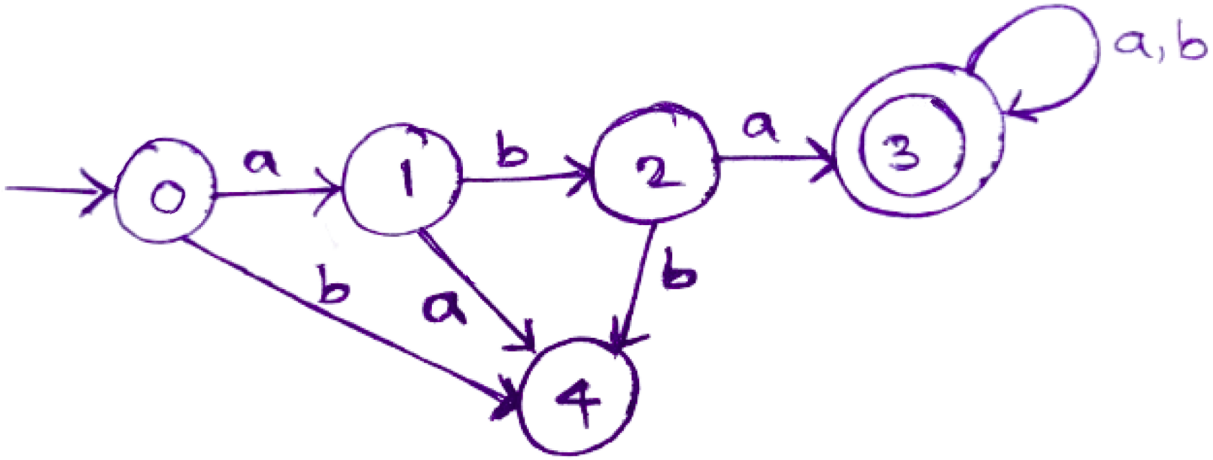
```
        break;
        case 1:
        if (input[i] == 'a') {
                currentState = 3;
        } else {
                currentState = 4;
        }
        break;
        case 2:
        if (input[i] == 'b') {
                currentState = 3;
        } else {
                currentState = 4;
        }
        break;
        case 3:
        case 4:
        break; // Stay in the accepting state for any input
        default:
        currentState = 5; // Invalid state
        }
        }
        if (currentState == 3) {
        printf("Accepted: The string starts with 'aa' or 'bb'\n");
        } else {
        printf("Rejected: The string does not start with 'aa' or 'bb'\n");
        }
        return 0;
}
```

OUTPUT

```
Enter a string: aabaa
Accepted: The string starts with 'aa' or 'bb'
johns@johns-Inspiron-3593:~/Desktop/Compiler /Experiment 11/11 B$ ./a.out
Enter a string: abaabb
Rejected: The string does not start with 'aa' or 'bb'
```

EXPERIMENT NO. 11 C

DFA



PROGRAM

```
#include <stdio.h>
#include <string.h>
int main() {
        char input[100];
        printf("Enter a string: ");
        scanf("%s", input);
        int currentState = 0;
        int len = strlen(input);
        for (int i = 0; i < len; i++) {
        switch (currentState) {
        case 0:
        if (input[i] == 'a') {
                currentState = 1;
        } else {
                currentState = 0;
        }
        break;
        case 1:
        if (input[i] == 'b') {
```

```
            currentState = 2;
        } else if (input[i] == 'a') {
            currentState = 1;
        } else {
            currentState = 0;
        }
        break;
        case 2:
        if (input[i] == 'a') {
            currentState = 3;
        } else {
            currentState = 0;
        }
        break;
        case 3:
        break;
        default:
        currentState = 0; // Invalid state
        }
        }
        if (currentState == 3) {
        printf("Accepted: The string starts with 'aba'\n");
        } else {
        printf("Rejected: The string does not start with 'aba'\n");
        }
        return 0;
}
```

OUTPUT

```
Enter a string: aabbaaa
Rejected: The string does not start with 'aba'
johns@johns-Inspiron-3593:~/Desktop/Compiler /Experiment 11/11 C$ ./a.out
Enter a string: abaab
Accepted: The string starts with 'aba'
```

EXPERIMENT NO. 12 A

PROGRAM

WITHOUT LOOPING

```
#include<stdio.h>
#include <time.h>
int main(void)
{
   clock_t start, end;
   double cpu_time_used;
   start = clock();
         for (int i=0; i<5; i++)
         printf("Hello\n"); //print hello 5 times
    end = clock();
    cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
    printf("Time taken: %f\n", cpu_time_used);
         return 0;
}
```

OUTPUT

```
johns@johns-Inspiron-3593:~/Desktop/Compiler /Experiment 12/A$ ./1
Hello
Hello
Hello
Hello
Hello
Time taken: 0.000054
```

WITH UNLOOPING

```
#include<stdio.h>
#include <time.h>
int main(void){
   clock_t start, end;
   double cpu_time_used;
   start = clock();
           printf("Hello\n");
           printf("Hello\n");
           printf("Hello\n");
           printf("Hello\n");
           printf("Hello\n");
    end = clock();
    cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
    printf("Time taken: %f\n", cpu_time_used);
         return 0;
}
```

OUTPUT

```
johns@johns-Inspiron-3593:~/Desktop/Compiler /Experiment 12/A$ ./2
Hello
Hello
Hello
Hello
Hello
Time taken: 0.000049
```

EXPERIMENT NO. 12 B

PROGRAM

WITHOUT CONSTANT PROPAGATION

```
#include <stdio.h>
#include <time.h>
int square(int a) {
        return a * a;
}
int add(int a, int b) {
        return a + b;
}
int calculate(int a, int b) {
        int e = square(a);
        int f = add(b, b);
        return e + f;
}
int main() {
   clock_t start,end;
   double cpu_time_used;
        start = clock();
        int a = 3;
        int b = 4;
        int result = calculate(a, b);
        printf("Result: %d\n", result);
        end = clock();
   cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
        printf("Time taken: %f\n", cpu_time_used);
        return 0;
}
```

OUTPUT

```
johns@johns-Inspiron-3593:~/Desktop/Compiler /Experiment 12/B$ ./1
Result: 17
Time taken: 0.000072
```

WITH CONSTANT PROPAGATION

```
#include <stdio.h>
#include <time.h>
int main() {
        clock_t start, end;
        double cpu_time_used;
        start = clock();
        printf("Result: %d\n", (3*3)+(4+4));
        end = clock();
        cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
        printf("Time taken: %f\n", cpu_time_used);
        return 0;
}
```

OUTPUT

```
johns@johns-Inspiron-3593:~/Desktop/Compiler /Experiment 12/B$ ./2
Result: 17
Time taken: 0.000062
```

EXPERIMENT NO. 13

PROGRAM

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int i = 1, j = 0, no = 0, tmpch = 90;
char str[100], left[15], right[15];
void findopr();
void explore();
void fleft(int);
void fright(int);
struct exp
{
        int pos;
        char op;
} k[15];
void main()
{
        printf("\t\tINTERMEDIATE CODE GENERATION\n\n");
        printf("Enter the Expression :");
        scanf("%s", str);
        printf("The intermediate code:\n");
        findopr();
        explore();
}
void findopr()
{
        for (i = 0; str[i] != '\0'; i++)
        if (str[i] == ':')
        {
        k[j].pos = i;
        k[j++].op = ':';
        }
        for (i = 0; str[i] != '\0'; i++)
        if (str[i] == '/')
```

```
        {
        k[j].pos = i;
        k[j++].op = '/';
        }
        for (i = 0; str[i] != '\0'; i++)
        if (str[i] == '*')
        {
        k[j].pos = i;
        k[j++].op = '*';
        }
        for (i = 0; str[i] != '\0'; i++)
        if (str[i] == '+')
        {
        k[j].pos = i;
        k[j++].op = '+';
        }
        for (i = 0; str[i] != '\0'; i++)
        if (str[i] == '-')
        {
        k[j].pos = i;
        k[j++].op = '-';
        }
}
void explore()
{
        i = 1;
        while (k[i].op != '\0')
        {
        fleft(k[i].pos);
        fright(k[i].pos);
        str[k[i].pos] = tmpch--;
        printf("\t%c := %s%c%s\t\t", str[k[i].pos], left, k[i].op, right);
        printf("\n");
        i++;
        }
        fright(-1);
        if (no == 0)
```

```
        {
        fleft(strlen(str));
        printf("\t%s := %s", right, left);
        exit(0);
        }
        printf("\t%s := %c", right, str[k[--i].pos]);
}
void fleft(int x)
{
        int w = 0, flag = 0;
        x--;
        while (x != -1 && str[x] != '+' && str[x] != '*' && str[x] != '=' && str[x] != '\0' && str[x] != '-'
&& str[x] != '/' && str[x] != ':')
        {
        if (str[x] != '$' && flag == 0)
        {
        left[w++] = str[x];
        left[w] = '\0';
        str[x] = '$';
        flag = 1;
        }
        x--;
        }
}
void fright(int x)
{
        int w = 0, flag = 0;
        x++;
        while (x != -1 && str[x] != '+' && str[x] != '*' && str[x] != '\0' && str[x] != '=' && str[x] != ':'
&& str[x] != '-' && str[x] != '/')
        {
        if (str[x] != '$' && flag == 0)
        {
        right[w++] = str[x];
        right[w] = '\0';
        str[x] = '$';
        flag = 1;
```

```
        }
        x++;
        }
}
```

OUTPUT

```
                INTERMEDIATE CODE GENERATION

Enter the Expression :w:=a*b+c/d-e/f+g*h
The intermediate code:
        Z := c/d
        Y := e/f
        X := a*b
        W := g*h
        V := X+Z
        U := Y+W
        T := V-U
```

EXPERIMENT NO. 14

PROGRAM

```
#include <stdio.h>
#include <string.h>
void main()
{
        char icode[10][30], str[20], opr[10];
        int i = 0;
        printf("\n Enter the set of intermediate code(terminated by exit):\n");
        do
        {
        scanf("%s", icode[i]);
        } while (strcmp(icode[i++], "exit") != 0);
        printf("\n target code generation");
        printf("\n**********************");
        i = 0;
        do
        {
        strcpy(str, icode[i]);
        switch (str[3])
        {
        case '+':
        strcpy(opr, "ADD ");
        break;
        case '-':
        strcpy(opr, "SUB ");
        break;
        case '*':
        strcpy(opr, "MUL ");
        break;
        case '/':
        strcpy(opr, "DIV ");
        break;
        }
        printf("\n\tMov %c,R%d", str[2], i);
```

```
        printf("\n\t%s%c,R%d", opr, str[4], i);
        printf("\n\tMov R%d,%c", i, str[0]);
        } while (strcmp(icode[++i], "exit") != 0);
}
```

OUTPUT

```
 Enter the set of intermediate code(terminated by exit):
A=A*B
C=F*H
G=A*H
F=Q+W
T=Q-J
exit

 target code generation
************************
        Mov A,R0
        MUL B,R0
        Mov R0,A
        Mov F,R1
        MUL H,R1
        Mov R1,C
        Mov A,R2
        MUL H,R2
        Mov R2,G
        Mov Q,R3
        ADD W,R3
        Mov R3,F
        Mov Q,R4
        SUB J,R4
        Mov R4,T
```