Project Report

on

# Compiler for Layman Friendly Energy Conversion Program

Developed by

**Harsh Bhalala-IT007-19ITUOS088**

**Akshar Bhalodia-IT008-19ITUEF005**

**Bhargav Bhatiya-IT009-19ITUOS060**

Guided By

**Prof. Nikita P. Desai**

Dept. of Information Technology

**Department of Information Technology**

**Faculty of Technology**

**Dharmsinh Desai University**

**College Road, Nadiad-387001**

**2021-2022**

# CERTIFICATE

This is to certify that the project entitled “**Layman friendly Energy Conversion Program”** is a bonafide report of the work carried out by

1) Mr. Harsh Bhalala, Student ID No: 19ITUOS088
2) Mr. Akshar Bhalodia, Student ID No: 19ITUEF005
3) Mr. Bhargav Bhatiya, Student ID No: 19ITUOS060

of Department of Information Technology, semester VI, under the guidance and supervision for the award of the degree of Bachelor of Technology at Dharmsinh Desai University, Nadiad (Gujarat). They were involved in Project in subject of “**Language Translator**” during the academic year 2021-2022.

Prof. N. P. Desai
(Lab In charge)
Department of Information Technology,
Faculty of Technology,
Dharmsinh Desai University, Nadiad
Date:

Prof. (Dr.) V. K. Dabhi,
Head, Department of Information Technology,
Faculty of Technology,
Dharmsinh Desai University, Nadiad
Date:

# Introduction

## Project Details:

**Grammar Name:** Layman Friendly Energy Conversion Program.

**Valid Sentences in Language:**

1. 1 Joule means how many Calary?
2. 1 Joule/Second means how many watt?
3. 1 hp means how many killo watt?

**Keywords:**

Means

How Many

And

Convert

Into

**Operators:**

Joule

Calary

Watt

Hp

joule/second

Kilowat

**Digit:**

| | |
|---|---|
| [0-9]+ | int |
| [0-9]+(. [0-9]+) | float |

**Punctuation :**

| | | |
|---|---|---|
| Qmark | ? | q |
| NewLine | [\n] | n |
| Eos | . | e |
| Separator | , | s |
| WhiteSpace | [/t] | w |

# First & Follow

**Grammar:**

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | joule | means | how | many | calary | ? |
| DIGIT | OP | KW | KW | KW | OP | PUN |

**Non-terminals:**

S, E, KW, OP, KW, D, PUN, Q, G, T

**Terminals:**

Joule | Calary | watt | Hp | joule/second | kilowatt | means | how many | and | convert | into | int | float | ? | . | ,

S -> E KW

E -> D OP

KW -> KW KW | KW OP

OP -> OP KW | OP PUN

KW -> means | how many | and | convert | into

OP -> Joule | Calary | Watt | Hp | joule/second | Kilowatt

D -> int | float

PUN -> Q | G | T

Q -> ?

G -> .

T -> ,

**Output:**

**First:**

FIRST[S] = { int }

FIRST[E] = { int }

FIRST[KW] = {means, how, many, and, convert, into}

FIRST[D] = { int }

FIRST[OP] = {Joule, calary, watt, Hp, joule/second, kilowatt}

FIRST[PUN] = { ? . ,}

FIRST[Q] = { ? }

FIRST[G] = { . }

FIRST[T] = { , }

**Follow:**

FOLLOW[S] = {}

FOLLOW[E] = { means, how, many, and, convert, into }

FOLLOW[KW] = { means, how, many, and, convert, into, Joule, calary, watt, Hp, joule/second, kilowatt}

FOLLOW[D] = Joule

FOLLOW[OP] = { Joule, calary, watt, Hp, joule/second, kilowatt }

FOLLOW[PUN] = {$}

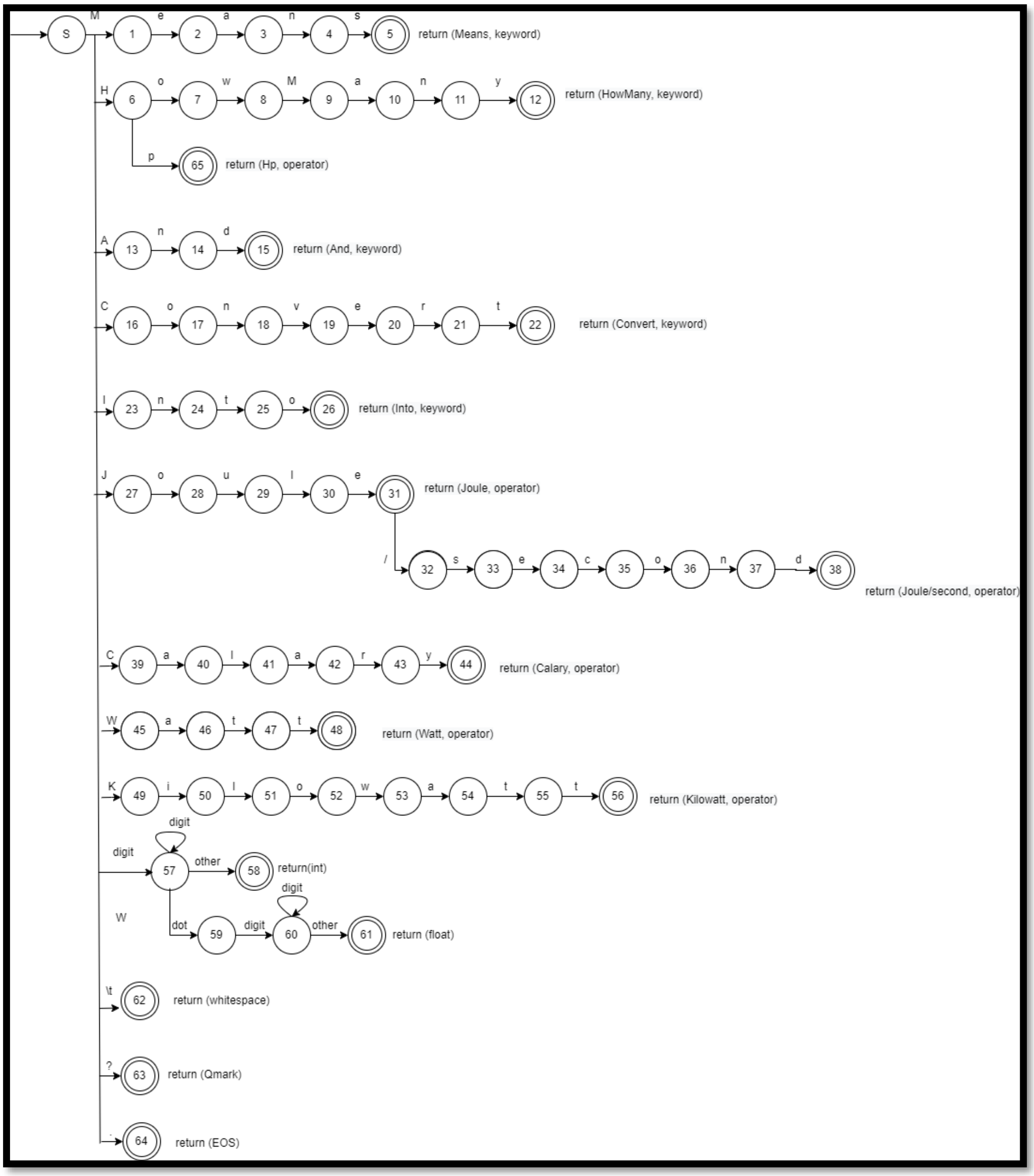
FOLLOW[Q] = {$}

FOLLOW[G] = {$}

FOLLOW[T] = {$}

# DFA & Algorithm

## DFA:



**Algorithm:**

```
lexer {
  int c = 0;
  bool f = false;
  int len = string.length();
  while not eof do {
    state = "S";
    while not eof do(c < len) {
      if (f) {
        f = false;
      }
      char ch = nextchar();
      switch (state) {
      case state of "S": {
        case state of 'M': state = "1";
        ch = nextchar();
        break;
        'H':
        state = "6";
        ch = nextchar();
        break;
        'A':
        state = "13";
        ch = nextchar();
        break;
        'C':
        state = "16";
        ch = nextchar();
        break;
```

```
'I':
state = "23";
ch = nextchar();
break;
'J':
state = "27";
ch = nextchar();
break;
'C':
state = "39";
ch = nextchar();
break;
'W':
state = "45";

ch = nextchar();
break;
'K':
state = "49";
ch = nextchar();
break;
[0 - 9]:
state = "57";
ch = nextchar();
f = true;
break;
'.':
state = "64";
ch = nextchar();
f = true;
```

```
  break;
  '\t':
  state = "62";
  ch = nextchar();
  break;
  '?':
  state = "63";
  ch = nextchar();
  break;
  Default: f = true;
  end
  case
}
case state of "1": {
  case state of 'e':
      state = "2";
  ch = nextchar();
  break;
}
case state of "2": {
  case state of 'a':
       state = "3";
  ch = nextchar();
  break;
}
case state of "3": {
  case state of 'n':
      state = "4";
  ch = nextchar();
  break;
```

```
}
case state of "4": {

  case state of 's':
      state = "5";
  ch = nextchar();
  break;
}
case state of "1": {
  case state of 'e':
      state = "2";
  ch = nextchar();
  break;
}
case state of "2": {
  case state of 'a':
      state = "3";
  ch = nextchar();
  break;
}
case state of "3": {
  case state of 'n':
      state = "4";
  ch = nextchar();
  break;
}
case state of "6": {
  case state of 'o':
      state = "7";
  ch = nextchar();
  break;
  case state of 'p':
      state = "65";
  ch = nextchar();
  break;
}
case state of "7": {
  case state of 'w':
      state = "8";
  ch = nextchar();
  break;
}
case state of "8": {
  case state of 'M':
      state = "9";
  ch = nextchar();
  break;
}
case state of "9": {
  case state of 'a':
      state = "10";
  ch = nextchar();
  break;
}
case state of "10": {
  case state of 'n':
      state = "11";
  ch = nextchar();
  break;
}
```

```
case state of "11": {
  case state of 'y':
      state = "12";
  ch = nextchar();
  break;
}
case state of "13": {
  case state of 'n':
      state = "14";
  ch = nextchar();
  break;
}
case state of "14": {
  case state of 'd':
      state = "15";
  ch = nextchar();
  break;
}
case state of "16": {
  case state of 'o':
      state = "17";
  ch = nextchar();
  break;
}
case state of "17": {
  case state of 'n':
      state = "18";
  ch = nextchar();
  break;
}
```

```
case state of "18": {
  case state of 'v':
      state = "19";
  ch = nextchar();
  break;
}
case state of "19": {
  case state of 'e':
      state = "20";
  ch = nextchar();
  break;
}
case state of "20": {
  case state of 'r':
      state = "21";
  ch = nextchar();
  break;
}
case state of "21": {
  case state of 't':
      state = "22";
  ch = nextchar();
  break;
}
case state of "23": {
  case state of 'n':
      state = "24";
  ch = nextchar();
  break;
}
```

```
case state of "24": {
  case state of 't':
      state = "25";
  ch = nextchar();
  break;
}
case state of "25": {
  case state of 'o':
      state = "26";
  ch = nextchar();
  break;
}
case state of "27": {
  case state of 'o':
      state = "28";
  ch = nextchar();
  break;
}
case state of "28": {
  case state of 'u':
      state = "29";
  ch = nextchar();
  break;
}
case state of "29": {
  case state of 'l':
      state = "30";
  ch = nextchar();
  break;
}
```

```
case state of "30": {
  case state of 'e':
      state = "31";
  ch = nextchar();
  break;
}
case state of "31": {
  case state of '/':
      state = "32";
  ch = nextchar();
  break;
}
case state of "32": {
  case state of 's':
      state = "33";
  ch = nextchar();
  break;
}
case state of "33": {
  case state of 'e':
      state = "34";
  ch = nextchar();
  break;
}
case state of "34": {
  case state of 'c':
      state = "35";
  ch = nextchar();
  break;
}
```

```
case state of "35": {
  case state of 'o':
      state = "36";
  ch = nextchar();
  break;
}
case state of "36": {
  case state of 'n':
      state = "37";
  ch = nextchar();
  break;
}
case state of "37": {
  case state of 'd':
      state = "38";
  ch = nextchar();
  break;
}
}
case state of "39": {
  case state of 'a': state = "40";
  ch = nextchar();
  break;
}
case state of "40": {
  case state of 'l': state = "41";
  ch = nextchar();
  break;
}
case state of "41": {
  case state of 'a': state = "42";
  ch = nextchar();
  break;
}
case state of "42": {
  case state of 'r': state = "43";
  ch = nextchar();
  break;
}
case state of "43": {
  case state of 'y': state = "44";
  ch = nextchar();
  break;
}

case state of "44": {
  case state of 'l': state = "45";
  ch = nextchar();
  break;
}
case state of "45": {
  case state of 'a': state = "46";
  ch = nextchar();
  break;
}

case state of "46": {
  case state of 't': state = "47";
  ch = nextchar();
  break;
```

```
}
case state of "47": {
  case state of 't': state = "48";
  ch = nextchar();
  break;
}
case state of "49": {
  case state of 'i': state = "50";
  ch = nextchar();
  break;
}

case state of "50": {
  case state of 'l': state = "51";
  ch = nextchar();
  break;
}

case state of "51": {
  case state of 'o: state = "52";
  ch = nextchar();
  break;
}
case state of "52": {
  case state of 'w': state = "53";
  ch = nextchar();
  break;
}
case state of "53": {
  case state of 'a': state = "54";
```

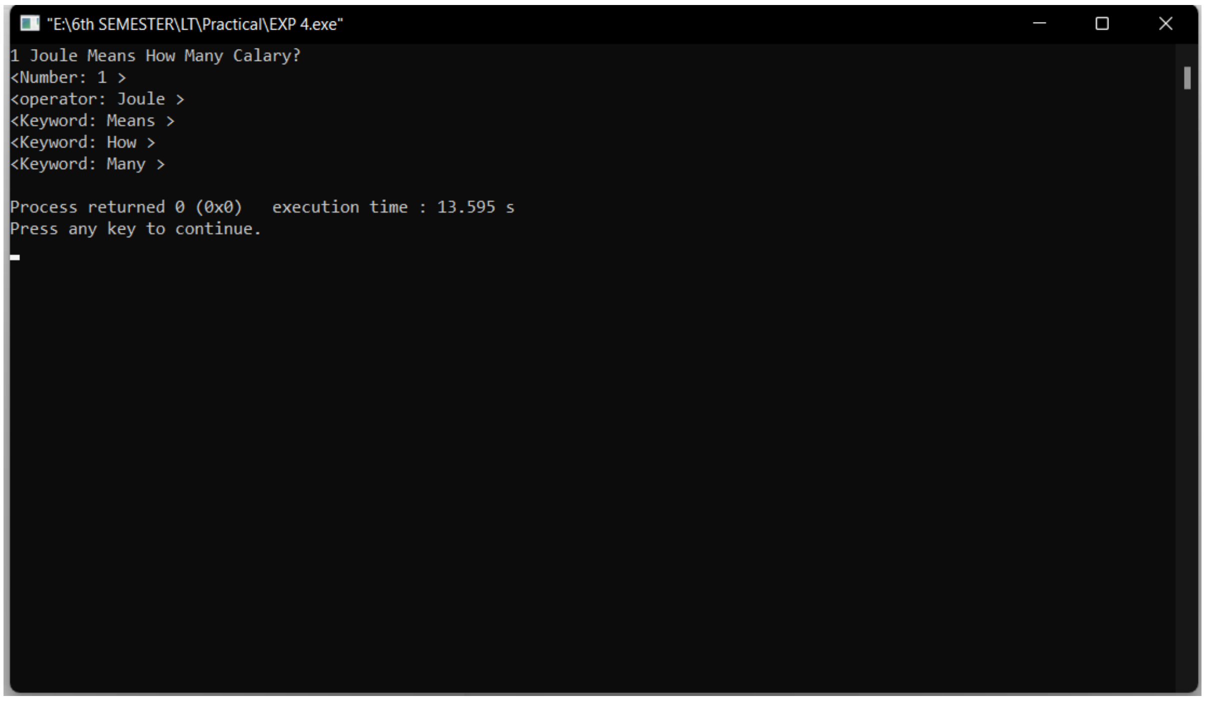
```
   ch = nextchar();
  break;
}

case state of "54": {
  case state of 'a': state = "55";
  ch = nextchar();
  break;
}

case state of "55": {
  case state of 't': state = "56";
  ch = nextchar();
  break;
}

case state of "57": {
  case state of [0 - 9]: state = "57";
  ch = nextchar();
  break;
  '.': state = "59";ch = nextchar();
  break;
}

case state of "59": {
  case state of [0 - 9]: state = "60";
  ch = nextchar();
  break;
}
case state of "60": {
      case state of [0 - 9]: state = "60";
      ch = nextchar();
      break;
      'other': state = "61";
       ch = nextchar();
      break;
    }

    case state of
      "12" | "5" | "15" | "22" | "26":
      print("Keyword");
    "31" | "38" | "44" | "56" | "48" | "65":
    print("operator");
    "58":
    print("int");
    "61":
    print("float");
    "62":
    print("whitespace");
    "63":
    print("Qmark");
    "64":
    print("EOS");

    ch: = nextchar();
    end
    case;

  }
}
```

# Scanner Phase in C++ Language

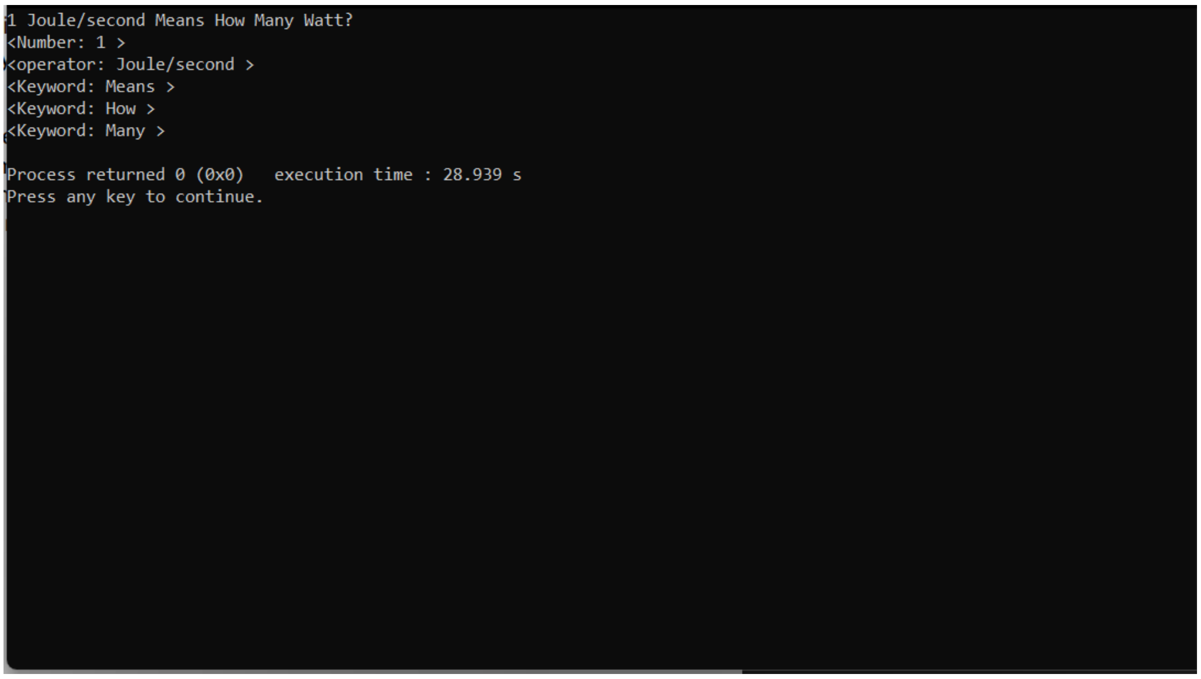
**Code:**

```
#include<bits/stdc++.h>
#include<iostream>

using namespace std;

bool isNumber(string s) {
  if (s.size() == 0)
    return false;
  for (int i = 0; i < s.size(); i++) {
    if ((s[i] >= '0' && s[i] <= '9') == false) {
      return false;
    }
  }
  return true;
}
int main() {
  string keywords[12] = {
    "Means",
    "MEANS",
    "HOW",
    "How",
    "Many",
    "MANY",
    "And",
    "AND",
    "Convert",
  "CONVERT",
  "Into",
  "INTO"
};
string operators[6] = {
  "Joule",
  "Calary",
  "Watt",
  "Hp",
  "Joule/second",
  "Kilowatt"
};
string EndOfLine = "?";
string str;
getline(cin, str);
istringstream iss(str);
string word;
string EOL = "?";

while (iss >> word) {
  for (int j = 0; j < 12; j++) {
    if (word.compare(keywords[j]) == 0) {


      cout << "<Keyword: " << keywords[j] << " >\n";
    }
  }
  for (int k = 0; k < 6; k++) {
    if (word.compare(operators[k]) == 0) {
      cout << "<operator: " << operators[k] << " >\n";
    }
    }
    if (isNumber(word)) {
      cout << "<Number: " << word << " >\n";
    }
    if (word == EOL) {
      cout << "<End of Line: " << EOL << " >\n";
    }
  }
}
```

## Output:



```
1 Joule Means How Many Calary?
<Number: 1 >
<operator: Joule >
<Keyword: Means >
<Keyword: How >
<Keyword: Many >

Process returned 0 (0x0)   execution time : 13.595 s
Press any key to continue.
```



```
1 Joule/second Means How Many Watt?
<Number: 1 >
<operator: Joule/second >
<Keyword: Means >
<Keyword: How >
<Keyword: Many >

Process returned 0 (0x0)   execution time : 28.939 s
Press any key to continue.
```

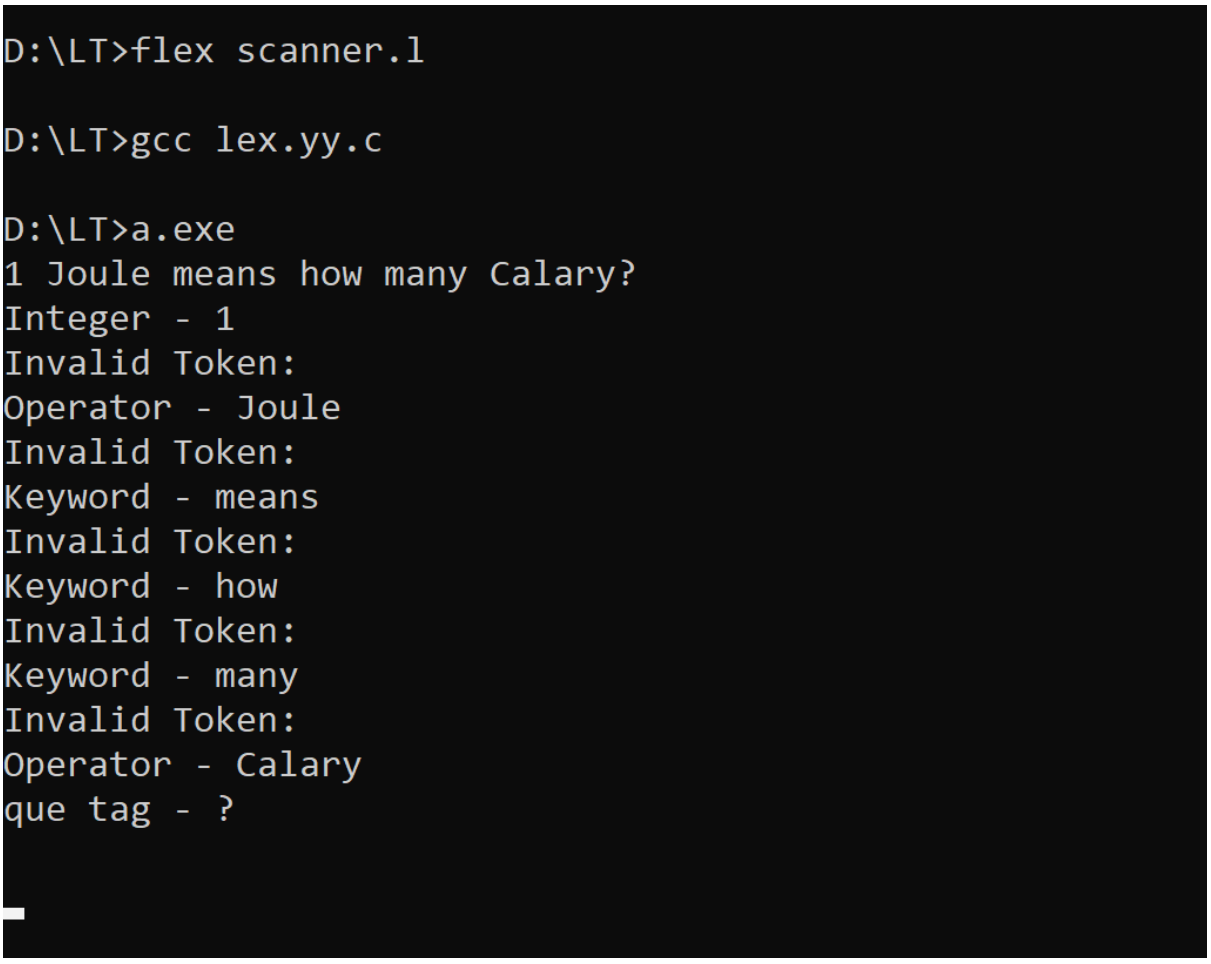
## Scanner Phase in Lex

**Code:**

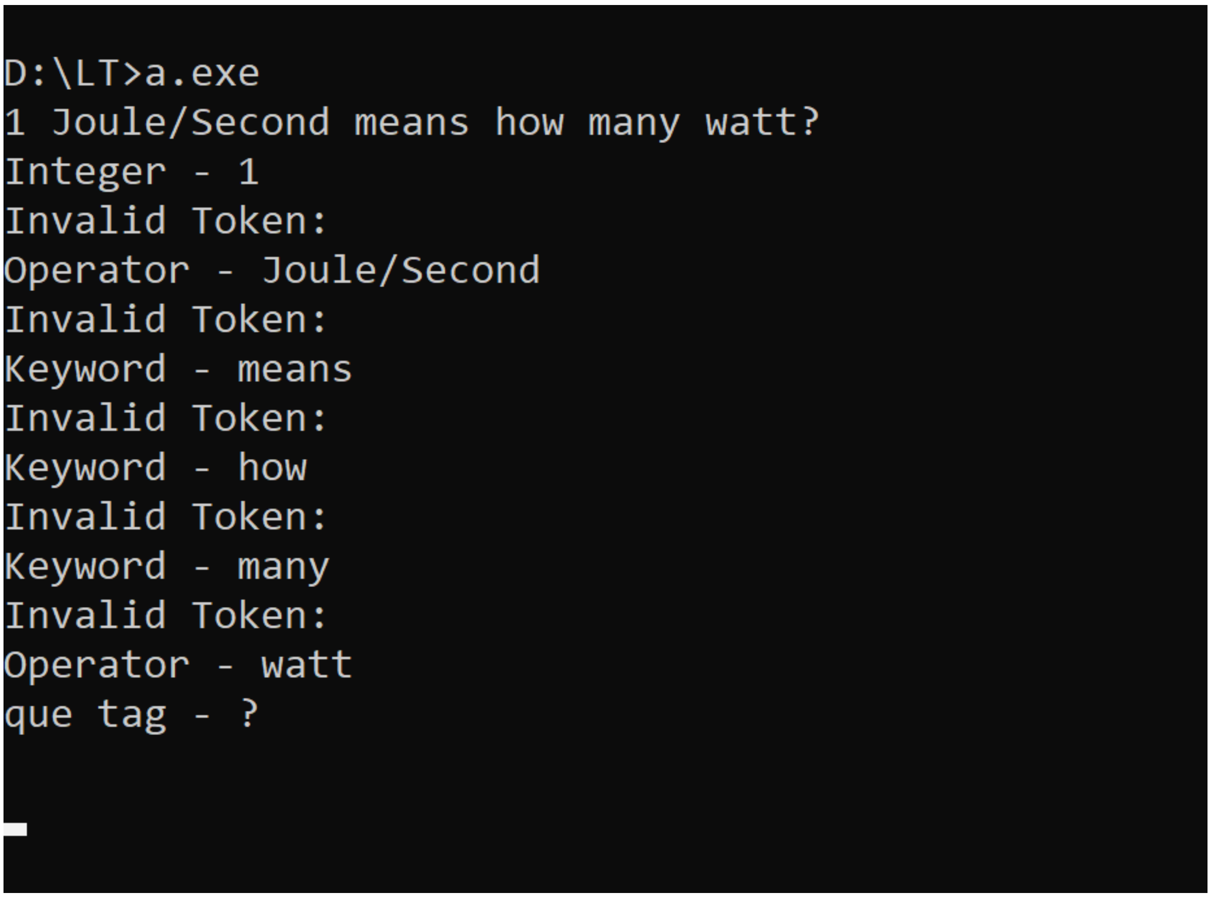
```
%{
#include<stdio.h>
%}

kw "how"|"many"|"means"|"and"|"convert"|"into"
op "Joule"|"hp"|"watt"|"kilowatt"|"Calary"|"Joule/Second"
digit [0-9]
Int {digit}+
Float {digit}+(.{digit})
qm "?"
ws [\t]
eos "."
sep ","

%%

{kw} {printf("Keyword - %s\n",yytext);}
{op} {printf("Operator - %s\n",yytext);}
{Int} {printf("Integer - %s\n",yytext);}
{Float} {printf("Float Number - %s\n",yytext);}
{qm} {printf("que tag - %s\n",yytext);}
{eos} {printf("eos - %s\n",yytext);}
{sep} {printf("sep - %s\n",yytext);}
{ws} {printf("ws \n",yytext);}
. {printf("Invalid Token: %s\n",yytext);}
```

```
%%

int yywrap(){
        return 1;
}


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

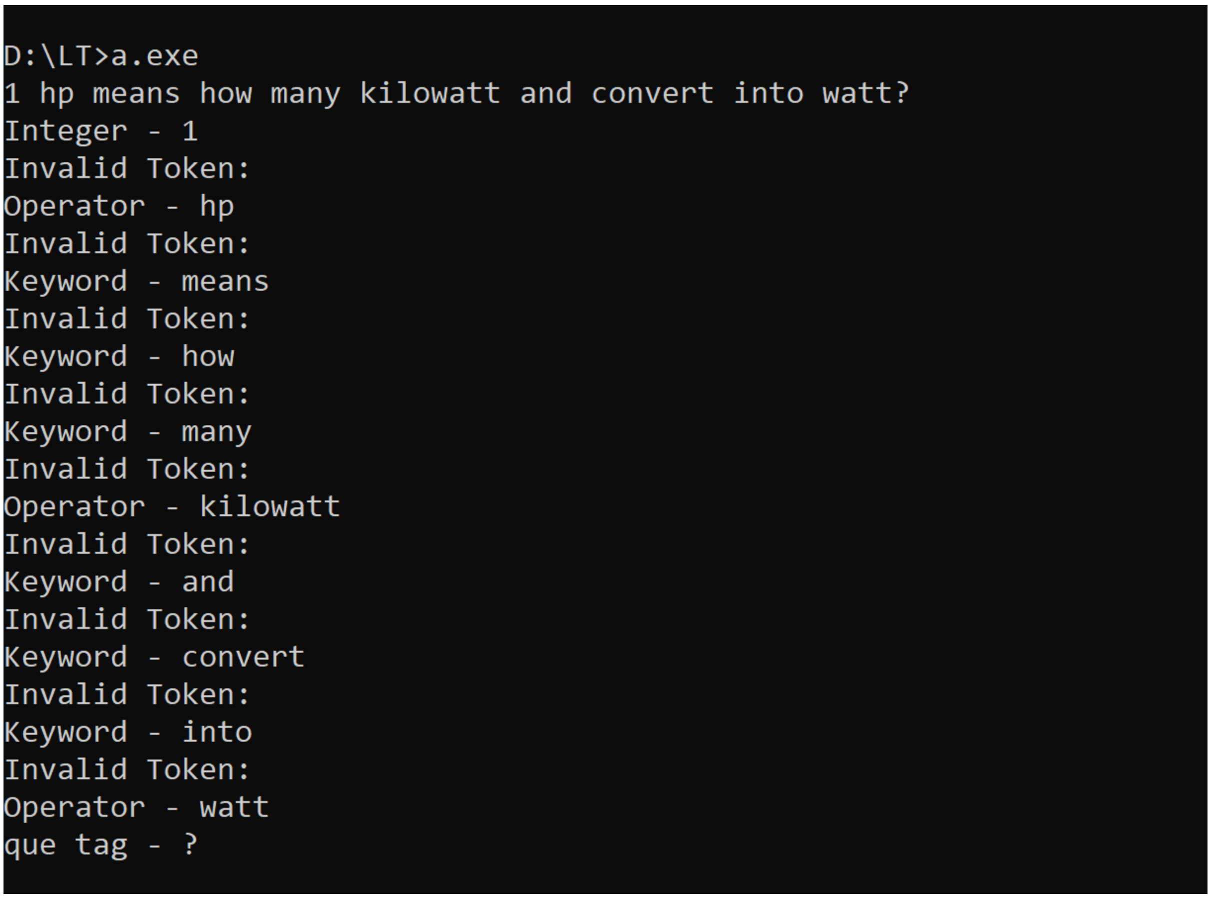
**Output:**



```
D:\LT>flex scanner.l

D:\LT>gcc lex.yy.c

D:\LT>a.exe
1 Joule means how many Calary?
Integer - 1
Invalid Token:
Operator - Joule
Invalid Token:
Keyword - means
Invalid Token:
Keyword - how
Invalid Token:
Keyword - many
Invalid Token:
Operator - Calary
que tag - ?
```



```
D:\LT>a.exe
1 Joule/Second means how many watt?
Integer - 1
Invalid Token:
Operator - Joule/Second
Invalid Token:
Keyword - means
Invalid Token:
Keyword - how
Invalid Token:
Keyword - many
Invalid Token:
Operator - watt
que tag - ?
```



```
D:\LT>a.exe
1 hp means how many kilowatt and convert into watt?
Integer - 1
Invalid Token:
Operator - hp
Invalid Token:
Keyword - means
Invalid Token:
Keyword - how
Invalid Token:
Keyword - many
Invalid Token:
Operator - kilowatt
Invalid Token:
Keyword - and
Invalid Token:
Keyword - convert
Invalid Token:
Keyword - into
Invalid Token:
Operator - watt
que tag - ?
```

# Yacc useful in our Language

**Code:**

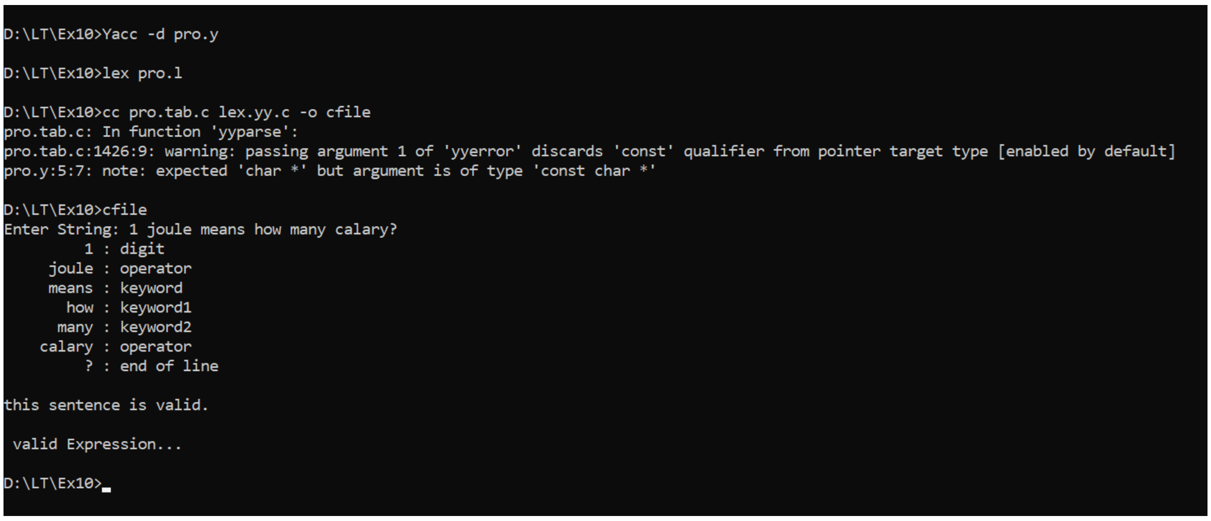
***pro.l***

```
%{
        #include<stdio.h>
        #include "pro.tab.h"
%}

keyword "means"
keyword1 "how"
keyword2 "many"
op "joule"|"calary"
eos1 "?"
digit [0-9]
ws " "



%%
{keyword} {
        printf("%10s : keyword\n",yytext);
        return K;
        }
{keyword1} {
        printf("%10s : keyword1\n",yytext);
        return K1;
        }
{keyword2} {
        printf("%10s : keyword2\n",yytext);
        return K2;
        }

{op} {
        printf("%10s : operator\n",yytext);
        return O;
        }

{digit} {
        printf("%10s : digit\n",yytext);
        return D;
        }

{eos1} {
        printf("%10s : end of line \n",yytext);
        return E1;
        }

{ws} {
        return W;
        }

. {
        printf("%10s : invalid\n",yytext);
        }

%%
int yywrap(){
        return 1;
        }
```

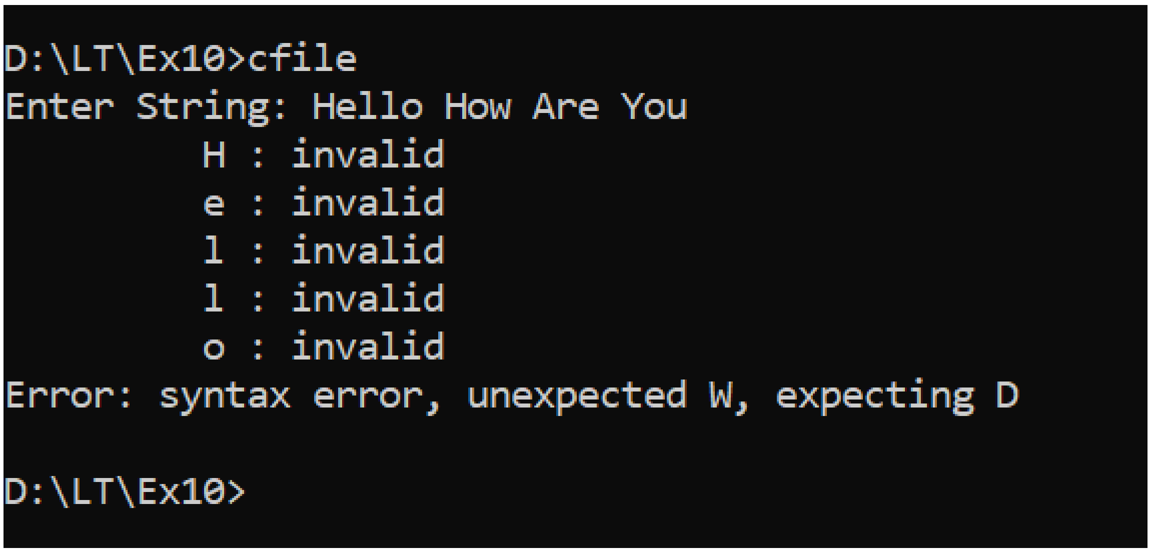
***pro.y***

```
%{
	#include<stdio.h>
	#include<stdlib.h>
	#define YYERROR_VERBOSE 1
	void yyerror(char *err);
%}

%token K K1 K2 O D E1 W

%%
S: A {printf("\nthis sentence is valid.\n"); return 0;};
A: D W O W K W R {};
R: K1 W K2 W O E1 {}



%%


void yyerror(char *err) {
	printf("Error: ");
	fprintf(stderr, "%s\n", err);
	exit(1);
}
void main(){
	printf("Enter String: ");
	yyparse();
	printf("\n valid Expression...\n");
}
```

**Output:**



```
D:\LT\Ex10>Yacc -d pro.y

D:\LT\Ex10>lex pro.l

D:\LT\Ex10>cc pro.tab.c lex.yy.c -o cfile
pro.tab.c: In function 'yyparse':
pro.tab.c:1426:9: warning: passing argument 1 of 'yyerror' discards 'const' qualifier from pointer target type [enabled by default]
pro.y:5:7: note: expected 'char *' but argument is of type 'const char *'

D:\LT\Ex10>cfile
Enter String: 1 joule means how many calary?
         1 : digit
     joule : operator
     means : keyword
       how : keyword1
      many : keyword2
    calary : operator
         ? : end of line

this sentence is valid.

 valid Expression...

D:\LT\Ex10>_
```



```
D:\LT\Ex10>cfile
Enter String: Hello How Are You
         H : invalid
         e : invalid
         l : invalid
         l : invalid
         o : invalid
Error: syntax error, unexpected W, expecting D

D:\LT\Ex10>
```

## Conclusion

As an IT student we have learned and worded many programming languages but we have hardly thought about the process that is running behind the picture. There is very complex logic that is implemented to make this language that we are using right now. But the subject of **Language Translator** makes us understand this complex mechanism.

By implementing this project, we have learnt the lexical and parsing phase of compiler design. This project has been using the knowledge that we have gained from college curriculum and from the internet. After doing this project we conclude that we have got more knowledge about how different compilers are working and handling the errors.

We would like to thank **Prof. Nikita P. Desai** for teaching this interesting subject, for the guidance in the project and for giving us a chance to know how compilers work.