Project Report on

Compiler for Weight Conversion

Developed by

IT041 HIREN JADAV

IT042 DISHANK INANI

IT043 VARSHITA JAIN

IT044 NAAMSUKH JOBANPUTRA

Guided By:

Prof. Nikita P. Desai

Department of Information Technology
Faculty of Technology, Dharmsinh Desai University
College Road, Nadiad-387001 2021-2022

DHARMSINH DESAI UNIVERSITY

NADIAD-387001, GUJARAT

CERTIFICATE

This is to certify that the project entitled "**Weight Conversion"** is a bonafied report of the work carried out by

1)	Hiren Jadav	ID No:19ITUOS022
2)	Dishank Inani, Student	ID No:19ITUOS107
3)	Varshita Jain, Student	ID No:19ITUOS136
4)	Naamsukh Jobanputra	ID No:19ITUOS118

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 academic year 2021-2022.

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

Prof. Yogendra Patel,
Department of Information Technology,
Faculty of Technology,
Dharmsinh Desai University, Nadiad

INDEX

1.0 INTRODUCTION

1.0.1 Project Details

Language Name: Weight Conversion using English language

Language description:

Write an appropriate language description for a layman language which can do weight conversion using English sentences.

Example of valid program in this language is

20 kilograms is how many grams?

1.0.2 Project Planning

List of Students with their Roles/Responsibilities:

IT041 Hiren Jadav: - Regular Expression, Implementation of lexer, DFA design, Final Report

IT042 Dishank Inani:- Regular Expression, Implementation of lexer, Algorithm design and implementation

IT043 Varshita Jain: - Regular Expression, Implementation of lexer, Grammar Rules, YACC implementation

IT044 Naamsukh Jobanputra: - Regular Expression, Final Report, Implementation of lexer, Scanner Phase implementation

2.0 LEXICAL PHASE DESIGN

2.0.1 Regular Expression:

Keyword:

RE	Token
kilograms	kilograms
kilogram	kilogram
kg	kg
kgs	kgs
milligrams	milligrams
mgs	mgs
mg	mg
much	much
many	many
grams	grams
gram	gram
gms	gms
gm	gm
how	how
to	to
is	is

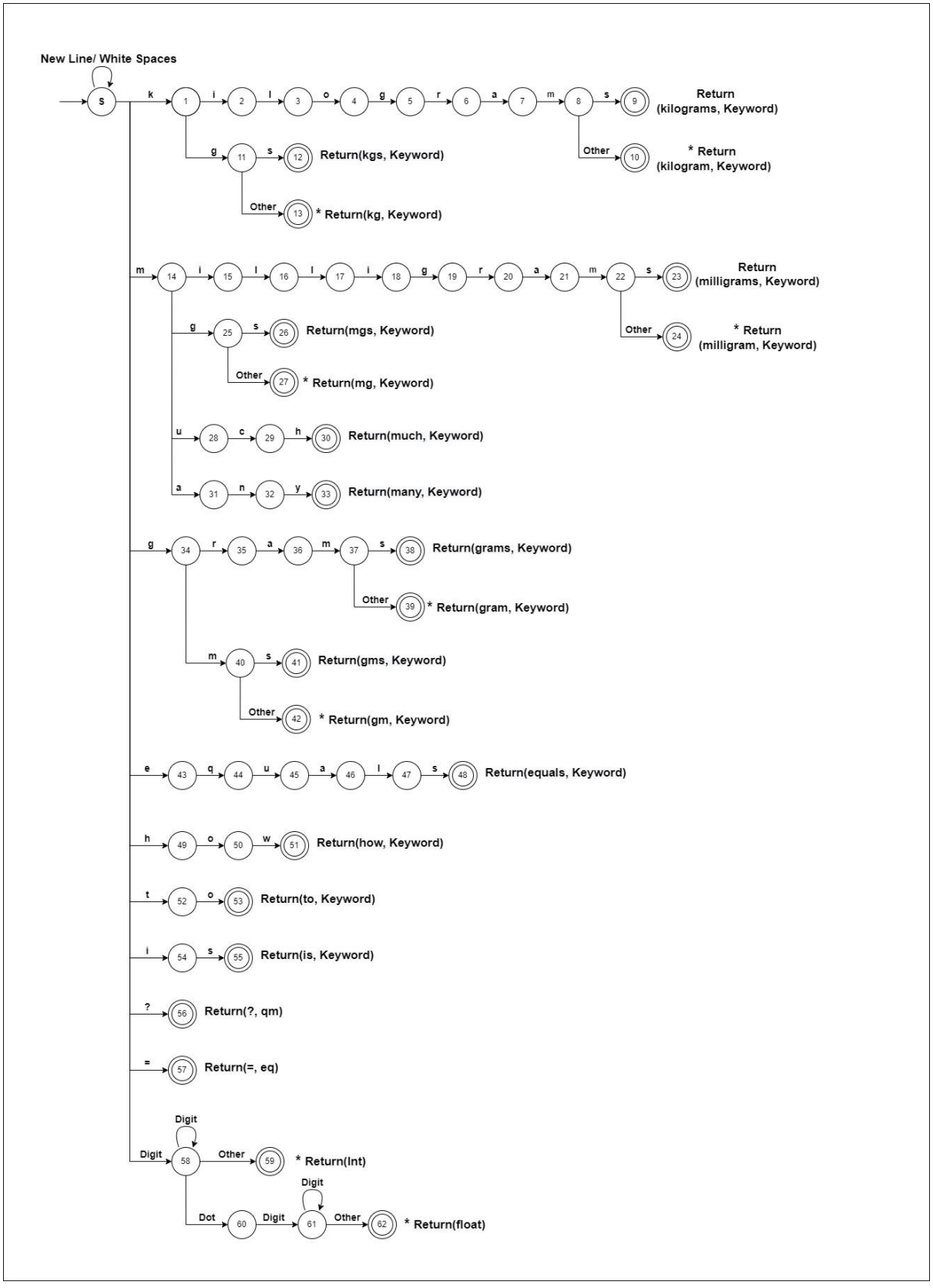
Values type: int and float

RE	Token
[0-9] +	int
[0-9] + (. [0-9] +)	float

Delimiters: {? \t}

RE	Token
?	qm
[\t]	ws
[\n]	nl

2.0.2 Deterministic Finite Automata design for lexer



2.0.3 Algorithm of lexer

```
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
                  'k':
                     state = "1";
                     ch = nextchar();
                     break;
```

```
'm':
    state = "14";
    ch = nextchar();
    break;

'g':
    state = "34";
    ch = nextchar();
    break;

'e':
    state = "43";
    ch = nextchar();
    break;

'h':
    state = "49";
    ch = nextchar();
    break;

't':
    state = "52";
    ch = nextchar();
    break;

'i':
    state = "54";
    ch = nextchar();
    break;
```

```
        '?':
            state = "56";
            ch = nextchar();
            f = true;
            break;

        '=':
            state = "57";
            ch = nextchar();
            f = true;
            break;

        [0-9]:
            state = "58";
            ch = nextchar();
            break;

        Default:
            f = true;

    end case
}

case state of "1": {

    case state of
        'i':
            state = "2";
```

```
            ch = nextchar();
            break;

        'g':
            state = "11";
            ch = nextchar();
            f = true;
            break;

        'Default':
            f=true;

    end case
}

case state of "2": {
    case state of 'l':
        state = "3";
        ch = nextchar();
        break;
}

case state of "3": {
    case state of 'o':
        state = "4";
        ch = nextchar();
        break;
}
```

```
case state of "4": {
    case state of 'g':
        state = "5";
        ch = nextchar();
        break;
}

case state of "5": {
    case state of 'r':
        state = "6";
        ch = nextchar();
        break;
}

case state of "6": {
    case state of 'a':
        state = "7";
        ch = nextchar();
        break;
}

case state of "7": {
    case state of 'm':
        state = "8";
        ch = nextchar();
        f = true;
        break;

}
```

```
case state of "8": {
    case state of
    's':
        state = "9";
        ch = nextchar();
        f = true;
        break;

    Default:
        state = "10";
        f = true;
}

case state of "11": {
    case state of
    's':
        state = "12";
        ch = nextchar();
        f = true;
        break;

    Default:
        state = "13";
        f = true;
}

case state of "14": {
```

```
    case state of
        'i':
            state = "15";
            ch = nextchar();
            break;

        'g':
            state = "25";
            ch = nextchar();
            f = true;
            break;

        'u':
            state = "28";
            ch = nextchar();
            break;

        'a':
            state = "31";
            ch = nextchar();
            break;

        'Default':
            f=true;

    end case
}

case state of "15": {
```

```
    case state of 'l':
        state = "16";
        ch = nextchar();
        break;
}

case state of "16": {
    case state of 'l':
        state = "17";
        ch = nextchar();
        break;
}

case state of "17": {
    case state of 'i':
        state = "18";
        ch = nextchar();
        break;
}

case state of "18": {
    case state of 'g':
        state = "19";
        ch = nextchar();
        break;
}

case state of "19": {
    case state of 'r':
```

```
        state = "20";
        ch = nextchar();
        break;
}

case state of "20": {
    case state of 'a':
        state = "21";
        ch = nextchar();
        break;
}

case state of "21": {
    case state of 'm':
        state = "22";
        ch = nextchar();
        break;
}

case state of "22": {
    case state of
    's':
        state = "23";
        ch = nextchar();
        f = true;
        break;

    Default:
        state = "24";
```

```
        f = true;
}

case state of "25": {
    case state of
    's':
        state = "26";
        ch = nextchar();
        f = true;
        break;

    Default:
        state = "27";
        f = true;
}

case state of "28": {
    case state of 'c':
        state = "29";
        ch = nextchar();
        break;
}

case state of "29": {
    case state of 'h':
        state = "30";
        ch = nextchar();
        f = true;
        break;
```

```
}

case state of "31": {
    case state of 'n':
        state = "32";
        ch = nextchar();
        break;
}

case state of "32": {
    case state of 'y':
        state = "33";
        ch = nextchar();
        f = true;
        break;
}

case state of "34": {
    case state of
    'r':
        state = "35";
        ch = nextchar();
        break;

    'm':
        state = "40";
        ch = nextchar();
        break;
}
```

```
case state of "35": {
    case state of 'a':
        state = "36";
        ch = nextchar();
        break;
}

case state of "36": {
    case state of 'm':
        state = "37";
        ch = nextchar();
        break;
}

case state of "37": {
    case state of
    's':
        state = "38";
        ch = nextchar();
        f = true;
        break;

    Default:
        state = "39";
        f = true;
}

case state of "40": {
```

```
    case state of
    's':
        state = "41";
        ch = nextchar();
        f = true;
        break;

    Default:
        state = "42";
        f = true;
}




case state of "43": {
    case state of 'q':
        state = "44";
        ch = nextchar();
        break;
}

case state of "44": {
    case state of 'u':
        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 'l':
        state = "47";
        ch = nextchar();
        break;
}

case state of "47": {
    case state of 's':
        state = "48";
        ch = nextchar();
        f = true;
        break;
}

case state of "49": {
    case state of 'o':
        state = "50";
        ch = nextchar();
        break;
}

case state of "50": {
```

```
    case state of 'w':
        state = "51";
        ch = nextchar();
        f = true;
        break;
}

case state of "52": {
    case state of 'o':
        state = "53";
        ch = nextchar();
        f = true;
        break;
}

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

case state of "58": {
    case state of
    [0-9]:
        ch = nextchar();
        break;
```

```
    '.':
        state = "60";
        ch = nextchar();
        break;

    default:
        state = "59";
        f = true;
}

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

    default:
        f = true;
}

case state of "61": {
    case state of
    [0-9]:
        ch = nextchar();

    default:
        state = "62";
        f = true;
```

```
            }
        }
    }

    case state of

"9"|"10"|"12"|"13"|"23"|"24"|"26"|"27"|"30"|"33"|"38"|"39"|"41"|"42"|"48"|"51"|"53"|"55":
            print(" keyword");

        "56":
            print("qm");

        "57":
            print("eq");

        "59":
            print("Int");

        "62":
            print("Float");

        default:
            print("invalid input");
            ch := nextchar();

    end case;
  }
}
```

2.0.4 Implementation of lexer Flex

Program:

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

Keyword
"is"|"how"|"many"|"kilograms"|"kilogram"|"milligrams"|"milligram"|"grams"|"g
ram"|"to"|"much"|"equals"|"kg"|"kgs"|"gms"|"gm"|"mgs"|"mg"

qm "?"
Digit [0-9]
Int {Digit}+
Float {Digit}+(.{Digit})
eq "="
nl "\n"
ws " "

%%
{Keyword} {printf("Keyword - %s\n",yytext);}
{Int} {printf("Integer - %s\n",yytext);}
{Float} {printf("Float - %s\n",yytext);}
{qm} {printf("que tag - %s\n",yytext);}
{eq} {printf("eq - %s\n",yytext);}
{ws} {}
. {printf("Invalid token - %s\n",yytext);}
%%
```

```
int yywrap(){}

int main()
{
	FILE *fp;
	char fileName[50];
	printf("Enter File Name: ");
	scanf("%s",fileName);
	fp = fopen(fileName,"r");
	yyin = fp;
	yylex();
	return 0;
}
```

2.0.5 Execution environment setup

Step by Step Guide to Install FLEX and Run FLEX Program using Command Prompt(cmd)

Step 1

/*For downloading CODEBLOCKS */

- Open your Browser and type in "codeblocks"
- Goto to Code Blocks and go to downloads section
- Click on "Download the binary release"
- Download codeblocks-20.03mingw-setup.exe
- Install the software keep clicking on next

/*For downloading FLEX GnuWin32 */

- Open your Browser and type in "download flex gnuwin32"
- Goto to "Download GnuWin from SourceForge.net"
- Downloading will start automatically
- Install the software keep clicking on next

/*SAVE IT INSIDE C FOLDER*/

Step 2 /*PATH SETUP FOR CODEBLOCKS*/

- After successful installation

Goto program files->CodeBlocks-->MinGW-->Bin

- Copy the address of bin :- it should somewhat look like this

C:\Program Files (x86)\CodeBlocks\MinGW\bin

- Open Control Panel-->Goto System-->Advance System Settings-- >Environment Variables
- Environment Variables--> Click on Path which is inside System variables - Click on edit
- Click on New and paste the copied path to it:- - C:\Program Files (x86)\CodeBlocks\MinGW\bin - Press Ok!

Step 3 /*PATH SETUP FOR GnuWin32*/

- After successful installation Goto C folder
- Goto GnuWin32-->Bin
- Copy the address of bin it should somewhat look like this

C:\GnuWin32\bin

- Open Control Panel-->Goto System-->Advance System Settings-->Environment Variables
- Environment Variables--> Click on Path which is inside System variables
- Click on edit
- Click on New and paste the copied path to it:- - C:\GnuWin32\bin - Press Ok!

/*WARNING!!! PLEASE MAKE SURE THAT PATH OF CODEBLOCKS IS BEFORE GNUWIN32---THE ORDER MATTERS*/

Step 4

- Create a folder on Desktop flex_programs or whichever name you like - Open notepad type in a flex program - Save it inside the folder like filename.l

-Note :- also include “”” void yywrap(){} “””””””” in the .l file

/*Make sure while saving save it as all files rather than as a text document*/ Step 5 /*To RUN FLEX PROGRAM*/

- Goto to Command Prompt(cmd)
- Goto the directory where you have saved the program - Type in command :- **flex filename.l** - Type in command :- **gcc lex.yy.c**
- Execute/Run for windows command promt :- a.exe

Step 6

- Finished

PROGRAM:

```
%{
        #include<stdio.h>
%}
Keyword
"is"|"how"|"many"|"kilograms"|"kilogram"|"milligrams"|"milligram"|"grams"|"g
ram"|"to"|"much"|"equals"|"kg"|"kgs"|"gms"|"gm"|"mgs"|"mg"
qm "?"
Digit [0-9]
Int {Digit}+
Float {Digit}+(.{Digit})
eq "="
nl "\n"
ws " "

%%
{Keyword} {printf("Keyword - %s\n",yytext);}
{Int} {printf("Integer - %s\n",yytext);}
{Float} {printf("Float - %s\n",yytext);}
{qm} {printf("que tag - %s\n",yytext);}
{eq} {printf("eq - %s\n",yytext);}
{ws} {}
. {printf("Invalid token - %s\n",yytext);}
%%

int yywrap(){}
int main()
{
```

```
FILE *fp;
char fileName[50];
printf("Enter File Name: ");
scanf("%s",fileName);
fp = fopen(fileName,"r");
yyin = fp;
yylex();
return 0;
}
```

2.0.6 Output screenshots of lexer

OUTPUT FOR VALID TOKEN:

OUTPUT FOR INVALID TOKEN:

3.0 SYNTAX ANALYZER DESIGN

3.0.1 Grammar Rules

Start: S | ^

S: ‘How many’ UNIT ‘is’ VALUE UNIT EOS ‘\n’ | VALUE UNIT ‘is how many’ UNIT EOS ‘\n’ | VALUE UNIT equals how many UNIT EOS ‘\n’

VALUE: INTEGER | FLOAT

3.0.2 Yacc Implementation

Flex Code:

```
%{
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
void yyerror(char *);
#include "proj.tab.h"
%}
%%
[0-9]+ { yylval.num = atoi(yytext); return INTEGER; }
[0-9]+[.][0-9]+ { yylval.val = atof(yytext); return FLOAT; }
"exit" { return EXIT; }
[isequalshowmany] { return *yytext; }
```

```
"kilograms"|"kgs"|"kilogram"|"kg"|"grams"|"gram"|"milligrams"|"milligram"|"g
ms"|"gm"|"mgs"|"mg" { yylval.str = strdup(yytext); return UNIT; }

"?" { return EOS; }

[!\n] { return *yytext; }

[ |\t] { /* Ignoring Whitespace */ }

. { yyerror("Unrecognized Character\n"); }

%%


void main(){

 yyparse();

}

int yywrap(){}

void yyerror(char *s) {

 fprintf(stderr, "%s\n", s);

}

void convert(float value, char *unit1, char *unit2)

{

double newValue = value;

 if(strcmp(unit1, "kg") == 0 || strcmp(unit1, "kilogram") == 0 ||  strcmp(unit1,
"kgs") == 0 || strcmp(unit1, "kilograms") == 0)

        {

                if(strcmp(unit2, "gram") == 0 || strcmp(unit2, "grams") == 0 ||
strcmp(unit2,"gms") == 0 || strcmp(unit2,"gm") == 0)

                {

                        newValue = value * 1000.0;

                }

                else if(strcmp(unit2, "mgs") == 0 || strcmp(unit2, "mg") == 0 ||
strcmp(unit2,"milligram") == 0 || strcmp(unit2,"milligrams") == 0)

                {
```

```
                newValue = value * 1000000.0;
            }
        }
 else if(strcmp(unit1, "gram") == 0 || strcmp(unit1, "grams") == 0 || strcmp(unit1,"gms") == 0 || strcmp(unit1,"gm") == 0)
        {
            if(strcmp(unit2, "kilogram") == 0 || strcmp(unit2, "kilograms") == 0 || strcmp(unit2,"kgs") == 0 || strcmp(unit2,"kg") == 0)
            {
                newValue = (value / 1000.0);
            }
            else if(strcmp(unit2, "milligram") == 0 || strcmp(unit2, "milligrams") == 0 || strcmp(unit2,"mgs") == 0 || strcmp(unit2,"mg") == 0)
            {
                newValue = (value * 1000.0);
            }

        }
 else if(strcmp(unit1, "milligrams") == 0 || strcmp(unit1, "milligram") == 0 || strcmp(unit1,"mgs") == 0 || strcmp(unit1,"mg") == 0)
        {
            if(strcmp(unit2, "kgs") == 0 || strcmp(unit2, "kilograms") == 0 || strcmp(unit2, "kg") == 0 || strcmp(unit2, "kilogram") == 0)
            {
                newValue = (double) value / 1000000.0;
            }

            else if(strcmp(unit2, "gram") == 0 || strcmp(unit2, "grams") == 0 || strcmp(unit2,"gms") == 0 || strcmp(unit2,"gm") == 0)
```

```
                {
                        newValue = value / 1000.0 ;
                }
        }
printf("%.4f %s\n\n", newValue, unit2);
}
```

YACC Code:

```
%{
 #include <stdio.h>
 #include <stdlib.h>
 #include <math.h>
 #include <string.h>
 int yylex(void);
 void yyerror(char *);
 void convert(float value, char *unit1, char *unit2 );
%}
%union{ int num; float val; char* str};
%token <num> INTEGER
%token <val> FLOAT
%token <str> UNIT
%token EOS
%token EXIT
%type <val> VALUE
%%
Program : Program Start {}
 |
```

```
 ;
Start : 'h"o"w"m"a"n"y' UNIT 'i"s' VALUE UNIT EOS '\n'
 { convert( $11, $12, $8 ); }
 | VALUE UNIT 'e"q"u"a"l"s"h"o"w"m"a"n"y' UNIT EOS '\n'
{ convert( $1, $2, $16 ); }
 | VALUE UNIT 'i"s' 'h"o"w' 'm"a"n"y' UNIT EOS '\n'
{ convert( $1, $2, $12 ); }
 | EXIT { exit(0); }
 ;
VALUE : INTEGER { $$ = (float)$1; }
 | FLOAT {}
 ;
%%
```

3.0.3 Output screenshots of yacc based implementation

OUTPUT FOR VALID:

```
E:\SEM 6\LT\Project\Lab 10>project.exe
how many gram is 1 kilogram?
1000.0000 gram

2 kg equals how many grams?
2000.0000 grams

10 gram is how many mgs?
10000.0000 mgs

exit

E:\SEM 6\LT\Project\Lab 10>
```

OUTPUT FOR INVALID:

```
E:\SEM 6\LT\Project\Lab 10>project.exe
how many meter is 1 kg?
syntax error
```

```
E:\SEM 6\LT\Project\Lab 10>project.exe
2 rupees equals how many paisa?
Unrecognized Character

syntax error
```

4.0 CONCLUSION

This project has been implemented by applying the concepts learned in our college curriculum in addition to many rich resources from the web. This project gave us an opportunity to have a better understanding of the theoretical concepts learned in the subject by implementing them. We would like to thank Prof. Nikita P. Desai for encouraging us to implement this project. This project has indeed helped us improve our problem solving and team works skills and has given us a better insight of the inner workings of a compiler**.**