Code:-

```
#include <iostream>
#include <iostream>
#include <cctype>
#include <regex>
#include<map>
#include <stack>
#include<string.h>
#include <ctype.h>
using namespace std;
string postfix(string pp){
    map<char,int> preced;
    preced['*']=2;
    preced['/']=2;
    preced['-']=1;
    preced['+']=1;
    stack<char> st;
    string fin;
    //cout<<pp<<endl;
    reverse(pp.begin(), pp.end());
    for(auto x:pp){
        if(x=='*'||x=='/'||x=='+'||x=='-'){
            if(st.empty()==true){
                st.push(x);
            }
            else{
                int a = preced[st.top()];
                int b = preced[x];
                if(a>b){
                    fin=fin+st.top();
                    st.pop();
                    st.push(x);
                }
                else if(a<=b){
                    st.push(x);
                }
            }

        }
        else{
        fin=fin+x;
```

```
        }
    }
    while(st.empty()==false){
    fin=fin+st.top();
    st.pop();
    }
    //cout<<fin<<endl;
    return fin;
}
int eval(string s,map<string,int> finall){
    stack<float> st;
    for(auto x:s){
         if(x=='*'){
             float a = st.top();
             st.pop();
             float b = st.top();
             st.pop();
             st.push(a*b);
         }
        else if(x=='/'){
             float a = st.top();
             st.pop();
             float b = st.top();
             st.pop();
             st.push(a/b);
         }
        else if(x=='-'){
             float a = st.top();
             st.pop();
             float b = st.top();
             st.pop();
             st.push(a-b);
         }
        else if(x=='+'){
             float a = st.top();
             st.pop();
             float b = st.top();
             st.pop();
             st.push(a+b);
         }
        else{
        string p;
```

```
        p.push_back(x);
        if(isdigit(x)){
            st.push(stoi(p));
            continue;
        }
        //cout<<finall[p]<<endl;
        st.push(finall[p]);
        }
    }
    //cout<<"value"<<"-->"<<st.top()<<endl;
    return st.top();

}
void parser(map<pair<int,string>,string> expr,int n,vector<int> check,map<s
tring,int> &finall){
stack<string> ss;
int iter = 1;
auto vt = expr.begin();
string pp;
for(;vt!=expr.end();vt++){
//first
if((vt->first).second=="let"&&check[iter]==4){
for(int z=1;z<=4;z++){
if((vt->second)=="keyword"){
    vt++;
}
else if(vt->second=="identifier"){
    auto kk = (vt->first).second;
    ss.push(kk);
    vt++;
}
else if(vt->second=="equalsign"){
    vt++;
}
else if(vt->second=="value"){
    auto l = (vt->first).second;
    int pp=stoi(l);
    auto top = ss.top();
    ss.pop();
    finall[top]=pp;
}
}
```

```
}
//second
else if((vt->first).second=="print"){
    for(int z=1;z<=2;z++){
    if((vt->first).second=="print"){
    vt++;
    }
    else {
        if((vt->second)=="direct"){
            cout<<(vt->first).second<<endl;
            continue;
        }
        if((vt->second)=="directto"){
            auto z = (vt->first).second;
            string mm;
            string a;
            string b;
            int flag=0;
            for(auto ll:z){
                if(ll=='*'){
                    a=mm;
                    if(finall.find(mm) != finall.end()){
                    a=to_string(finall[mm]);
                    }
                    mm="";
                    flag=1;
                    continue;
                }
                else if(ll=='/'){
                    a=mm;
                    if(finall.find(mm) != finall.end()){
                    a=to_string(finall[mm]);
                    }
                    mm="";
                    flag=2;
                    continue;
                }
                else if(ll=='+'){
                    a=mm;
                    if(finall.find(mm) != finall.end()){
                    a=to_string(finall[mm]);
                    }
```

```
                mm="";
                flag=3;
                continue;
            }
            else if(ll=='-'){
                a=mm;
                if(finall.find(mm) != finall.end()){
                a=to_string(finall[mm]);
                }
                mm="";
                flag=4;
                continue;
            }
           mm=mm+ll;
        }
        b=mm;
        if(flag==1){
            int xx=stoi(a);
            int yy=stoi(b);
            cout<<(xx*yy)<<endl;
        }
        if(flag==2){
            int xx=stoi(a);
            int yy=stoi(b);
            cout<<(xx/yy)<<endl;
        }
        if(flag==3){
            int xx=stoi(a);
            int yy=stoi(b);
            cout<<(xx+yy)<<endl;
        }
        if(flag==3){
            int xx=stoi(a);
            int yy=stoi(b);
            cout<<(xx-yy)<<endl;
        }
        continue;
    }
    else{
    cout<<finall[(vt->first).second]<<endl;
    }
}
```

```
    }
}
//third
else if((vt->first).second=="let"&&check[iter]>4){
for(int z=1;z<=check[iter];z++){
if((vt->second)=="keyword"){
    vt++;
}
else if(vt->second=="identifier"&&z<=2){
    auto kk = (vt->first).second;
    ss.push(kk);
    vt++;
}
else if(vt->second=="equalsign"){
    vt++;
}
else{
  pp=pp+(vt->first).second;
  if(z==check[iter]){
      pp=postfix(pp);
      int kl = eval(pp,finall);
      auto top = ss.top();
      ss.pop();
      finall[top]=kl;
      continue;
  }
  vt++;
}
}
}
//fourth
iter++;
}

}
int main() {
    map<pair<int,string>,string> expr;
    int ll;
    string s;
    string para;
    vector<int> check;
    map<string,int> finall;
```

```
    while(getline(cin,s)){
        para=para+s+" ";
    }
    string word;
    int i=0;
    int z=0;
    string zzz;
    //basically here i am tokenizing for difffernt tokenz
    for(auto x:para){
        if(x==' '){
                i++;
                if((word.find('0') != string::npos || word.find('1') != str
ing::npos ||word.find('2') != string::npos ||word.find('3') != string::npos
 ||word.find('4') != string::npos ||word.find('5') != string::npos ||word.f
ind('6') != string::npos ||word.find('7') != string::npos ||word.find('8')
!= string::npos ||word.find('9') != std::string::npos)){
                if(zzz=="print"){
                expr[(make_pair(i,word))]="directto";
                }
                }
                if(word.find_first_not_of("0123456789") == string::npos){
                expr[(make_pair(i,word))]="value";
                zzz="value";
                }
                else if(word.find("\"")!=string::npos){
                expr[(make_pair(i,word))]="direct";
                zzz="direct";
                }
                else if(word=="LET"||word=="let"||word=="print"||word=="pri
ntln"||word=="PRINT"||word=="PRINTLN"||word=="print"){
                expr[(make_pair(i,word))]="keyword";
                check.push_back((i-z));
                z=i;
                ll=z;
                if(word=="print"){
                zzz="print";
                }
                else{
                    zzz="random";
                }
                 }
```

```
                else if(regex_match(word,regex("^[A-Za-
z]+$"))&&word.length()==1){
                expr[(make_pair(i,word))]="identifier";
                zzz="identifier;";
                 }
                else if(word=="/"||word=="*"||word=="+"||word=="-"){
                expr[(make_pair(i,word))]="operator";
                zzz="operator";
                }
                else if(word=="="){
                expr[(make_pair(i,word))]="equalsign";
                zzz="equalsign";
                }
                word="";
            continue;
        }
        word=word+x;
    }
    check[0]=0;
    check.push_back((i-ll+1));
    i=i+1;
    expr[(make_pair(i,"END"))]="END";

    //basically sending to parser which will take care of everything
    parser(expr,i,check,finall);

    return 0;
}
```

INPUT-OUPUT:-

```
                }
                }
                else if(regex_match(word,regex("^[A-Za-z]+$"))&&word.length()==1){
                expr[(make_pair(i,word))]="identifier";
                zzz="identifier;";
                }
                else if(word=="/"||word=="*"||word=="+"||word=="-"){
                expr[(make_pair(i,word))]="operator";
                zzz="operator";
                }
                else if(word=="="){
                expr[(make_pair(i,word))]="equalsign";
                zzz="equalsign";
                }
                word="";
            continue;
        }
        word=word+x;
    }
    check[0]=0;
    check.push_back((i-ll+1));
    i=i+1;
    expr[(make_pair(i,"END"))]="END";

    //basically sending to parser which will take care of everything
    parser(expr,i,check,finall);

    return 0;
}
```

Input

```
let a = 4
let b = 5
let c = 8
let d = a * b + b
let f = a / b + d - c + b
print d
print f
print "*"
print "/+/+"
print 64/16
print c/2
```

Output

```
25
12
"*"
"/+/+"
4
4
```

RUN C++ INPUT FILE VIEW SHARE code.cpp WEB EDITOR NEW Code Compiled Successfully

```
                }
                }
                else if(regex_match(word,regex("^[A-Za-z]+$"))&&word.length()==1){
                expr[(make_pair(i,word))]="identifier";
                zzz="identifier;";
                }
                else if(word=="/"||word=="*"||word=="+"||word=="-"){
                expr[(make_pair(i,word))]="operator";
                zzz="operator";
                }
                else if(word=="="){
                expr[(make_pair(i,word))]="equalsign";
                zzz="equalsign";
                }
                word="";
            continue;
        }
        word=word+x;
    }
    check[0]=0;
    check.push_back((i-ll+1));
    i=i+1;
    expr[(make_pair(i,"END"))]="END";

    //basically sending to parser which will take care of everything
    parser(expr,i,check,finall);

    return 0;
}
```

Input

```
let h = 8
print h/2
print 64/16
print h*8
print h/8
```

Output

```
4
4
64
1
```

RUN C++ INPUT FILE VIEW SHARE code.cpp WEB EDITOR NEW Code Compiled Successfully

```
                }
                }
                else if(regex_match(word,regex("^[A-Za-z]+$"))&&word.length()==1){
                expr[(make_pair(i,word))]="identifier";
                zzz="identifier;";
                }
                else if(word=="/"||word=="*"||word=="+"||word=="-"){
                expr[(make_pair(i,word))]="operator";
                zzz="operator";
                }
                else if(word=="="){
                expr[(make_pair(i,word))]="equalsign";
                zzz="equalsign";
                }
                word="";
            continue;
        }
        word=word+x;
    }
    check[0]=0;
    check.push_back((i-ll+1));
    i=i+1;
    expr[(make_pair(i,"END"))]="END";

    //basically sending to parser which will take care of everything
    parser(expr,i,check,finall);

    return 0;
}
```

Input

```
let a = 4
let b = 5
let c =7
let d = a * b + b
print "*"
print "=7="
print a/2
print 88/4
```

Output

```
"*"
"=7="
2
22
```

Logic :-

Here the language used is c++. While lexical analysis I stored everything in a map which had key as par<int,string> then used this to iterate in parsing.

For expression evaluation I reversed the string did precedence evaluation then used stack to evaluat. So basically infix to postfix then evaluation.

It is working with every test case total of 303 lines of code

The code can calculate any expression and takes into account the precedence and if equal follows left to right.

```
                }
                else if(word=="/"||word=="*"||word=="+"||word=="-"){
                expr[(make_pair(i,word))]="operator";
                zzz="operator";
                }
                else if(word=="="){
                expr[(make_pair(i,word))]="equalsign";
                zzz="equalsign";
                }
                word="";
            continue;
        }
        word=word+x;
    }
    check[0]=0;
    check.push_back((i-ll+1));
    i=i+1;
    expr[(make_pair(i,"END"))]="END";

    //basically sending to parser which will take care of everything
    parser(expr,i,check,finall);

    return 0;
}
```

```
RUN   C++   INPUT   FILE   VIEW   SHARE   ASSIGNMENT_7_code.cpp   WEB EDITOR (NEW)   LOGIN   MADE WITH ♥ BY CODING BLOCKS

            st.pop();
            float b = st.top();
            st.pop();
            st.push(a+b);
        }
        else{
        string p;
        p.push_back(x);
        if(isdigit(x)){
            st.push(stoi(p));
            continue;
        }
        //cout<<finall[p]<<endl;
        st.push(finall[p]);
        }
    }
    //cout<<"value"<<"-->"<<st.top()<<endl;
    return st.top();

}
void parser(map<pair<int,string>,string> expr,int n,vector<int> check,map<string,int> &finall){
stack<string> ss;
int iter = 1;
auto vt = expr.begin();
string pp;
for(;vt!=expr.end();vt++){
//first
if((vt->first).second=="let"&&check[iter]==4){
for(int z=1;z<=4;z++){
if((vt->second)=="keyword"){
```

```
Input
let a = 4
let b = 5
let d = a * b + 7
print "a"
print "*"
print "b"
print "="
print "+7"
print d

Output
"a"
"*"
"b"
0
"+7"
27
```