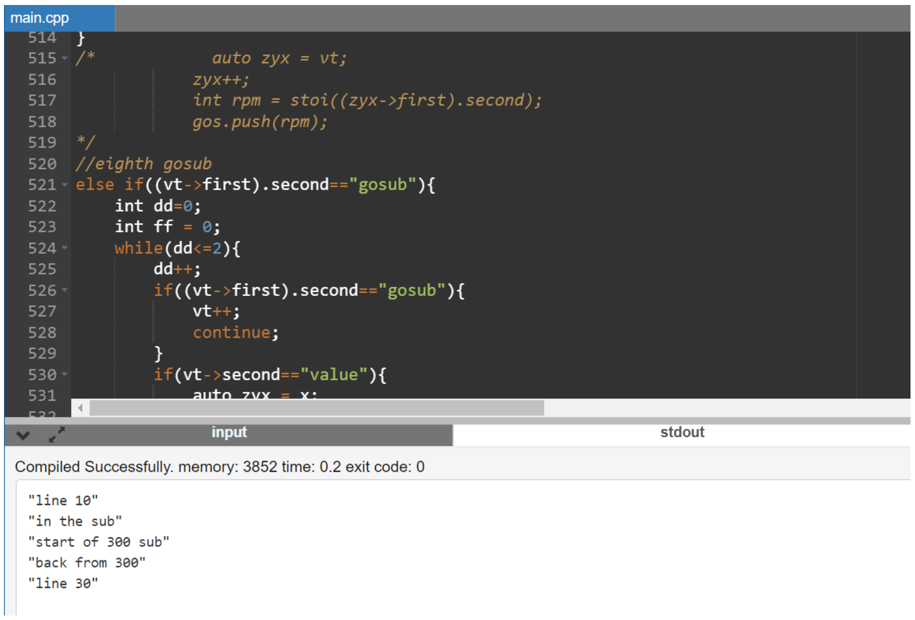
Example 1:-


main.cpp

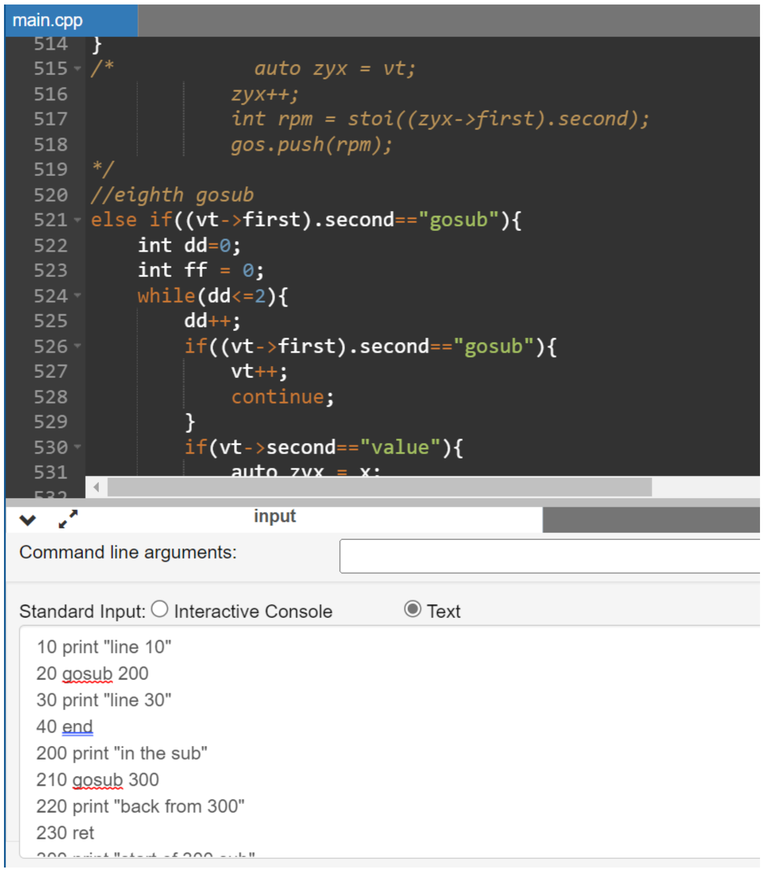
```cpp
}
/*          auto zyx = vt;
          zyx++;
          int rpm = stoi((zyx->first).second);
          gos.push(rpm);
*/
//eighth gosub
else if((vt->first).second=="gosub"){
    int dd=0;
    int ff = 0;
    while(dd<=2){
        dd++;
        if((vt->first).second=="gosub"){
            vt++;
            continue;
        }
        if(vt->second=="value"){
            auto zyx = x;
```

input stdout

Compiled Successfully. memory: 3852 time: 0.2 exit code: 0

```
"line 10"
"in the sub"
"start of 300 sub"
"back from 300"
"line 30"
```

Input :-


main.cpp

```cpp
}
/*          auto zyx = vt;
          zyx++;
          int rpm = stoi((zyx->first).second);
          gos.push(rpm);
*/
//eighth gosub
else if((vt->first).second=="gosub"){
    int dd=0;
    int ff = 0;
    while(dd<=2){
        dd++;
        if((vt->first).second=="gosub"){
            vt++;
            continue;
        }
        if(vt->second=="value"){
            auto zyx = x;
```

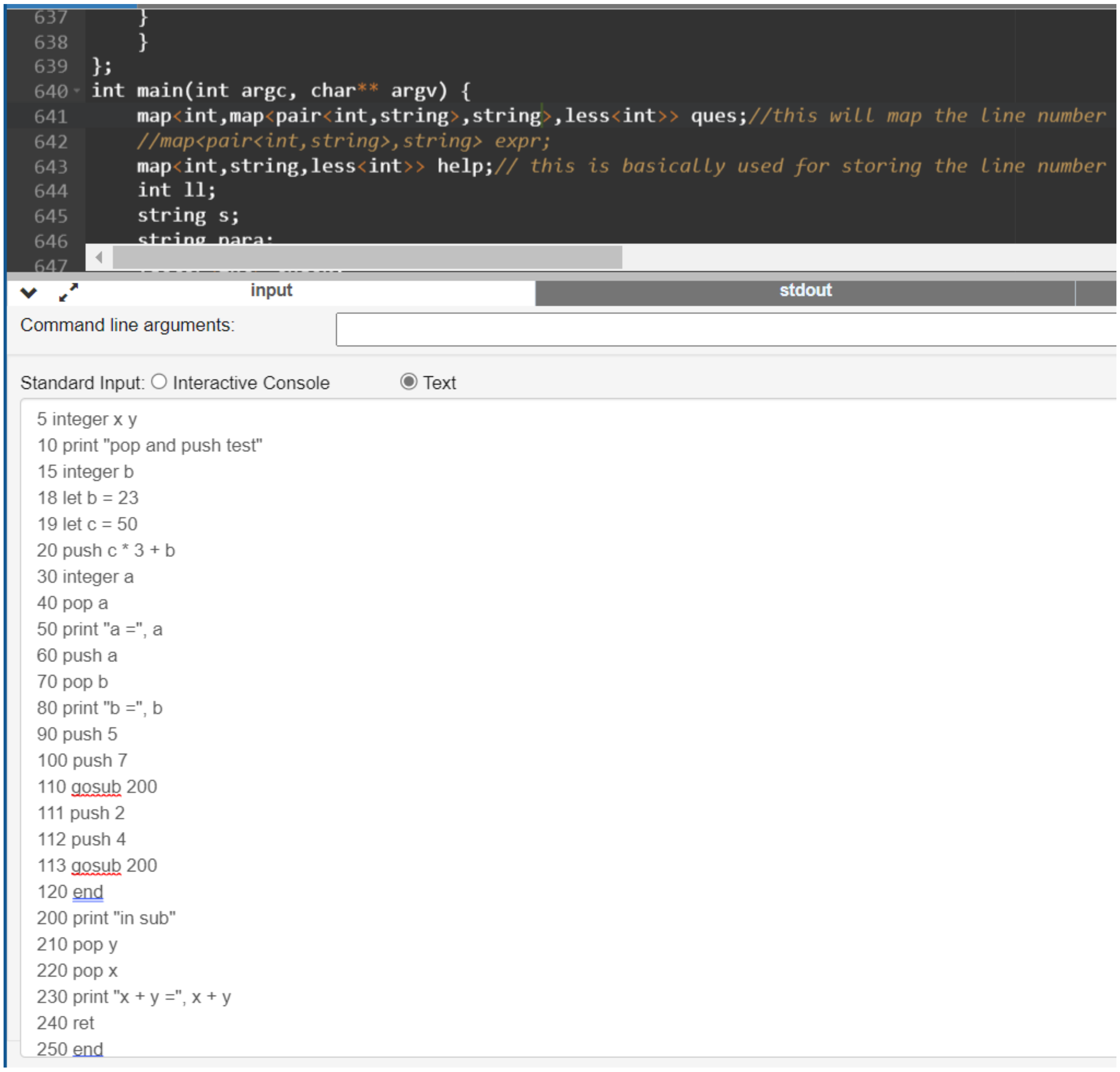
input

Command line arguments:

Standard Input: Interactive Console [✓ Text]

```
10 print "line 10"
20 gosub 200
30 print "line 30"
40 end
200 print "in the sub"
210 gosub 300
220 print "back from 300"
230 ret
300 print "start of 300 sub"
```

Input:-



```
    }
    }
};
int main(int argc, char** argv) {
    map<int,map<pair<int,string>,string>,less<int>> ques;//this will map the line number
    //map<pair<int,string>,string> expr;
    map<int,string,less<int>> help;// this is basically used for storing the line number
    int ll;
    string s;
    string para;
```

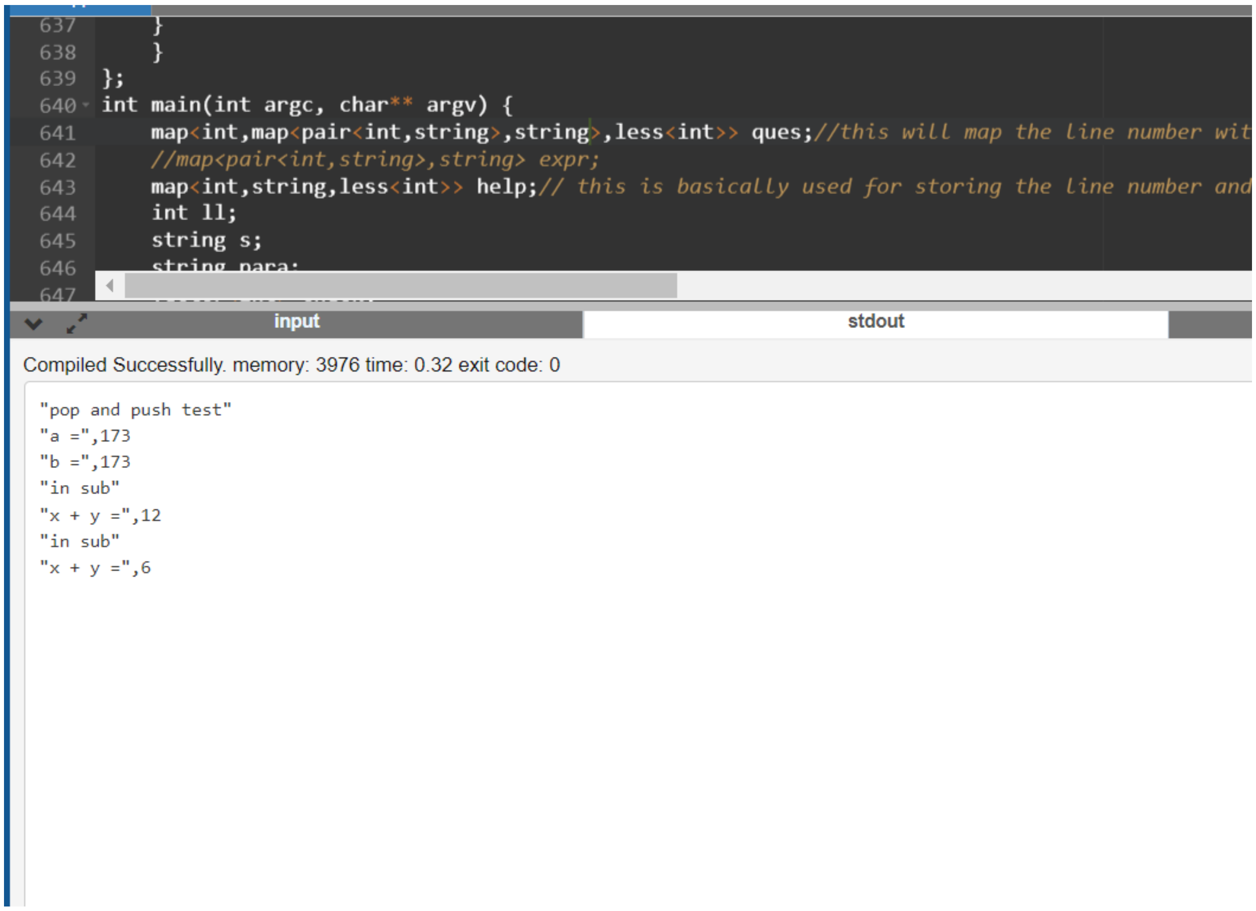
Command line arguments:

Standard Input: ○ Interactive Console [✓ Text]

```
5 integer x y
10 print "pop and push test"
15 integer b
18 let b = 23
19 let c = 50
20 push c * 3 + b
30 integer a
40 pop a
50 print "a =", a
60 push a
70 pop b
80 print "b =", b
90 push 5
100 push 7
110 gosub 200
111 push 2
112 push 4
113 gosub 200
120 end
200 print "in sub"
210 pop y
220 pop x
230 print "x + y =", x + y
240 ret
250 end
```

(here I guess in the input the line number 11,12 and 13 were wrongly given so I corrected to 111, 112 and 113)

Example 2:-

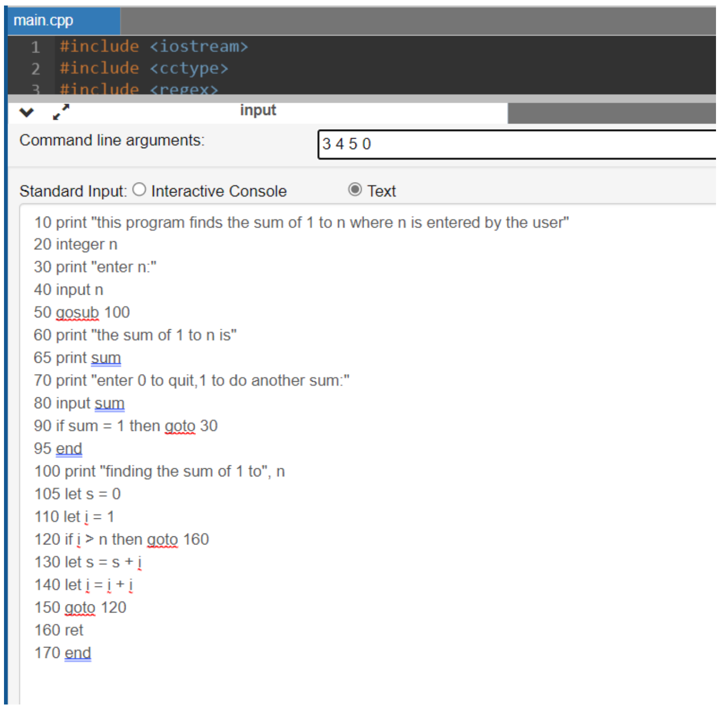


```
    }
    }
};
int main(int argc, char** argv) {
    map<int,map<pair<int,string>,string>,less<int>> ques;//this will map the line number wit
    //map<pair<int,string>,string> expr;
    map<int,string,less<int>> help;// this is basically used for storing the line number and
    int ll;
    string s;
    string para;
```

Compiled Successfully. memory: 3976 time: 0.32 exit code: 0

```
"pop and push test"
"a =",173
"b =",173
"in sub"
"x + y =",12
"in sub"
"x + y =",6
```

Example 3:



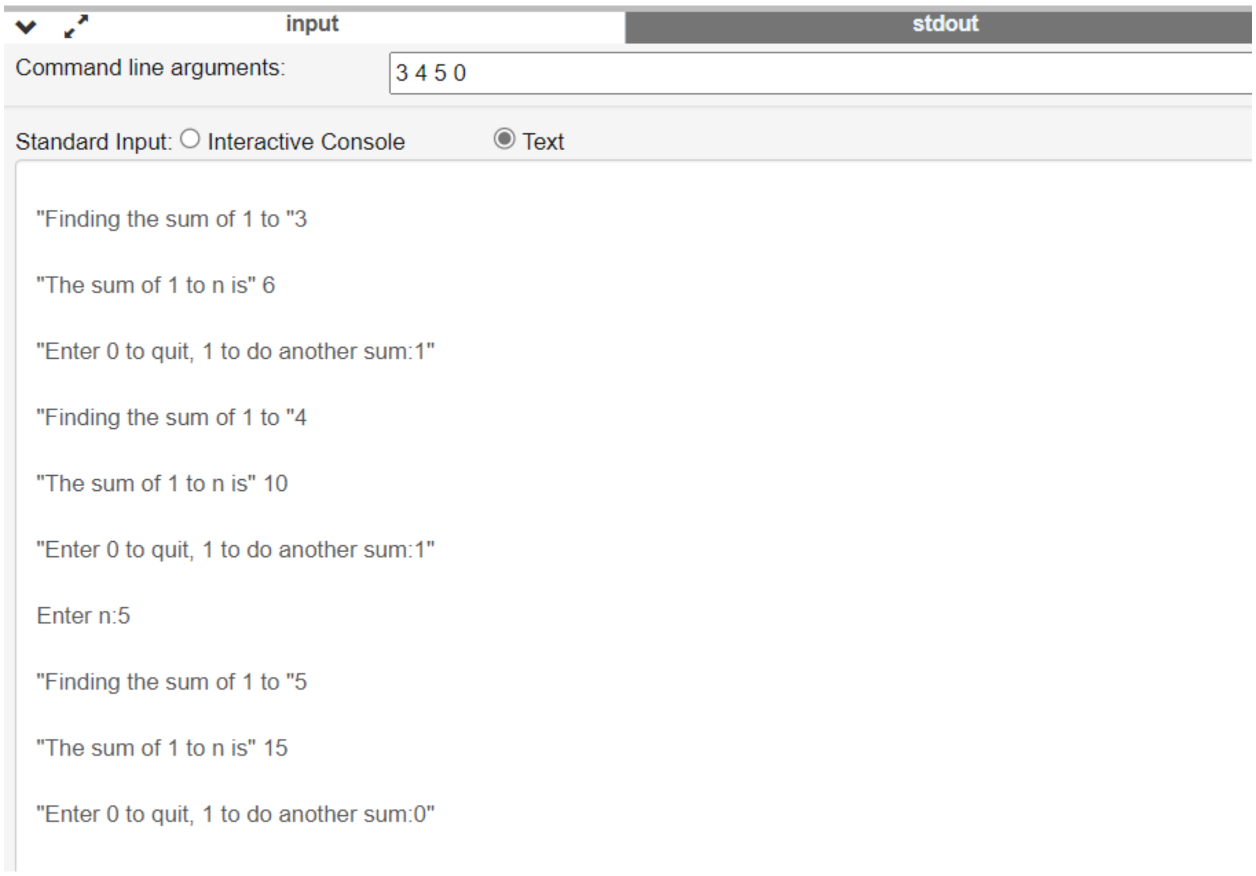
```
#include <iostream>
#include <cctype>
#include <regex>
```

Command line arguments: 3 4 5 0

Standard Input: ○ Interactive Console [✓ Text]

```
10 print "this program finds the sum of 1 to n where n is entered by the user"
20 integer n
30 print "enter n:"
40 input n
50 gosub 100
60 print "the sum of 1 to n is"
65 print sum
70 print "enter 0 to quit,1 to do another sum:"
80 input sum
90 if sum = 1 then goto 30
95 end
100 print "finding the sum of 1 to", n
105 let s = 0
110 let i = 1
120 if i > n then goto 160
130 let s = s + i
140 let i = i + i
150 goto 120
160 ret
170 end
```

OUPUT:-



```
Command line arguments: 3 4 5 0

Standard Input: ○ Interactive Console  ● Text

"Finding the sum of 1 to "3

"The sum of 1 to n is" 6

"Enter 0 to quit, 1 to do another sum:1"

"Finding the sum of 1 to "4

"The sum of 1 to n is" 10

"Enter 0 to quit, 1 to do another sum:1"

Enter n:5

"Finding the sum of 1 to "5

"The sum of 1 to n is" 15

"Enter 0 to quit, 1 to do another sum:0"
```

Logic :-

4 new functions were used: -

1)gosub :- for this I used a stack, whenever there was a gosub then I implemented the same functionality as goto but here I pushed the line number of that particular line in the stack.

2)ret:- whenever I encountered ret I popped the stack and got the top element, and then made the iterator to point to the next of the top element.

3)push :- For this whenever push keyword came in the parser from that point to the EOD the string was sent to the postfix function and then to evaluation . the output was pushed in stack .

4)pop:- for this whenever pop keyword came the next variable map was set to the top element of the stack and the element was popped.

5)print " ", a :- for this same as print but I checked the length of that line.

```
#include <iostream>

#include <cctype>

#include <regex>

#include<map>

#include <stack>

#include<string.h>

#include <ctype.h>

using namespace std;

int fg=0;

int ghgh=0;

stack<int> gos;

string anura;

stack<int> user;

string sat;

//this will make the conversion from infix to postfix

string postfix(string pp){

    map<char,int> preced;

    preced['*']=2;

    preced['/']=2;

    preced['-']=1;

    preced['+']=1;

    stack<char> st;

    string fin;

    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();
    }
    return fin;
}
//this will evaluate the string
//the method is if we get an operator we will pop the top two operator and do operation
```

```
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;

        }

        st.push(finall[p]);

        }

    }

    return st.top();


}

//this is used to check the if statement

//we will pass to postfix then check

bool letmecheck(string a,map<string,int> finall){

    int aa=0;

    int bb=0;

    string ll;

    string pp;

    string op;

    bool flag = false;

    int i=0;

    for(auto l:a){

        if(l==' '){

          i++;

          if(i==1){
```

```
pp=postfix(pp);

aa = eval(pp,finall);

pp="";

continue;

}

if(i==2){

  if(pp==">"){

      op = "greater";

  }

  else if(pp=="<"){

      op = "less";

  }

  else if(pp==">="){

      op="greaterequal";

  }

  else if(pp=="<="){

      op="lessqual";

  }

  else if(pp=="="){

      op="equal";

  }

  else if(pp=="!"){

      op="not";

  }

  pp="";

  continue;

}

if(i==3){

pp=postfix(pp);
```

```
      bb = eval(pp,finall);
      pp="";
      continue;
      }
    }
    else{
    pp = pp + l;
    }
 }
 if(op=="greater"){
    if(aa>bb){
       flag=true;
    }
    return flag;
 }
 else if(op=="less"){
    if(aa<bb){
       flag=true;
    }
    return flag;
 }
 else if(op=="greaterequal"){
    if(aa>=bb){
       flag=true;
    }
    return flag;
 }
 else if(op=="lessqual"){
    if(aa<=bb){
            flag=true;
        }
        return flag;
    }
    else if(op=="equal"){
         if(aa==bb){
            flag=true;
        }
        return flag;
    }
    else if(op=="not"){
        if(aa!=bb){
            flag=true;
        }
        return flag;
    }
   return flag;
}
void parser(map<int,map<pair<int,string>,string>,less<int>> ques,int n,vector<int>
check,map<string,int> &finall,int argc, char** argv){
stack<string> ss;
int flag;
auto x = ques.begin();
for(;x!=ques.end();x++){
auto expr = x->second;
flag = x->first;
auto vt = expr.begin();
string pp;
for(;vt!=expr.end();vt++){
```

```
//first if it is let and it is initilizing
//cout<<"checking the variable-->"<<(vt->first).second<<endl;
if(fg==1){
   break;
}
if((vt->first).second=="let"&&expr.size()==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 if it is print
else if((vt->first).second=="print"){
```

```
//cout<<"size of expr is"<<expr.size()<<endl;
if(expr.size()>=3){
  for(int i=1;i<=expr.size();i++){
        //cout<<"value of i is"<<i<<endl;
        if((vt->first).second=="print"){
        vt++;
        continue;
        }
        else if(vt->second=="direct"){
        cout<<(vt->first).second;
        vt++;
        continue;
        }
        else{
        anura = anura + (vt->first).second;
        if(i==expr.size()){
        //anura=anura+(vt->first).second;
        anura=postfix(anura);
        //cout<<"postfix is"<<anura<<endl;
        int kl = eval(anura,finall);
        cout<<kl<<endl;
        anura="";
        //kl=0;
        continue;
        }
        vt++;
        continue;
        }
  }
```

```
}
else{
for(int z=1;z<=2;z++){
if((vt->first).second=="print"){
vt++;
}
else {
   if((vt->second)=="direct"){
      //cout<<"yaha pe toh nahi aagya???";
      cout<<(vt->first).second<<endl;
      //cout<<"size of expr is"<<expr.size()<<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="";https://www.onlinegdb.com/#tab-stdout
    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 if it is let but it is for evaluation
else if((vt->first).second=="let"&&expr.size()>4){
for(int z=1;z<=expr.size();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==expr.size()){
      pp=postfix(pp);
      int kl = eval(pp,finall);
      auto top = ss.top();
      ss.pop();
      finall[top]=kl;
      continue;
  }
  vt++;
}
}
```

```
}
//fourth if it is if stattemtn
else if((vt->first).second=="if"){
    int lol = 0;
    string ap;
    int dd=0;
    int zt = 0;
    while(dd<=(expr.size())){
        dd++;
        if((vt->first).second=="if"){
            vt++;
            continue;
        }
        else if(vt->second!="comp"&&zt==0&&(vt->first).second!="then"){
             ap=ap+(vt->first).second;
             vt++;
             continue;
        }
        else if(vt->second=="comp"){
             ap=ap+" ";
             ap=ap+(vt->first).second;
             ap=ap+" ";
             vt++;
             continue;
        }
        else if((vt->first).second=="then"){
            zt=1;
            ap=ap+" ";
            if(letmecheck(ap,finall)==true){
```

```
          lol=1;
        }
        else{
           lol=0;
        }
        vt++;
        continue;
    }
    else if((vt->first).second=="print"&&lol==1){
    for(int z=1;z<=2;z++){
    if((vt->first).second=="print"){
    vt++;
    }
    else {
    if((vt->second)=="direct"){
        //cout<<"isit";
        cout<<(vt->first).second<<endl;
        vt++;
        continue;
    }
    else{
    cout<<finall[(vt->first).second]<<endl;
    vt++;
    continue;
    }
    }
    }
    }
else if((vt->first).second=="goto"&&lol==1){
    int dd=0;
    int ff = 0;
    while(dd<=2){
        dd++;
        if((vt->first).second=="goto"){
            vt++;
            continue;
        }
        if(vt->second=="value"){
            auto zzz= (vt->first).second;
            auto xxx = ques.begin();
            auto zll=xxx;
          for(;xxx!=ques.end();xxx++){
             auto eee = xxx->second;
             auto at = eee.begin();
            if(xxx->first==stoi(zzz)){
                x=zll;
                vt++;
                break;
            }
            zll = xxx;
            }
            }
        }
    }
        vt++;

}
}
```

```
//fifth if it is directly an goto statement
else if((vt->first).second=="goto"){
    int dd=0;
    int ff = 0;
    while(dd<=2){
        dd++;
        if((vt->first).second=="goto"){
            vt++;
            continue;
        }
        if(vt->second=="value"){
            auto zzz= (vt->first).second;
            auto xxx = ques.begin();
            auto zll=xxx;
          for(;xxx!=ques.end();xxx++){
             auto eee = xxx->second;
             auto at = eee.begin();
            if(xxx->first==stoi(zzz)){
                x=zll;
                vt++;
                break;
            }
            zll = xxx;
            }
            }
        }
}
//sixth(initilize x,y,z)
else if((vt->first).second=="integer"){
    int dd=0;
    while(dd<=expr.size()){
        dd++;
        if((vt->first).second=="integer"){
            vt++;
            continue;
        }
        else if(vt->second=="value"){
            finall[(vt->first).second];
        }
    }
}
//seventh
else if((vt->first).second=="input"){
    int dd=0;
    while(dd<=expr.size()){
        dd++;
        if((vt->first).second=="input"){
            vt++;
            continue;
        }
        else if(vt->second=="identifier"){
            ghgh++;
            finall[(vt->first).second]=stoi(argv[ghgh]);
            vt++;
        }

    }
}
```

```
/*          auto zyx = vt;
          zyx++;
          int rpm = stoi((zyx->first).second);
          gos.push(rpm);
*/
//eighth gosub
else if((vt->first).second=="gosub"){
    int dd=0;
    int ff = 0;
    while(dd<=2){
        dd++;
        if((vt->first).second=="gosub"){
            vt++;
            continue;
        }
        if(vt->second=="value"){
            auto zyx = x;
            zyx++;
            int rpm = (zyx->first);
            gos.push(rpm);
            auto zzz= (vt->first).second;
            auto xxx = ques.begin();
            auto zll=xxx;
          for(;xxx!=ques.end();xxx++){
             auto eee = xxx->second;
             auto at = eee.begin();
            if(xxx->first==stoi(zzz)){
                x=zll;
                vt++;
                break;
            }
            zll = xxx;
            }
            }
        }
}
//ninth if it is ret
else if((vt->first).second=="ret"){
    int temp = gos.top();
    gos.pop();
    vt++;
    int zzz= temp;
    auto xxx = ques.begin();
    auto zll=xxx;
  for(;xxx!=ques.end();xxx++){
     auto eee = xxx->second;
     auto at = eee.begin();
    if(xxx->first==zzz){
        x=zll;
        vt++;
        break;
    }
    zll = xxx;
    }
}
//10th
else if((vt->first).second=="end"){
    //cout<<"babrbarb"<<endl;
    x=ques.end();
    x++;
    //x=ques.end();
}
//11 th push
else if((vt->first).second=="push"){
    int dd=0;
    while(dd<=expr.size()){
        dd++;
        if((vt->first).second=="push"){
            vt++;
            continue;
        }
        else{
            sat = sat + (vt->first).second;
            if(dd==expr.size()){
                //cout<<sat<<endl;
                sat = postfix(sat);
                //cout<<sat<<endl;
                int ml = eval(sat,finall);
                user.push(ml);
                sat="";
                break;
            }
            vt++;
            }
        }
}
//12 th pop
```

```
else if((vt->first).second=="pop"){
  int dd=0;
  while(dd<=2){
    dd++;
    if((vt->first).second=="pop"){
      vt++;
      continue;
    }
    if(vt->second=="identifier"){
      int t = user.top();
      user.pop();
      finall[(vt->first).second]=t;
      vt++;
      break;
      }
    }
}

}


}
}
class memory{
  public:
  memory(map<int,string,less<int>> &help,string s,string &para){
    while(getline(cin,s)){
    auto first_token = s.substr(0, s.find(' '));
    s=s.substr(s.find_first_of(" ")+1);// removing the line number from the string
    s=s+" "; // giving blank spaces
```

```
        para=para+s+" ";
        help[stoi(first_token)]=s;
    }
    }
};
int main(int argc, char** argv) {
    map<int,map<pair<int,string>,string>,less<int>> ques;//this will map the line number with the lexical
analysis of that line
    //map<pair<int,string>,string> expr;
    map<int,string,less<int>> help;// this is basically used for storing the line number and the string
    int ll;
    string s;
    string para;
    vector<int> check;
    map<string,int> finall;
    //calling memory
    memory obj= memory(help,s,para);
    string word;
    int i=0;
    int z=0;
    int ffo=0;
    string zzz;
    string to;
    //
    //basically here i am tokenizing for difffernt tokenz
    ///lexer
    for(auto lmn:help){
    string paraa = lmn.second;
    for(auto x:paraa){
        if(x==' '){

            i++;

            if(ffo!=0&&word.find("\"")!=string::npos){

              to=to+word;

              ques[lmn.first].insert(make_pair(make_pair(i,to),"direct"));

              zzz="direct";

              ffo=0;

              word="";

              to="";

              continue;

            }

            if(ffo!=0){

              to=to+word+" ";

              i--;

              word="";

              continue;

            }

            if((word.find("\"")!=string::npos)&&ffo==0){

              to = to + word + " ";

              ffo=1;

              i--;

              word="";

              continue;

            }

            if((word.find('0') != string::npos || word.find('1') != string::npos ||word.find('2') != string::npos
||word.find('3') != string::npos ||word.find('4') != string::npos ||word.find('5') != string::npos
||word.find('6') != string::npos ||word.find('7') != string::npos ||word.find('8') != string::npos
||word.find('9') != std::string::npos)){

            if(zzz=="print"){

            ques[lmn.first].insert(make_pair(make_pair(i,word),"directto"));
}
}
if(word=="end"){
   ques[lmn.first].insert(make_pair(make_pair(i,word),"end"));
   zzz="end";
}
if(word=="integer"){
   ques[lmn.first].insert(make_pair(make_pair(i,word),"init"));
   zzz="init";
}
if(word=="input"){
   ques[lmn.first].insert(make_pair(make_pair(i,word),"input"));
   zzz="input";
}
if(word=="gosub"){
   ques[lmn.first].insert(make_pair(make_pair(i,word),"gosub"));
   zzz="gosub";
}
if(word=="ret"){
   ques[lmn.first].insert(make_pair(make_pair(i,word),"ret"));
   zzz="ret";
}
if(word=="pop" or word=="push"){
   ques[lmn.first].insert(make_pair(make_pair(i,word),"stack"));
   zzz="stack";
}
if(word.find_first_not_of("0123456789") == string::npos){
ques[lmn.first].insert(make_pair(make_pair(i,word),"value"));
zzz="value";
```

```
        }
        else if(word=="if"){
        ques[lmn.first].insert(make_pair(make_pair(i,word),"comparsion"));
        zzz="comparision";
        }
        else if(word=="goto"){
        ques[lmn.first].insert(make_pair(make_pair(i,word),"travel"));
        zzz="travel";
        }
        else if(word.find("\"")!=string::npos){
        ques[lmn.first].insert(make_pair(make_pair(i,word),"direct"));
        zzz="direct";
        }
        else
if(word=="LET"||word=="let"||word=="print"||word=="println"||word=="PRINT"||word=="PRINTLN"|
|word=="print"){
        if(word=="println"){
          word="print";
        }
        ques[lmn.first].insert(make_pair(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){
           ques[lmn.first].insert(make_pair(make_pair(i,word),"identifier"));
           zzz="identifier";
            }
           else if(word=="then"){
           ques[lmn.first].insert(make_pair(make_pair(i,word),"then"));
           zzz="then";
           }
           else if(word=="/"||word=="*"||word=="+"||word=="-"){
           ques[lmn.first].insert(make_pair(make_pair(i,word),"operator"));
           zzz="operator";
           }
           else if(word=="!"||word=="<"||word==">"){
           ques[lmn.first].insert(make_pair(make_pair(i,word),"comp"));
           zzz="comp";
           }
           else if(word=="="){
           ques[lmn.first].insert(make_pair(make_pair(i,word),"equalsign"));
           zzz="equalsign";
           }
           word="";
       continue;
   }
   word=word+x;
}
}
check[0]=0;
check.push_back((i-ll+1));
```

```
//    for(auto x:ques){
//    cout<<x.first<<"-->";
//    auto p = x.second;
//    auto mapp = x.second;
//    for(auto z:mapp){
//        auto zz = z.first;
//        cout<<"("<<zz.first<<",";
//        cout<<zz.second<<")";
//        auto qq = z.second;
//        cout<<"--->"<<qq<<endl;;
//    }
//   }
    // i=i+1;
    //basically sending to parser which will take care of everything
    parser(ques,i,check,finall,argc,argv);
    //cout<<finall.size()<<endl;
    for(auto x:finall){
        cout<<x.first<<"--->"<<x.second<<endl;
    }
    return 0;
}
```