INDEX

AIM.....2

OPERATIONS ALLOWED.....2

SYNTAX SHEET FOR BRACKET 3

SCOPE FOR DEVELOPMENT 6

STEPS INVOLVED IN THE INTERPRETATION 6

INTERMEDIATE CODE GENERATION FOR C++ 6

THE SOURCE CODE FOR LEXER.H 7

THE SOUCE CODE FOR EXECUTE.CPP 12

SCREEN OUTPUTS.....46

INTERPRETER

AIM: To develop a program that interprets and implements the given instructions in a language that has our own defined syntax and keywords .An interpreter transforms or interprets a high-level programming code into code that can be understood by the machine (machine code) or into an intermediate language that can be easily executed as well. In our project the programs interprets the input through c++ as an intermediate.

OPERATIONS ALLOWED:

The language allows the user to perform the following functions:

- Prints the given set of statements.
- Accepts variable of type character, decimal and integer
- Also accepts a sequence of characters(string).
- Checks or evaluates a given condition.
- Allows to execute iterative statements.
- Permits the usage of one dimensional array.

The syntax for the above operations are mentioned in the below syntax sheet.

SYNTAX SHEET FOR BRACKET :

•ALWAYS A STRING IS GIVEN WITHN DOUBLE QUOTES, A SINGLE CHARACTER OR LITERAL WITHIN SINGLE QUOTES!!

FOR DISPLAYING DATA:

out(....);

• The data displayed :

if an number or expression involving operations with variables and numbers or simply a variable, just put them inside out() without any quotes.

if a single character, put the character within ' ' (single quotes).

if a sentence or word, put them within " " (double quotes).

if a string variable, put them as varID# .

if a new line, put en.

• There can be a combination or sequence of the above separated by commas (,) .

FOR INPUTTING DATA:

in(....);

• The data inputted :

if a variable, just put the varID.

if a string, put varID# .

FOR COMMENTS:

cmt(ANYTHING YOU WISH);

DATATYPES:

• num //FOR INTEGER AND DECIMAL NUMBERS

• alpha //FOR ALPHABETS, SINGLE CHARACTERS

ARRAYS/STRINGS:

It should be given as varID#size during declaration and initialisation, and varID#index during usage of that element index.

Strings follow the same rules as a character array except there has to be a null character at the end of the string, which requires an extra size.If we dont provode that extra size it cannot be treated as a string.

Strings need not be given implicitly a null character at the end. It is automatically done by the compiler but that extra size has to be provided.

The array index or size , if a character(whose ASCII value will be taken) or an expression or another array's element, has to be given as varID#(expr) .

The array or string is numbered starting from 0 (zero) .

DECLARATION:

It is given as:

datatype varID_1,varID_2,.....;

• For array declaration and string declaration provide size too like: varID#(size) or varID#size as required .

INITIALISATION AND DECLARATION:

It is given as:

(FOR INITIALISATION): datatype varID_1=(expr),.....; //(...) needed only for expressions, not for single characters or single integer, decimal

(FOR DECLARATION): varID_1=(expr),.....; //(...) needed only for expressions, not for single characters or single integer, decimal

•For arrays, varID#(size)(value_1,value_2,.....) //As per number of values given, that many number of elements from the beginning is filled with the corresponding values. The number of values given should be less than or equal to size of array

•For strings, varID#(size)("string") //The number of characters must be one less than size of string

---

• CONTROL STRUCTURES:-

LOOP:

iter(condition){....} //{,} are necessary for even single statement loops

CONDITION STATEMENTS REPLACING IF,ELSE OF, AMD ELSE IN C++:

• One way:

check(condition){...}

• Two way:

check(condition){...}

orelse{...}

• N-way:

check(condition){...}

or(condition){...}

orelse{...}

• LOGICAL OPERATORS:-

1.OR ^

2.AND &

3.NOT !

• RELATIONAL OPERATORS:-

1.EQUAL TO ==

2.GREATER >

3.LESSER <

4.GREATER THAN OR EQUAL TO >=

5.LESSER THAN OR EQUAL TO <=

6.NOT EQUAL TO !=

• ALL OTHER RELATIONAL, LOGICAL AND ARITHMETIC OPERATORS SAME AS C++

• NO POST OR PRE- INCREMENT OR DECREMENT IS AVAILABLE

• MULTIPLE AND NESTED LOOPS AND CHECK'S ARE ALLOWED!!

• ANY OTHER METHOD OR SYNTAX ADOPTED MAY LEAD TO PROGRAM CRASHING!!

SCOPE FOR DEVELOPMENT :

The project can be fully developed by checking for all possible errors in the language syntax. Other features such as functions and multi dimensional array can also be developed.

STEPS INVOLVED IN THE INTERPRETATION:

Lexer:

The first step that we have done is to break the given set of statements into individual tokens. This task is done by lexer. The separated tokens are further processed and interpreted by the parser.

INTERMEDIATE CODE GENERATION FOR C++ :

The second step involves analyzing the tokens generated by the lexer .The tokens are analysed and changed into format that is executed using c++. It identifies the keywords and performs the corresponding operations.

THE SOURCE CODE FOR LEXER.H IS GIVEN BELOW:

```
1  #ifndef LEXER_H_INCLUDED
2  #define LEXER_H_INCLUDED
3
4  ///THIS CODE SEGMENT TOKENISES THE GIVEN SET OF INSTRUCTIONS AND MAKES
THE ANALYSIS
5  ///AND EXECUTION OF THE INSTRUCTIONS EASIER FOR THE SUBSEQUENT STEP
6  ///THIS CODE SEGMENT DOES THE ROLE OF LEXICAL ANALYSIS
7  #include<iostream>
8  #include<bits/stdc++.h>
9  #include<fstream>
10  #include<windows.h>
11  #include<string>
12
13  using namespace std;
14
15  string tokens;
16  vector<string> keyword;
17
18  int iskeyword(string a) { ///CHECKS IF THE GIVEN WORD IS A KEYWORD
19      ///KEYWORDS :
20      ///  iter  (while type)
21      ///  out
22      ///  in
23      ///  check
24      ///  or
25      ///  orelse
26      ///  num
27      ///  alpha
28      ///  string
29      ///  cmt
30
31      if(find(keyword.begin(),keyword.end(),a)!=keyword.end())
32          return 1;
33      else return 0;
34  }
35
36  bool isvariable(string t){ ///TO CHECK IF THE GIVEN WORD CAN BE A VARIABLE
37      long int c=0;
38      for(long int i=0;i<t.length();i++){
39          if(isalpha(t[i])||isdigit(t[i])||t[i]=='_') c++;
40      }
41      if(t!=""&&isalpha(t[0])&&c==t.length()) return 1;
42      else return 0;
43  }
44
45  string lexer(string filepath) { ///PERFORMS THE LEXICAL ANALYSIS
46      fstream fin(filepath,ios::in); ///THE INPUT IS TO BE TAKEN FROM THE FILEPATH
PROVIDED
47      fstream fout("errors.txt",ios::out|ios::ate);
48
49      keyword.push_back("iter");
50      keyword.push_back("out");
51      keyword.push_back("in");
52      keyword.push_back("check");
```

```
53      keyword.push_back("or");
54      keyword.push_back("orelse");
55      keyword.push_back("num");
56      keyword.push_back("alpha");
57      keyword.push_back("type");
58      keyword.push_back("string");
59      keyword.push_back("inc");
60      keyword.push_back("cmt");
61
62      char tmp;
63      string t;
64      fin.get(tmp);
65      long int cnt=0;
66
67      while(!fin.eof()) {
68        if(tmp=='%'||tmp=='&'||tmp=='^'||tmp=='!'||tmp=='.'||
            (isdigit(tmp)==1&&!isvariable(t))||
69          tmp=='>'||tmp=='<'||tmp=='+'||tmp=='-'||tmp=='*'||tmp=='/'
            ||tmp=='='||tmp=='-'||
70          tmp=='('||tmp==')'||tmp=='{'||tmp=='}'||tmp==';'||tmp==','
            ||tmp==':'||tmp=='\''||
71          tmp=='\"'||tmp=='\n'||tmp=='#'||tmp=='~') {
72             if(tmp==' ');
73             if(t==keyword[11]){
74                 char te='\0';
75                 int ob=0,cb=0;
76                 if(tmp=='(') ob++;
77                 do{
78                    fin.get(te);
79                    if(te=='(') ob++;
80                    else if(te==')') cb++;
81                 }while(ob!=cb);
82                 t="";
83                 tmp='\0';
84             }
85             if (iskeyword(t)==1&&cnt==0){
86                 tokens.append("RESERVED: ");
87                 tokens.append(t);
88                 tokens.append(" ");
89                 t="";
90             }
91             if(tmp=='~'){
92                 cnt++;
93             }
94             if(tmp=='\"'){
95                  string te;
96                  char c=0;
97                  while(c!='\"'){
98                     fin.get(c);
99                     if(c!='\"')
100                     te+=c;
101                 }
102                 te+=' ';
103                 tokens.append("STRING: ");
104                 tokens.append(te);
105                 t="";
106             }
```

```
107             else if(tmp=='\''){
108                 fin.get(tmp);
109                 tokens.append("LIT: ");
110                 tokens+=tmp;
111                 tokens+=" ";
112                 fin.get(tmp);
113                 tmp='\0';
114             }
115             else if(tmp=='.'||isdigit(tmp)==1){
116                 tokens.append("NUMB: ");
117                 while(isdigit(tmp)==1||tmp=='.'){
118                     tokens+=tmp;
119                     fin.get(tmp);
120                 }
121                 fin.unget();
122                 tmp='\0';
123                 tokens+=" ";
124             }
125             else if(t=="en"){
126                 tokens.append("en ");
127                 t="";
128             }
129             else if(t!=""){
130                 if(cnt==0){
131                   tokens.append("VARIABLE: ");
132                   tokens.append(t);
133                   tokens.append(" ");
134                   t="";
135                 }
136                 else if(cnt==1) {
137                   tokens.append("RETURNTYPE: ");
138                   tokens.append(t);
139                   tokens+=' ';
140                   t="";
141                   cnt++;
142                 }
143                 else if(cnt==2){
144                   tokens.append("FNNAME: ");
145                   tokens.append(t);
146                   tokens+=' ';
147                   t="";
148                   cnt=0;
149                 }
150             }
151             if(tmp==',') tokens.append(" ");
152             if(tmp=='(') {
153                 tokens.append("OPENB ");
154                 t="";
155             }
156             if(tmp==')') {
157                 tokens.append("CLOSEB ");
158                 t="";
159             }
160             if(tmp=='{') {
161                 tokens.append("OPENCB\n ");
162                 t="";
163             }
```

```
164                 if (tmp=='}') {
165                     tokens.append("CLOSECB\n ");
166                     t="";
167                 }
168                 if(tmp==';') tokens.append("EOLN\n ");
169                 if(tmp=='%') tokens.append("OP: % ");
170                 if(tmp=='!'){
171                     char ch=tmp;
172                     fin.get(tmp);
173                     if(tmp=='=') {
174                        tokens.append("RELATION: ");
175                        string a="";
176                        a+=ch;
177                        a.append("= ");
178                        tokens.append(a);
179                     }
180                     else{
181                        fin.unget();
182                        tokens.append("NEGATION ");
183                     }
184                 }
185                 if(tmp=='='){
186                     char ch=tmp;
187                     fin.get(tmp);
188                     if(ch==tmp) {
189                       tokens.append("RELATION: ");
190                       tokens.append("== ");
191                     }
192                     else{
193                       fin.unget();
194                       tokens.append("ASSIGN ");
195                     }
196                 }
197                 if(tmp=='#'){
198                     tokens.append("INDEX: ");
199                     string a="";
200                     while(fin.get(tmp)&&isdigit(tmp)) a+=tmp;
201                     tokens.append(a);
202                     if(a!="")
203                     tokens+=' ';
204                     tmp='\0';
205                     fin.unget();
206                 }
207                 if(tmp=='&') tokens.append("AND ");
208                 if(tmp=='^') tokens.append("OR ");
209                 if(tmp=='>'||tmp=='<'){
210                     char ch=tmp;
211                     fin.get(tmp);
212                     if(tmp=='=') {
213                         tokens.append("RELATION: ");
214                         string a="";
215                         a+=ch;
216                         a.append("= ");
217                         tokens.append(a);
218                     }
219                     else{
220                         fin.unget();
```

```
                    tokens.append("RELATION: ");
                    string a;
                    a+=ch;
                    a+=" ";
                    tokens.append(a);
                }
            }
            if(tmp=='+'||tmp=='-'||tmp=='*'||tmp=='/'){
                tokens.append("OP: ");
                string a="";
                a+=tmp;
                tokens.append(a);
                tokens.append(" ");
            }
        }
        else{
            if(tmp!='\n'&&tmp!=' '){
               t+=tmp;
            }
        }
        fin.get(tmp);
    }
    fin.close();
    fout.close();
    return tokens;
}

string puttotxt(string filepath){
    string a=lexer(filepath);
    return a;
}

#endif
```

THE SOUCE CODE FOR EXECUTE.CPP IS GIVEN BELOW:

```
1  /// PROJECT BRACKET((.))
2  ///PROJECT DONE BY ABISHEK.S AND AHMED SHAMSUDEEN, class 12-A
3  ///THIS CODE SEGMENT DOES THE ROLE OF INTERMEDIATE CODE GENERATION
4  ///THAT CAN BE PROCESSED BY C++
5
6  #include"lexer.h"
7  #include<iostream>
8  #include<fstream>
9  #include<bits/stdc++.h>
10  #include<string>
11  using namespace std;
12
13  ///FILE OBJECT TO THE TEMPORARILY CREATED LEXER.TXT
14  fstream fin;
15
16  ///CLASS DEFINITIONS.....
17  class var{
18      public:
19      string name;
20      string dat;
21      string val;
22      var(string a,string b,string c){
23          name=a;
24          dat=b;
25          val=c;
26
27      }
28  };
29  vector<var> vec;
30
31  ///FUNCTIONS DEFINITIONS
32  double pow10(int);
33  double tonum(string,int,int);
34  double tonumdeci(string,int,int);
35  string toalpha(string);
36  string tonumber(string);
37  long int search(string);
38  int prior(char);
39  bool find_no_op(string);
40  string arrexp();
41  string arrexp(stringstream&);
42  string main_fn(string);
43  void iter();
44  void iter(stringstream&,streampos);
45  int check(int);
46  int check(int cond,stringstream&,streampos);
47  string infix(string);
48  void in();
49  void in(stringstream&,streampos);
50  void out();
51  void out(stringstream&,streampos);
52  void variable(string,stringstream&,streampos);
53  void variable(string);
54
```

```
55  ///FUNCTION DEFINITIONS
56  double pow10(int j) { ///FOR GETTING 10 RAISED TO AN INTEGER
57      double v = 1;
58      if (j>=0) {
59          for(int i=0;i<j;i++) {
60              v*=10;
61          }
62      }
63      else {
64          for(int i=0;i>j;i--)
65              v/=10;
66      }
67      return v;
68  }
69
70  double tonum(string b, int s, int e) { ///FOR CONVERTING A NON-DECIMAL NUMBER IN
                                             STRING FORM TO DOUBLE TYPE
71      double v = 0;int j = 0;
72      for (int i=e;i>=s;i--,j++) {
73          v+=(b[i]-48)*pow10(j);
74      }
75      return v;
76  }
77
78  double tonumdeci(string b, int s, int e) { ///FOR CONVERTING A DECIMAL NUMBER IN
                                                 STRING FORM TO DOUBLE TYPE
79      double v = 0;int j = 0, i = e, k;
80      for(;b[i]!='.';i--);
81      k=i;
82      for(i=k-1;i>=s;i--,j++) {
83          v+=(b[i]-48)*pow10(j);
84      }
85      j=-1;
86      for(i=k+1;i<=e;i++,j--)
87          v+=(b[i]-48)*pow10(j);
88      return v;
89  }
90  string toalpha(string tmp){ ///FOR CONVERTING THE ASCII VALUE TO THE CHARACTER
91      double n=tonum(tmp,0,tmp.length()-1);
92      char s=(char)n;
93      string a;
94      a+=s;
95      return a;
96  }
97  string tonumber(string tmp){ ///FOR CONVERTING A CHARACTER TO ITS ASCII EQUIVALENT
                                   IN STRING FORM
98      char s=tmp[0];
99      double n=(double)s;
100      stringstream ss;
101      ss << n;
102      string a=ss.str();
103      return a;
104  }
105  long int search(string nam){ ///FOR RETURNING THE INDEX OF A VARIABLE IN THE
                                   VARIABLE ARRAY
106      long int index=0,c=0;
107      for(long int it=vec.size()-1;it>=0&&(vec[it].name!=""||c==0);it--){
```

```
108         if(vec[it].name==nam){index=it;c++;}
109     }
110     return index;
111 }
112 int prior(char v) { ///FOR RETURNING PRIORITY OF A GIVEN OPERATOR
113     int i = 0;
114     switch (v) {
115     case '-':i = 5;
116     break;
117     case '+':i = 5;
118     break;
119     case '*':i = 6;
120     break;
121     case '/':i = 6;
122     break;
123     case '%':i = 6;
124     break;
125     case '!':i = 7;
126     break;
127     case '&':i = 2;
128     break;
129     case '^':i = 1;
130     break;
131     case '@':i = 4;  ///@ replacing <= for infix function alone
132     break;
133     case '<':i = 4;
134     break;
135     case '_':i = 4;  ///_ replacing >= for infix function alone
136     break;
137     case '>':i = 4;
138     break;
139     case '=':i = 3;  ///= replacing == for infix function alone
140     break;
141     case '#':i = 3;  ///# replacing != for infix function alone
142     }
143     return i;
144 }
145
146 bool find_no_op(string st){ ///CHECKS FOR ANY OPERATOR IN A GIVEN STRING
147     bool op=0;
148     op=(st.find('+')==string::npos)&&(st.find('-')==string::npos)
            &&(st.find('/')==string::npos)&&
149         (st.find('*')==string::npos)&&(st.find('^')==string::npos)
            &&(st.find('=')==string::npos)&&
150         (st.find('>')==string::npos)&&(st.find('<')==string::npos)
            &&(st.find('!')==string::npos)&&
151         (st.find('&')==string::npos);
152     return op;
153 }
154
155 string arrexp(){ ///TO EVALUATE A GIVEN EXPRESSION INVOLVING ARRAY INDEX
156     string tmp,str1;
157     int ob=1,cb=0;
158     str1+='(';
159     do{
160         getline(fin,tmp,' ');
161         if(tmp=="RELATION:"){
```

```
162             getline(fin,tmp,' ');
163             str1+=tmp;
164         }
165         else if(tmp=="OPENB") {
166             str1+='(';
167             ob++;
168         }
169         else if(tmp=="INDEX:"){
170             getline(fin,tmp,' ');
171             str1+='#';
172             if(tmp=="VARIABLE:")
173                  getline(fin,tmp,' ');
174             else if(tmp=="OPENB") tmp=arrexp();
175             str1+=tmp;
176         }
177         else if(tmp=="CLOSEB"){
178             str1+=')';
179             cb++;
180         }
181         else if(tmp=="VARIABLE:"){
182             getline(fin,tmp,' ');
183             str1+=tmp;
184         }
185         else if(tmp=="OP:"){
186             getline(fin,tmp,' ');
187             str1+=tmp;
188         }
189         else if(tmp=="NUMB:"){
190             getline(fin,tmp,' ');
191             str1+=tmp;
192             str1+=' ';
193         }
194         else if(tmp=="LIT:"){
195             getline(fin,tmp,' ');
196             if(tmp=="") str1+=tonumber(" ");
197             else
198             str1+=tonumber(tmp);
199             str1+=' ';
200         }
201         else if(tmp=="NEGATION")
202             str1+='!';
203         else if(tmp=="AND")
204             str1+='&';
205         else if(tmp=="OR")
206             str1+='^';
207     }while(ob!=cb);
208     return infix(str1);
209 }
210
211 string arrexp(stringstream& fin){ ///OVERLOADED FOR A NESTED STATEMENT BLOCK
212     string tmp,str1;
213     int ob=1,cb=0;
214     str1+='(';
215     do{
216         getline(fin,tmp,' ');
217         if(tmp=="RELATION:"){
218             getline(fin,tmp,' ');
```

```
219             str1+=tmp;
220         }
221         else if(tmp=="OPENB") {
222             str1+='(';
223             ob++;
224         }
225         else if(tmp=="INDEX:"){
226             getline(fin,tmp,' ');
227             str1+='#';
228             if(tmp=="VARIABLE:")
229                  getline(fin,tmp,' ');
230             else if(tmp=="OPENB") tmp=arrexp();
231             str1+=tmp;
232         }
233         else if(tmp=="CLOSEB"){
234             str1+=')';
235             cb++;
236         }
237         else if(tmp=="VARIABLE:"){
238             getline(fin,tmp,' ');
239             str1+=tmp;
240         }
241         else if(tmp=="OP:"){
242             getline(fin,tmp,' ');
243             str1+=tmp;
244         }
245         else if(tmp=="NUMB:"){
246             getline(fin,tmp,' ');
247             str1+=tmp;
248             str1+=' ';
249         }
250         else if(tmp=="LIT:"){
251             getline(fin,tmp,' ');
252             if(tmp=="") str1+=tonumber(" ");
253             else
254             str1+=tonumber(tmp);
255             str1+=' ';
256         }
257         else if(tmp=="NEGATION")
258             str1+='!';
259         else if(tmp=="AND")
260             str1+='&';
261         else if(tmp=="OR")
262             str1+='^';
263     }while(ob!=cb);
264     return infix(str1);
265 }
266
267 string main_fn(string a){ ///MAIN FUNCTION FOR NESTED STATEMENT BLOCK
268     stringstream inp;
269     inp<<a;
270     string str;
271     var openb("","","");
272     vec.push_back(openb);
273     long int cb=0,ob=1;
274     long int condi=0,gap=0;
275     while(ob!=cb&&getline(inp,str,' ')){
```

```
276         if(str=="OPENCB\n"){
277             gap++;
278             ob++;
279             vec.push_back(openb);
280         }
281         else if(str=="CLOSECB\n"){
282           gap++;
283           cb++;
284           if(ob!=cb){
285           for(long int it=vec.size()-1;vec[it].name!="";it--){
286                 vec.pop_back();
287           }
288           vec.pop_back();
289
290           }
291         }
292         else if(str=="EOLN\n")gap++;
293         else if(str=="en") {cout<<'\n';gap++;}
294         else if(str=="RESERVED:"){
295           str="";
296           getline(inp,str,' ');
297           if(str=="out") {
298             streampos posi=inp.tellg();
299             out(inp,posi);
300           }
301           else if(str=="in") {
302             gap++;
303             streampos posi=inp.tellg();
304             in(inp,posi);
305           }
306           else if(str=="alpha"||str=="num") {
307             gap++;
308             streampos posi=inp.tellg();
309             variable(str,inp,posi);
310           }
311           else if(str=="iter") {
312             gap++;
313             streampos posi=inp.tellg();
314             iter(inp,posi);
315           }
316           else if(str=="check"||str=="or"||str=="orelse"){
317             streampos posi=inp.tellg();
318             if(str=="check") gap++;
319             if(gap!=0){condi=0;gap=0;}
320             if(str=="orelse"){
321                 int ob=1,cb=0;
322                 getline(inp,str,' ');
323                 string st="";
324                 while(ob!=cb&&getline(inp,str,' ')){
325                     if(str=="OPENCB\n") ob++;
326                     else if(str=="CLOSECB\n") cb++;
327                     st+=str;
328                     st+=' ';
329                 }
330                 if(condi==0&&gap==0)
331                 main_fn(st);
332           }
```

```
333             else if(str=="or"){
334                 condi+=check(condi,inp,posi);
335             }
336             else {
337                 condi+=check(condi,inp,posi);
338             }
339           }
340         }
341         else if(str=="VARIABLE:"){
342             long int ind=0;
343             gap++;
344             string v;
345             getline(inp,str,' ');
346             long int n=search(str),a;
347             getline(inp,str,' ');
348             if(str=="INDEX:"){
349                 getline(inp,str,' ');
350                 if(str=="VARIABLE:") {
351                     getline(inp,str,' ');
352                     long int n1=search(str);
353                     ind=tonum(vec[n1].val,0,vec[n1].val.length()-1);
354                 }
355                 else if(str=="OPENB") {str=arrexp(inp);
                                           ind=tonum(str,0,str.length()-1);}
356                 else
357                 ind=tonum(str,0,str.length()-1);
358                 getline(inp,str,' ');
359             }
360             if(str=="ASSIGN"){
361             while(getline(inp,str,' ')&&str!="EOLN\n"){
362                if(str=="VARIABLE:"){
363                  getline(inp,str,' ');
364                  v+=str;
365                }
366                else if(str=="RELATION:"){
367                  getline(inp,str,' ');
368                  v+=str;
369                }
370                else if(str=="OP:"){
371                  getline(inp,str,' ');
372                  v+=str;
373                }
374                else if(str=="INDEX:"){
375                  getline(inp,str,' ');
376                  v+='#';
377                  if(str=="VARIABLE:")
378                     getline(inp,str,' ');
379                  else if(str=="OPENB") str=arrexp(inp);
380                  v+=str;
381                }
382                else if(str=="NEGATION")
383                   str+='!';
384                else if(str=="AND")
385                   str+='&';
386                else if(str=="OR")
387                   str+='^';
388                else if(str=="CLOSEB") v+=')';
```

```
                else if(str=="OPENB") v+='(';
                else{
                 if(vec[n].dat=="alpha"){
                    if(str=="LIT:"){
                       getline(inp,str,' ');
                       if(str=="") v+=tonumber(" ");
                       else
                       v+=tonumber(str);
                       v+=' ';
                    }
                    else if(str=="NUMB:"){
                      getline(inp,str,' ');
                      v+=(str);
                      v+=' ';
                    }
                 }
                 else if(vec[n].dat=="num"){
                    if(str=="NUMB:"){
                       getline(inp,str,' ');
                       v+=str;
                       v+=' ';
                    }
                    else if(str=="LIT:"){
                       getline(inp,str,' ');
                       if(str=="") v+=tonumber(" ");
                       else
                       v+=tonumber(str);
                       v+=' ';
                    }
                 }
              }
           }
        }
        if(vec[n+ind].dat=="alpha")
          vec[n+ind].val=toalpha(infix(v));
        else {
              vec[n+ind].val=infix(v);
              if(vec[n+ind].val=="") vec[n].val="0";
        }
       }
     }
     return str;
}

void iter(){ ///PROCESSES AN ITERATIVE STATEMENT BLOCK
     string tmp,str1,str2;
     getline(fin,tmp,' ');
     str1+='(';
     long int ob=1,cb=0;
     while(ob!=cb&&getline(fin,tmp,' ')){
        if(tmp=="RELATION:") {
            getline(fin,tmp,' ');
            str1+=tmp;
        }
        else if(tmp=="OPENB") {
            str1+='(';
            ob++;
```

```
446         }
447         else if(tmp=="INDEX:") {
448              getline(fin,tmp,' ');
449              str1+='#';
450              if(tmp=="VARIABLE:")
451                  getline(fin,tmp,' ');
452              else if(tmp=="OPENB") tmp=arrexp();
453              str1+=tmp;
454         }
455         else if(tmp=="CLOSEB") {
456             str1+=')';
457             cb++;
458         }
459         else if(tmp=="VARIABLE:"){
460             getline(fin,tmp,' ');
461             str1+=tmp;
462         }
463         else if(tmp=="OP:"){
464             getline(fin,tmp,' ');
465             str1+=tmp;
466         }
467         else if(tmp=="NUMB:"){
468             getline(fin,tmp,' ');
469             str1+=tmp;
470             str1+=' ';
471         }
472         else if(tmp=="LIT:"){
473             getline(fin,tmp,' ');
474             if(tmp=="") str1+=tonumber(" ");
475             else
476             str1+=tonumber(tmp);
477             str1+=' ';
478         }
479         else if(tmp=="NEGATION")
480             str1+='!';
481         else if(tmp=="AND")
482             str1+='&';
483         else if(tmp=="OR")
484             str1+='^';
485     }
486     getline(fin,tmp,' ');
487     ob=1,cb=0;
488     while(ob!=cb&&getline(fin,tmp,' ')){
489          str2+=tmp;
490          if(tmp=="OPENCB\n") ob++;
491          else if(tmp=="CLOSECB\n") cb++;
492          str2+=' ';
493     }
494     stringstream ss;
495     ss<<infix(str1);
496     int res;
497     ss>>res;
498     while(res!=0){
499         main_fn(str2);
500         stringstream ss;
501         ss<<infix(str1);
502         res=0;
```

```
503          ss>>res;
504      }
505  }
506
507  void iter(stringstream& fin,streampos posi){ ///OVERLOADED FOR NESTED STATEMENT
                                                    BLOCK
508      fin.seekg(posi,ios::beg);
509      string tmp,str1,str2;
510      getline(fin,tmp,' ');
511      str1+='(';
512      long int ob=1,cb=0;
513      while(ob!=cb&&getline(fin,tmp,' ')){
514          if(tmp=="RELATION:"){
515             getline(fin,tmp,' ');
516             str1+=tmp;
517         }
518         else if(tmp=="OPENB") {
519             str1+='(';
520             ob++;
521         }
522         else if(tmp=="CLOSEB"){
523             str1+=')';
524             cb++;
525         }
526         else if(tmp=="INDEX:"){
527             getline(fin,tmp,' ');
528             str1+='#';
529             if(tmp=="VARIABLE:")
530                  getline(fin,tmp,' ');
531             else if(tmp=="OPENB") tmp=arrexp(fin);
532             str1+=tmp;
533         }
534         else if(tmp=="VARIABLE:"){
535             getline(fin,tmp,' ');
536             str1+=tmp;
537         }
538         else if(tmp=="OP:"){
539             getline(fin,tmp,' ');
540             str1+=tmp;
541         }
542         else if(tmp=="NUMB:"){
543             getline(fin,tmp,' ');
544             str1+=tmp;
545             str1+=' ';
546         }
547         else if(tmp=="LIT:"){
548             getline(fin,tmp,' ');
549             if(tmp=="") str1+=tonumber(" ");
550             else
551             str1+=tonumber(tmp);
552             str1+=' ';
553         }
554         else if(tmp=="NEGATION")
555             str1+='!';
556         else if(tmp=="AND")
557             str1+='&';
558         else if(tmp=="OR")
```

```
559             str1+='^';
560         }
561         getline(fin,tmp,' ');
562         ob=1,cb=0;
563         while(ob!=cb&&getline(fin,tmp,' ')){
564             str2+=tmp;
565             if(tmp=="OPENCB\n") ob++;
566             else if(tmp=="CLOSECB\n") cb++;
567             str2+=' ';
568         }
569         stringstream ss;
570         ss<<infix(str1);
571         int res;
572         ss>>res;
573         while(res!=0){
574             main_fn(str2);
575             stringstream ss;
576             ss<<infix(str1);
577             res=0;;
578             ss>>res;
579         }
580     }
581
582     int check(int cond){ ///PROCESSES A CONDITIONAL STATEMENT BLOCK
583         string tmp,str1,str2;
584         getline(fin,tmp,' ');
585         str1+='(';
586         long int ob=1,cb=0;
587         while(ob!=cb&&getline(fin,tmp,' ')) {
588             if(tmp=="RELATION:"){
589                 getline(fin,tmp,' ');
590                 str1+=tmp;
591             }
592             else if(tmp=="OPENB") {
593                 str1+='(';
594                 ob++;
595             }
596             else if(tmp=="INDEX:"){
597                 getline(fin,tmp,' ');
598                 str1+='#';
599                 if(tmp=="VARIABLE:")
600                     getline(fin,tmp,' ');
601                 else if(tmp=="OPENB") tmp=arrexp();
602                 str1+=tmp;
603             }
604             else if(tmp=="CLOSEB"){
605                 str1+=')';
606                 cb++;
607             }
608             else if(tmp=="VARIABLE:"){
609                 getline(fin,tmp,' ');
610                 str1+=tmp;
611             }
612             else if(tmp=="OP:"){
613                 getline(fin,tmp,' ');
614                 str1+=tmp;
615             }
```

```
616         else if(tmp=="NUMB:"){
617             getline(fin,tmp,' ');
618             str1+=tmp;
619             str1+=' ';
620         }
621         else if(tmp=="LIT:"){
622             getline(fin,tmp,' ');
623             if(tmp=="") str1+=tonumber(" ");
624             else
625             str1+=tonumber(tmp);
626             str1+=' ';
627         }
628         else if(tmp=="NEGATION")
629             str1+='!';
630         else if(tmp=="AND")
631             str1+='&';
632         else if(tmp=="OR")
633             str1+='^';
634     }
635     getline(fin,tmp,' ');
636     ob=1,cb=0;
637     while(ob!=cb&&getline(fin,tmp,' ')){
638         str2+=tmp;
639         if(tmp=="OPENCB\n") ob++;
640         else if(tmp=="CLOSECB\n") cb++;
641         str2+=' ';
642     }
643     stringstream ss;
644     ss<<infix(str1);
645     int res;
646     ss>>res;
647     if(res!=0&&cond==0){
648         main_fn(str2);
649     }
650     return res;
651 }
652
653 int check(int cond,stringstream&fin,streampos posi){ ///OVERLOADED FOR NESTED
                                                          STATEMENT BLOCK
654     fin.seekg(posi,ios::beg);
655     string tmp,str1,str2;
656     getline(fin,tmp,' ');
657     str1+='(';
658     long int ob=1,cb=0;
659     while(ob!=cb&&getline(fin,tmp,' ')) {
660         if(tmp=="RELATION:"){
661             getline(fin,tmp,' ');
662             str1+=tmp;
663         }
664         else if(tmp=="OPENB") {
665             str1+='(';
666             ob++;
667         }
668         else if(tmp=="CLOSEB"){
669             str1+=')';
670             cb++;
671         }
```

```
672         else if(tmp=="INDEX:"){
673              getline(fin,tmp,' ');
674              str1+='#';
675              if(tmp=="VARIABLE:")
676                  getline(fin,tmp,' ');
677              else if(tmp=="OPENB") tmp=arrexp(fin);
678              str1+=tmp;
679         }
680         else if(tmp=="VARIABLE:"){
681             getline(fin,tmp,' ');
682             str1+=tmp;
683         }
684         else if(tmp=="OP:"){
685             getline(fin,tmp,' ');
686             str1+=tmp;
687         }
688         else if(tmp=="NUMB:"){
689             getline(fin,tmp,' ');
690             str1+=tmp;
691             str1+=' ';
692         }
693         else if(tmp=="LIT:"){
694             getline(fin,tmp,' ');
695             if(tmp=="") str1+=tonumber(" ");
696             else
697             str1+=tonumber(tmp);
698             str1+=' ';
699         }
700         else if(tmp=="NEGATION")
701             str1+='!';
702         else if(tmp=="AND")
703             str1+='&';
704         else if(tmp=="OR")
705             str1+='^';
706     }
707     getline(fin,tmp,' ');
708     ob=1,cb=0;
709     while(ob!=cb&&getline(fin,tmp,' ')){
710          str2+=tmp;
711          if(tmp=="OPENCB\n") ob++;
712          else if(tmp=="CLOSECB\n") cb++;
713          str2+=' ';
714     }
715     stringstream ss;
716     ss<<infix(str1);
717     int res;
718     ss>>res;
719     if(res!=0&&cond==0){
720         main_fn(str2);
721     }
722     return res;
723 }
724
725 string infix(string a){ ///EVALUATES AN INFIX EXPRESSION
726     vector<char> c;
727     string b;
728     long int l = a.length();
```

```
729     if(a[l-1]!='+'&&a[l-1]!='-'&&a[l-1]!='*'&&a[l-1]!='/'&&a[l-1]!='^'&&
            a[l-1]!='&'&&a[l-1]!='!'&&a[l-1]!='='&&a[l-1]!='>'&&a[l-1]!='<'){
730     a.push_back(')');
731     c.push_back('(');
732     for (long int i = 0;i < (l + 1);i++) {
733         if (a[i] == ' ' ||( a[i] == '.'&&!(isdigit(a[i+1]))));
734         else if ((a[i] >= '0'&&a[i] <= '9')||a[i]=='.') {
735             if(a[i]=='.') b.push_back('0');
736             b.push_back(a[i]);
737             while ((a[i + 1] >= '0'&&a[i + 1] <= '9') || (a[i + 1] == '.'&&
                        a[i + 2] >= '0'&&a[i + 2] <= '9')) {
738                 b+=a[++i];
739             }
740             b+=' ';
741         }
742         else if (isalpha(a[i])) {
743             string abc,def;
744             int cnt=0,ind=0;
745             while(isalpha(a[i])){
746                 abc+=a[i];
747                 i++;
748             }
749             i--;
750             if(a[i+1]=='#') cnt++;
751
752             if(cnt!=0){
753                 i+=2;
754                 int start=i;
755                 while(isdigit(a[i])||isalpha(a[i])){
756                    def+=a[i];
757                    i++;
758                 }
759                 i--;
760                 if(isalpha(a[start])) {
761                     long int n1=search(def);
762                     ind=tonum(vec[n1].val,0,vec[n1].val.length()-1);
763                     }
764                 else
765                 ind=tonum(def,0,def.length()-1);
766             }
767             long int n=search(abc);
768             if(vec[n+ind].dat=="alpha")
769             b+=tonumber(vec[n+ind].val);
770             else b+=vec[n+ind].val;
771             b+=' ';
772         }
773         else if (a[i] == ')') {
774             char v = 0;
775             while (v != '(') {
776                 v=c[c.size()-1];
777                 c.pop_back();
778                 if (v != '(') {
779                     b+= v;
780                     b+=' ';
781                 }
782             }
783         }
```

```
784             else if (a[i] == '('){
785                 c.push_back(a[i]);
786                 }
787             else {
788                 int cnt=0;
789                 char cd=a[i];
790                 if((a[i]=='>'||a[i]=='<'||a[i]=='='||a[i]=='!')&&a[i+1]=='=')
                        {cnt++;i++;}
791                 if(cnt!=0){
792                     if(a[i-1]=='>') cd='_';
793                     else if(a[i-1]=='<') cd='@';
794                     else if(a[i-1]=='!') cd='#';
795                 }
796                 if (prior(cd) <= prior(c.back())) {
797                     while (prior(cd) <= prior(c.back())) {
798                         char v = 0;
799                         v=c[c.size()-1];
800                         c.pop_back();
801                         b+=v;
802                         b+=' ';
803                     }
804                     c.push_back(cd);
805                 }
806                 else
807                     c.push_back(cd);
808             }
809         }
810         }
811         else b=a;
812         vector<double> d;
813         for(int i = 0;b[i] != 0;i++) {
814             if (b[i] == ' ');
815             else if ((b[i] >= '0'&&b[i] <= '9')) {
816                 int j = i, k = 0;
817                 for (;b[i] != ' ';i++)
818                     if (b[i] == '.')
819                         k++;
820                 double v;
821                 if (k>0)
822                     v = tonumdeci(b,j, i - 1);
823                 else
824                     v = tonum(b, j, i - 1);
825                 d.push_back(v);
826             }
827             else if(isalpha(b[i])) {d.push_back((double)b[i]);}
828             else {
829                 double v1, v2;
830                 v2=d[d.size()-1];
831                 d.pop_back();
832                 if(d.size()==0) {v1=0;}
833                 else{
834                 v1=d[d.size()-1];
835                 d.pop_back();
836                 }
837                 switch (b[i]) {
838                 case'+':d.push_back(v1 + v2);
839                     break;
```

```
840                 case'-':d.push_back(v1 - v2);
841                     break;
842                 case'*':d.push_back(v1*v2);
843                     break;
844                 case'/':d.push_back(v1 / v2);
845                     break;
846                 case'%':d.push_back(int(v1) % int(v2));
847                     break;
848                 case '#':{
849                     bool res=v1!=v2;
850                     d.push_back((double)res);
851                 }
852                 break;
853                 case'!':{
854                     bool res=!v2;
855                     d.push_back(v1);
856                     d.push_back((double)res);
857                 }
858                 break;
859                 case '_':{
860                     bool res=v1>=v2;
861                     d.push_back((double)res);
862                 }
863                 break;
864                 case '>':{
865                     bool res=v1>v2;
866                     d.push_back((double)res);
867                 }
868                 break;
869                 case '@':{
870                     bool res=v1<=v2;
871                     d.push_back((double)res);
872                 }
873                 break;
874                 case '<':{
875                     bool res=v1<v2;
876                     d.push_back((double)res);
877                 }
878                 break;
879                 case '=':{
880                     bool res=v1==v2;
881                     d.push_back((double)res);
882                 }
883                 break;
884                 case '&':{
885                     bool res=v1&&v2;
886                     d.push_back((double)res);
887                 }
888                 break;
889                 case '^':{
890                     bool res=v1||v2;
891                     d.push_back((double)res);
892                 }
893                 break;
894                 }
895             }
896         }
```

```
897       double result=0;
898       if(d.size()!=0){
899       result=d[d.size()-1];
900       d.pop_back();
901       }
902       stringstream ss;
903       ss << result;
904       string a=ss.str();
905       return a;
906   }
907
908   void in(){ ///PROCESSES AN INPUT STATEMENT
909       string tmp="";
910       int chh=0,ind=0,str=0;
911       while(tmp!="EOLN\n"){
912           if(chh==0) getline(fin,tmp,' ');
913           if(tmp=="VARIABLE:"){
914               getline(fin,tmp,' ');
915               long int n=search(tmp);
916               getline(fin,tmp,' ');
917               chh++;
918               if(tmp=="INDEX:") {
919                   chh=0;
920                   getline(fin,tmp,' ');
921                   if(tmp=="VARIABLE:")
922                       getline(fin,tmp,' ');
923                   else if(tmp=="OPENB") tmp=arrexp();
924                   else if(tmp==""||tmp=="CLOSEB"){
925                      str++;
926                   }
927                   if(str==0){
928                   if(isdigit(tmp[0]))
929                   ind=tonum(tmp,0,tmp.length()-1);
930                   else{
931                       long int n1=search(tmp);
932                       ind=tonum(vec[n1].val,0,vec[n1].val.length()-1);
933                   }
934                   }
935               }
936               string value;
937               cin>>value;
938               if(str==0){
939               if(vec[n+ind].dat=="alpha") {
940                   if(isalpha(value[0]))
941                      vec[n+ind].val=value;
942                   else vec[n+ind].val=toalpha(value);
943               }
944               else if(vec[n+ind].dat=="num"){
945                   if(isalpha(value[0]))
946                       vec[n+ind].val=tonumber(value);
947                   else {
948                     if(value[0]=='.'){ vec[n+ind].val+='0';
949                     vec[n+ind].val+=value;
950                     }
951                     else vec[n+ind].val=value;
952                   }
953               }
```

```
954                 }
955                 else{
956                     for(long int i=0;i<value.length();i++){
957                         vec[n].val=value[i];
958                         n++;
959                     }
960                 }
961             }
962             else chh=0;
963             ind=0;
964             str=0;
965         }
966     }
967
968     void in(stringstream& sin,streampos posi){ ///OVERLOADED FOR NESTED STATEMENT
                                                    BLOCK
969         sin.seekg(posi,ios::beg);
970         string tmp;
971         int chh=0,ind=0,str=0;
972         while(tmp!="EOLN\n"){
973             if(chh==0)getline(sin,tmp,' ');
974             if(tmp=="VARIABLE:"){
975                 getline(sin,tmp,' ');
976                 long int n=search(tmp);
977                 chh++;
978                 getline(sin,tmp,' ');
979                 if(tmp=="INDEX:") {
980                     chh=0;
981                     getline(sin,tmp,' ');
982                     if(tmp=="VARIABLE:")
983                         getline(sin,tmp,' ');
984                     else if(tmp=="OPENB") tmp=arrexp(sin);
985                     else if(tmp==""||tmp=="CLOSEB") str++;
986                     if(str==0){
987                     if(isdigit(tmp[0]))
988                     ind=tonum(tmp,0,tmp.length()-1);
989                     else{
990                         long int n1=search(tmp);
991                         ind=tonum(vec[n1].val,0,vec[n1].val.length()-1);
992                     }
993                     }
994                 }
995                 string value;
996                 cin>>value;
997                 if(str==0){
998                 if(vec[n+ind].dat=="alpha") {
999                     if(isalpha(value[0]))
1000                        vec[n+ind].val=value;
1001                     else vec[n+ind].val=toalpha(value);
1002                 }
1003                 else if(vec[n+ind].dat=="num"){
1004                     if(isalpha(value[0]))
1005                         vec[n+ind].val=tonumber(value);
1006                     else {
1007                       if(value[0]=='.'){ vec[n+ind].val+='0';
1008                       vec[n+ind].val+=value;
1009                       }
```

```
1010                    else vec[n+ind].val=value;
1011                }
1012            }
1013            }
1014            else{
1015                for(long int i=0;i<value.length();i++){
1016                    vec[n].val=value[i];
1017                    n++;
1018                }
1019            }
1020        }
1021        else chh=0;
1022        ind=0;
1023        str=0;
1024    }
1025 }
1026
1027 void out(){ ///PROCESSES AN OUTPUT STATEMENT
1028    int op=0,co=0,sp=0,clit=0,str=0;
1029    string tmp,st;
1030    long int ind=0,a=-1;
1031    getline(fin,tmp,' ');
1032    while(tmp!="EOLN\n"){
1033        if(clit==0)
1034        getline(fin,tmp,' ');
1035        clit=0;
1036        if(tmp=="STRING:"){
1037        tmp="";
1038        while((tmp!="CLOSEB")&&(tmp!="OPENB")&&tmp!="STRING:"&&tmp!="en"&&
                tmp!="NUMB:"&&tmp!="LIT:"&&
1039              tmp!="OP:"&&tmp!="VARIABLE:"&&getline(fin,tmp,' ')){
1040        if(tmp=="")
              {if(getline(fin,tmp,' ')&&(tmp!="CLOSEB")&&
                 (tmp!="OPENB")&&tmp!="en"&&tmp!="STRING:"&&
1041               tmp!="NUMB:"&&tmp!="LIT:"&&tmp!="VARIABLE:"&&tmp!="OP:")
                     {cout<<" ";}}
1042        if((tmp!="CLOSEB")&&(tmp!="OPENB")&&tmp!="STRING:"&&tmp!="NUMB:"&&
              tmp!="LIT:"&&tmp!="en"&&
1043          tmp!="OP:"&&tmp!="VARIABLE:") {if(sp>0) cout<<" ";
1044        cout<<tmp;}
1045        sp++;
1046        if(tmp=="STRING:") clit++;
1047        }
1048        sp=0;
1049        }
1050        if(tmp=="OPENB"){
1051          st+='('; op++;
1052        }
1053        else if(tmp=="en") cout<<'\n';
1054        else if(tmp=="CLOSEB"){
1055           co++;
1056           if(op>=co){
1057             st+=')';
1058           }
1059           if(op==co){
1060             op=0;co=0;
1061           }
```

```
1062            else if(op==0&&co>0){
1063              if(st.length()!=0){
1064              int check=find_no_op(st);
1065              if(check!=1||a==-1){
1066              st=infix(st);
1067              cout<<st;
1068              st="";
1069              }
1070              else{
1071                  st=infix(st);
1072                  if(vec[a+ind].dat=="alpha") cout<<toalpha(st);
1073                  else cout<<st;
1074                  st="";
1075              }
1076              a=-1;
1077            }
1078            }
1079          }
1080          else if(tmp==""){
1081             if(st.length()!=0){
1082               int check=find_no_op(st);
1083               if(check!=1||a==-1){
1084               st=infix(st);
1085               cout<<st;
1086               st="";
1087               }
1088               else{
1089                   st=infix(st);
1090                   if(vec[a+ind].dat=="alpha") cout<<toalpha(st);
1091                   else cout<<st;
1092                   st="";
1093               }
1094               a=-1;
1095             }
1096          }
1097          else if(tmp=="RELATION:"){
1098              getline(fin,tmp,' ');
1099              st+=tmp;
1100          }
1101          else if(tmp=="VARIABLE:"){
1102              getline(fin,tmp,' ');
1103              a=search(tmp);
1104              ind=0;
1105              string tmp2;
1106              getline(fin,tmp2,' ');
1107              clit++;
1108              if(tmp2=="INDEX:"){
1109                   getline(fin,tmp2,' ');
1110                   if(tmp2=="VARIABLE:")
1111                       getline(fin,tmp2,' ');
1112                   else if(tmp2=="OPENB") {tmp2=arrexp();}
1113                   else if(tmp2==""||tmp2=="CLOSEB"){
1114                       while(vec[a].val!=""){
1115                            cout<<vec[a].val;
1116                            a++;
1117                       }
1118                       str++;
```

```
1119                        a=-1;
1120                        getline(fin,tmp2,' ');
1121                    }
1122                    if(str==0){
1123                      if(isdigit(tmp2[0]))
1124                         ind=tonum(tmp2,0,tmp2.length()-1);
1125                      else{
1126                         long int n1=search(tmp2);
1127                         ind=tonum(vec[n1].val,0,vec[n1].val.length()-1);
1128                      }
1129                      getline(fin,tmp2,' ');
1130                      tmp+='#';
1131                      stringstream ss;
1132                      ss<<ind;
1133                      tmp+=ss.str();
1134                    }
1135              }
1136              if(str==0){
1137                st+=tmp;
1138                st+=' ';
1139              }
1140              tmp=tmp2;
1141              str=0;
1142
1143          }
1144          else if(tmp=="OP:"){
1145              getline(fin,tmp,' ');
1146              st+=tmp;
1147          }
1148          else if(tmp=="NEGATION")
1149              st+='!';
1150          else if(tmp=="AND")
1151              st+='&';
1152          else if(tmp=="OR")
1153              st+='^';
1154          else if(tmp=="NUMB:"){
1155              getline(fin,tmp,' ');
1156              st+=tmp;
1157              st+=' ';
1158          }
1159          else if(tmp=="LIT:"){
1160              getline(fin,tmp,' ');
1161              string tmp2;
1162              getline(fin,tmp2,' ');
1163              clit++;
1164              if(tmp2==""||tmp2=="CLOSEB"){
1165                 if(tmp!="")
1166                   cout<<tmp;
1167                 else
1168                   cout<<' ';
1169              }
1170              else{
1171              if(tmp=="") st+=tonumber(" ");
1172              else
1173              st+=tonumber(tmp);
1174              st+=' ';
1175              }
```

```
1176            tmp=tmp2;
1177        }
1178     }
1179    tmp="";
1180 }
1181
1182 void out(stringstream& sin,streampos posi){ ///OVERLOADED FOR NESTED STATEMENT
                                                    BLOCK
1183    sin.seekg(posi,ios::beg);
1184    int op=0,co=0,sp=0,clit=0,str=0;
1185    string tmp,st;
1186    long int ind=0,a=-1;
1187    getline(sin,tmp,' ');
1188    while(tmp!="EOLN\n"){
1189        if(clit==0)
1190         getline(sin,tmp,' ');
1191         clit=0;
1192        if(tmp=="STRING:"){
1193         tmp="";
1194         while((tmp!="CLOSEB")&&(tmp!="OPENB")&&tmp!="STRING:"&&
                  tmp!="en"&&tmp!="NUMB:"&&
1195                tmp!="LIT:"&&tmp!="OP:"&&tmp!="VARIABLE:"&&getline(sin,tmp,' ')){
1196         if(tmp=="") {if(getline(sin,tmp,' ')&&(tmp!="CLOSEB")&&(tmp!="OPENB")&&
                          tmp!="en"&&tmp!="STRING:"&&
1197                       tmp!="NUMB:"&&tmp!="LIT:"&&tmp!="VARIABLE:"&&tmp!="OP:")
                              {cout<<" ";}}
1198         if((tmp!="CLOSEB")&&(tmp!="OPENB")&&tmp!="STRING:"&&tmp!="NUMB:"&&
               tmp!="LIT:"&&
1199            tmp!="en"&&tmp!="OP:"&&tmp!="VARIABLE:")
1200             {if(sp>0) cout<<" ";cout<<tmp;}
1201         sp++;
1202         if(tmp=="STRING:") clit++;
1203         }
1204        }
1205        if(tmp=="OPENB"){
1206          st+='('; op++;
1207        }
1208        else if(tmp=="en") cout<<'\n';
1209        else if(tmp=="CLOSEB"){
1210           co++;
1211           if(op>=co){
1212             st+=')';
1213           }
1214           if(op==co){
1215             op=0;co=0;
1216           }
1217           else if(op==0&&co>0){
1218             if(st.length()!=0){
1219             int check=find_no_op(st);
1220             if(check!=1||a==-1){
1221             st=infix(st);
1222             cout<<st;
1223             st="";
1224             }
1225             else{
1226                 st=infix(st);
1227                 if(vec[a+ind].dat=="alpha") cout<<toalpha(st);
```

```
1228                else cout<<st;
1229                st="";
1230              }
1231              a=-1;
1232            }
1233            }
1234         }
1235         else if(tmp==""){
1236            if(st.length()!=0){
1237              int check=find_no_op(st);
1238              if(check!=1||a==-1){
1239              st=infix(st);
1240              cout<<st;
1241              st="";
1242              }
1243              else{
1244                   st=infix(st);
1245                   if(vec[a+ind].dat=="alpha") cout<<toalpha(st);
1246                   else cout<<st;
1247                   st="";
1248              }
1249              a=-1;
1250            }
1251         }
1252        else if(tmp=="RELATION:"){
1253             getline(sin,tmp,' ');
1254             st+=tmp;
1255         }
1256         else if(tmp=="VARIABLE:"){
1257             getline(sin,tmp,' ');
1258             a=search(tmp);
1259             string tmp2;
1260             ind=0;
1261             getline(sin,tmp2,' ');
1262             clit++;
1263             if(tmp2=="INDEX:"){
1264                  getline(sin,tmp2,' ');
1265                  if(tmp2=="VARIABLE:")
1266                      getline(sin,tmp2,' ');
1267                  else if(tmp2=="OPENB") tmp2=arrexp(sin);
1268                  else if(tmp2==""||tmp2=="CLOSEB"){
1269                      while(vec[a].val!=""){
1270                          cout<<vec[a].val;
1271                          a++;
1272                      }
1273                      str++;
1274                      a=-1;
1275                      getline(sin,tmp2,' ');
1276                  }
1277                  if(str==0){
1278                  if(isdigit(tmp2[0]))
1279                  ind=tonum(tmp2,0,tmp2.length()-1);
1280                  else{
1281                      long int n1=search(tmp2);
1282                      ind=tonum(vec[n1].val,0,vec[n1].val.length()-1);
1283                  }
1284                  getline(sin,tmp2,' ');
```

```
1285                    tmp+='#';
1286                    stringstream ss;
1287                    ss<<ind;
1288                    tmp+=ss.str();
1289                    }
1290                }
1291                if(str==0){
1292                  st+=tmp;
1293                  st+=' ';
1294                }
1295                tmp=tmp2;
1296                str=0;
1297            }
1298            else if(tmp=="OP:"){
1299                getline(sin,tmp,' ');
1300                st+=tmp;
1301            }
1302            else if(tmp=="NEGATION")
1303                st+='!';
1304            else if(tmp=="AND")
1305                st+='&';
1306            else if(tmp=="OR")
1307                st+='^';
1308            else if(tmp=="NUMB:"){
1309                getline(sin,tmp,' ');
1310                st+=tmp;
1311            }
1312            else if(tmp=="LIT:"){
1313                getline(sin,tmp,' ');
1314                string tmp2;
1315                getline(sin,tmp2,' ');
1316                clit++;
1317                if(tmp2==""||tmp2=="CLOSEB"){
1318                   if(tmp!="")
1319                     cout<<tmp;
1320                   else
1321                     cout<<' ';
1322                }
1323                else{
1324                if(tmp=="") st+=tonumber(" ");
1325                else
1326                st+=tonumber(tmp);
1327                }
1328                tmp=tmp2;
1329            }
1330          }
1331        tmp="";
1332    }
1333
1334    void variable(string str){ ///PROCESSES A VARIABLE INITIALISATION STATEMENT
1335        string tmp,varn,v;
1336        long int n,a,ind;
1337        while(tmp!="EOLN\n"&&getline(fin,tmp,' ')){
1338            if(tmp=="VARIABLE:"){
1339                tmp="";
1340                getline(fin,tmp,' ');
1341                varn=tmp;
```

```
1342            var t(varn,str,"");
1343            vec.push_back(t);
1344            n=search(varn);
1345            v="";
1346        }
1347        else if(tmp=="INDEX:"){
1348            getline(fin,tmp,' ');
1349            if(tmp=="VARIABLE:")
1350                getline(fin,tmp,' ');
1351            else if(tmp=="OPENB") tmp=arrexp();
1352            if(isdigit(tmp[0]))
1353            ind=tonum(tmp,0,tmp.length()-1);
1354            else {
1355                long int n1=search(tmp);
1356                ind=tonum(vec[n1].val,0,vec[n1].val.length()-1);
1357            }
1358            var t(varn,str,"");
1359            for (long int i=1;i<ind;i++) vec.push_back(t);
1360        }
1361        else if(tmp=="OPENB"){
1362            long int cnt=0,strr=0;
1363            int ob=1,cb=0;
1364            while(ob!=cb){
1365              while(getline(fin,tmp,' ')&&tmp!=""&&ob!=cb){
1366               if(tmp=="VARIABLE:"){
1367                 getline(fin,tmp,' ');
1368                 v+=tmp;
1369               }
1370               else if(tmp=="STRING:"){
1371                 strr++;
1372                 long int sp=0;
1373                 string st="";
1374                 tmp="";
1375                 while((tmp!="CLOSEB")&&getline(fin,tmp,' ')){
1376                 if(tmp=="") {if(getline(fin,tmp,' ')&&(tmp!="CLOSEB")) {st+='
';}}
1377                 if((tmp!="CLOSEB")) {if(sp>0) st+=' ';
1378                 st+=tmp;}
1379                 sp++;
1380                 }
1381                 cb++;
1382                 for(long int i=0;i<st.length();i++){
1383                    vec[n+cnt].val=st[i];
1384                    cnt++;
1385                 }
1386               }
1387               else if(tmp=="INDEX:"){
1388                 getline(fin,tmp,' ');
1389                 v+='#';
1390                 if(tmp=="VARIABLE:")
1391                    getline(fin,tmp,' ');
1392                 else if(tmp=="OPENB") tmp=arrexp();
1393                 v+=tmp;
1394               }
1395               else if(tmp=="RELATION:"){
1396                 getline(fin,tmp,' ');
1397                 v+=tmp;
```

```
1398                }
1399                else if(tmp=="OP:"){
1400                  getline(fin,tmp,' ');
1401                  v+=tmp;
1402                }
1403                else if(tmp=="NEGATION")
1404                    v+='!';
1405                else if(tmp=="AND")
1406                    v+='&';
1407                else if(tmp=="OR")
1408                    v+='^';
1409                else if(tmp=="CLOSEB") {v+=')';cb++;}
1410                else if(tmp=="OPENB") {v+='(';ob++;}
1411                else{
1412                 if(str=="alpha"){
1413                    if(tmp=="LIT:"){
1414                       getline(fin,tmp,' ');
1415                       if(tmp=="") {v+=tonumber(" ");}
1416                       else
1417                       v+=tonumber(tmp);
1418                       v+=' ';
1419                    }
1420                    else if(tmp=="NUMB:"){
1421                      getline(fin,tmp,' ');
1422                      v+=tmp;
1423                      v+=' ';
1424                    }
1425                 }
1426                 else if(str=="num"){
1427                    if(tmp=="NUMB:"){
1428                       getline(fin,tmp,' ');
1429                       v+=tmp;
1430                       v+=' ';
1431                    }
1432                    else if(tmp=="LIT:"){
1433                       getline(fin,tmp,' ');
1434                       if(tmp=="") v+=tonumber(" ");
1435                       else
1436                       v+=tonumber(tmp);
1437                       v+=' ';
1438                    }
1439                 }
1440             }
1441             }
1442          if(strr==0){
1443          if(ob==cb) v.pop_back();
1444          if(str=="alpha")
1445          vec[n+cnt].val=toalpha(infix(v));
1446          else {
1447              vec[n+cnt].val=infix(v);
1448              if(vec[n+cnt].val=="") vec[n+cnt].val="0";
1449          }
1450          v="";
1451          cnt++;
1452          }
1453          }
1454          }
```

```
1455          else if(tmp=="ASSIGN"){
1456              while(getline(fin,tmp,' ')&&tmp!=""&&tmp!="EOLN\n"){
1457                 if(tmp=="VARIABLE:"){
1458                   getline(fin,tmp,' ');
1459                   v+=tmp;
1460                 }
1461                 else if(tmp=="INDEX:"){
1462                   getline(fin,tmp,' ');
1463                   v+='#';
1464                   if(tmp=="VARIABLE:")
1465                      getline(fin,tmp,' ');
1466                   else if(tmp=="OPENB") tmp=arrexp();
1467                   v+=tmp;
1468
1469                 }
1470                 else if(tmp=="RELATION:"){
1471                   getline(fin,tmp,' ');
1472                   v+=tmp;
1473
1474                 }
1475                 else if(tmp=="OP:"){
1476                   getline(fin,tmp,' ');
1477                   v+=tmp;
1478                 }
1479                 else if(tmp=="NEGATION")
1480                     v+='!';
1481                 else if(tmp=="AND")
1482                     v+='&';
1483                 else if(tmp=="OR")
1484                     v+='^';
1485                 else if(tmp=="CLOSEB") v+=')';
1486                 else if(tmp=="OPENB") v+='(';
1487                 else{
1488
1489                  if(str=="alpha"){
1490                     if(tmp=="LIT:"){
1491                        getline(fin,tmp,' ');
1492                        if(tmp=="") {v+=tonumber(" ");}
1493                        else
1494                        v+=tonumber(tmp);
1495                        v+=' ';
1496                     }
1497                     else if(tmp=="NUMB:"){
1498                       getline(fin,tmp,' ');
1499                       v+=tmp;
1500                       v+=' ';
1501                     }
1502                  }
1503                  else if(str=="num"){
1504                     if(tmp=="NUMB:"){
1505                        getline(fin,tmp,' ');
1506                        v+=tmp;
1507                        v+=' ';
1508                     }
1509                     else if(tmp=="LIT:"){
1510                        getline(fin,tmp,' ');
1511                        if(tmp=="") v+=tonumber(" ");
```

```
                        else
                        v+=tonumber(tmp);
                        v+=' ';
                    }
                }
            }
        }

        if(str=="alpha"){
        vec[n].val=toalpha(infix(v));
        }
        else {
            vec[n].val=infix(v);
            if(vec[n].val=="") vec[n].val="0";
        }
        v="";
        }

    }
    }

void variable(string str,stringstream& fin,streampos posi){
    ///OVERLOADED FOR NESTED STATEMENT BLOCK
    fin.seekg(posi,ios::beg);
    string tmp,varn,v;
    long int n,a,ind;
    while(tmp!="EOLN\n"&&getline(fin,tmp,' ')){
        if(tmp=="VARIABLE:"){
            tmp="";
            getline(fin,tmp,' ');
            varn=tmp;
            var t(varn,str,"");
            vec.push_back(t);
            n=search(varn);
            v="";
        }
        else if(tmp=="INDEX:"){
            getline(fin,tmp,' ');
            if(tmp=="VARIABLE:")
                getline(fin,tmp,' ');
            else if(tmp=="OPENB") tmp=arrexp(fin);
            if(isdigit(tmp[0]))
            ind=tonum(tmp,0,tmp.length()-1);
            else {
                long int n1=search(tmp);
                ind=tonum(vec[n1].val,0,vec[n1].val.length()-1);
            }
            var t(varn,str,"");
            for (long int i=1;i<ind;i++) vec.push_back(t);
        }
        else if(tmp=="OPENB"){
            long int cnt=0,strr=0;
            int ob=1,cb=0;
            while(ob!=cb){
              while(getline(fin,tmp,' ')&&tmp!=""&&ob!=cb){
               if(tmp=="VARIABLE:"){
                 getline(fin,tmp,' ');
```

```
1569                    v+=tmp;
1570                  }
1571                  else if(tmp=="STRING:"){
1572                    strr++;
1573                    long int sp=0;
1574                    string st="";
1575                    tmp="";
1576                    while((tmp!="CLOSEB")&&getline(fin,tmp,' ')){
1577                    if(tmp=="") {if(getline(fin,tmp,' ')&&(tmp!="CLOSEB")) {st+='
';}}
1578                    if((tmp!="CLOSEB")) {if(sp>0) st+=' ';
1579                    st+=tmp;}
1580                    sp++;
1581                    }
1582                    cb++;
1583                    for(long int i=0;i<st.length();i++){
1584                       vec[n+cnt].val=st[i];
1585                       cnt++;
1586                    }
1587                  }
1588                  else if(tmp=="RELATION:"){
1589                    getline(fin,tmp,' ');
1590                    v+=tmp;
1591                  }
1592                  else if(tmp=="OP:"){
1593                    getline(fin,tmp,' ');
1594                    v+=tmp;
1595                  }
1596                  else if(tmp=="NEGATION")
1597                      v+='!';
1598                  else if(tmp=="AND")
1599                      v+='&';
1600                  else if(tmp=="OR")
1601                      v+='^';
1602                  else if(tmp=="CLOSEB") {v+=')';cb++;}
1603                  else if(tmp=="OPENB") {v+='(';ob++;}
1604                  else{
1605                   if(str=="alpha"){
1606                      if(tmp=="LIT:"){
1607                         getline(fin,tmp,' ');
1608                         if(tmp=="") v+=tonumber(" ");
1609                         else
1610                         v+=tonumber(tmp);
1611                         v+=' ';
1612                      }
1613                      else if(tmp=="NUMB:"){
1614                        getline(fin,tmp,' ');
1615                        v+=tmp;
1616                        v+=' ';
1617                      }
1618                   }
1619                   else if(str=="num"){
1620                      if(tmp=="NUMB:"){
1621                         getline(fin,tmp,' ');
1622                         v+=tmp;
1623                         v+=' ';
1624                      }
```

```
1625                    else if(tmp=="LIT:"){
1626                       getline(fin,tmp,' ');
1627                       if(tmp=="") v+=tonumber(" ");
1628                       else
1629                       v+=tonumber(tmp);
1630                       v+=' ';
1631                    }
1632                 }
1633              }
1634            }
1635          if(strr==0){
1636          if(ob==cb) v.pop_back();
1637          if(str=="alpha")
1638          vec[n+cnt].val=toalpha(infix(v));
1639          else {
1640              vec[n+cnt].val=infix(v);
1641              if(vec[n+cnt].val=="") vec[n].val="0";
1642          }
1643          v="";
1644          cnt++;
1645          }
1646          }
1647          }
1648          else if(tmp=="ASSIGN"){
1649              while(getline(fin,tmp,' ')&&tmp!=""&&tmp!="EOLN\n"){
1650
1651                if(tmp=="VARIABLE:"){
1652                   getline(fin,tmp,' ');
1653                   v+=tmp;
1654                 }
1655                 else if(tmp=="INDEX:"){
1656                   getline(fin,tmp,' ');
1657                   v+='#';
1658                   if(tmp=="VARIABLE:")
1659                      getline(fin,tmp,' ');
1660                   else if(tmp=="OPENB") tmp=arrexp(fin);
1661                   v+=tmp;
1662                 }
1663                 else if(tmp=="RELATION:"){
1664                   getline(fin,tmp,' ');
1665                   v+=tmp;
1666                 }
1667                 else if(tmp=="OP:"){
1668                   getline(fin,tmp,' ');
1669                   v+=tmp;
1670                 }
1671                 else if(tmp=="NEGATION")
1672                     v+='!';
1673                 else if(tmp=="AND")
1674                     v+='&';
1675                 else if(tmp=="OR")
1676                     v+='^';
1677                 else if(tmp=="CLOSEB") v+=')';
1678                 else if(tmp=="OPENB") v+='(';
1679                 else{
1680                 if(str=="alpha"){
1681                     if(tmp=="LIT:"){
```

```
1682                        getline(fin,tmp,' ');
1683                        if(tmp=="") v+=tonumber(" ");
1684                        else
1685                        v+=tonumber(tmp);
1686                        v+=' ';
1687                    }
1688                    else if(tmp=="NUMB:"){
1689                      getline(fin,tmp,' ');
1690                      v+=tmp;
1691                      v+=' ';
1692                    }
1693                }
1694                else if(str=="num"){
1695                    if(tmp=="NUMB:"){
1696                        getline(fin,tmp,' ');
1697                        v+=tmp;
1698                        v+=' ';
1699                    }
1700                    else if(tmp=="LIT:"){
1701                        getline(fin,tmp,' ');
1702                        if(tmp=="") v+=tonumber(" ");
1703                        else
1704                        v+=tonumber(tmp);
1705                        v+=' ';
1706                    }
1707                }
1708            }
1709        }
1710        if(str=="alpha")
1711        vec[n].val=toalpha(infix(v));
1712        else {
1713            vec[n].val=infix(v);
1714            if(vec[n].val=="") vec[n].val="0";
1715        }
1716        }
1717    }
1718 }
1719
1720 int main(int argc,char** argv){    ///DECIDED WHAT FUNCTION TO CALL AND WHEN
1721     ///ARGUMENTS TO MAIN ARE USED AS THIS FUNCTION IS BEING CALLED BY THE IDE
1722     fin.open("lexer.txt",ios::out);
1723     fin<<puttotxt(argv[1]);
1724     fin.close();
1725     fin.open("lexer.txt",ios::in);
1726     var openb("","","");
1727     vec.push_back(openb);
1728     string str;
1729     int gap=0,condi=0;
1730     while(getline(fin,str,' ')){
1731         if(str=="OPENCB\n"){
1732             gap++;
1733             vec.push_back(openb);
1734         }
1735         else if(str=="CLOSECB\n"){
1736           gap++;
1737             for(long int it=vec.size()-1;vec[it].name!="";it--){
1738                 vec.pop_back();
```

```
1739              }
1740              vec.pop_back();
1741            }
1742          else if(str=="EOLN\n")gap++;
1743          else if(str=="en") {gap++;cout<<'\n';}
1744          else if(str=="RESERVED:"){
1745            str="";
1746            getline(fin,str,' ');
1747            if(str=="out") {gap++;out();}
1748            else if(str=="in") {gap++;in();}
1749            else if(str=="alpha"||str=="num") {gap++;variable(str);}
1750            else if(str=="iter") {gap++;iter();}
1751            else if(str=="check"||str=="or"||str=="orelse"){
1752              if(str=="check") gap++;
1753              if(gap!=0){condi=0;gap=0;}
1754              if(str=="orelse"){
1755                  int ob=1,cb=0;
1756                  getline(fin,str,' ');
1757                  string st="";
1758                  while(ob!=cb&&getline(fin,str,' ')){
1759                      if(str=="OPENCB\n") ob++;
1760                      else if(str=="CLOSECB\n") cb++;
1761                      st+=str;
1762                      st+=' ';
1763                  }
1764                  if(condi==0&&gap==0)
1765                  main_fn(st);
1766              }
1767              else if(str=="or"){
1768                  condi+=check(condi);
1769              }
1770              else{
1771                  condi+=check(condi);
1772              }
1773            }
1774          }
1775          else if(str=="VARIABLE:"){
1776              gap++;
1777              string v;
1778              long int ind=0;
1779              getline(fin,str,' ');
1780              long int n=search(str),a;
1781              getline(fin,str,' ');
1782              if(str=="INDEX:"){
1783                  getline(fin,str,' ');
1784                  if(str=="VARIABLE:") {
1785                      getline(fin,str,' ');
1786                      long int n1=search(str);
1787                      ind=tonum(vec[n1].val,0,vec[n1].val.length()-1);
1788                  }
1789                  else if(str=="OPENB")
                          {str=arrexp();ind=tonum(str,0,str.length()-1);}
1790                  else
1791                  ind=tonum(str,0,str.length()-1);
1792                  getline(fin,str,' ');
1793                  }
1794              if(str=="ASSIGN"){
```

```
1795              while(getline(fin,str,' ')&&str!="EOLN\n"){
1796                 if(str=="VARIABLE:"){
1797                   getline(fin,str,' ');
1798                   v+=str;
1799                 }
1800                 else if(str=="RELATION:"){
1801                   getline(fin,str,' ');
1802                   v+=str;
1803                 }
1804                 else if(str=="INDEX:"){
1805                   getline(fin,str,' ');
1806                   v+='#';
1807                   if(str=="VARIABLE:")
1808                      getline(fin,str,' ');
1809                   else if(str=="OPENB") str=arrexp();
1810                   v+=str;
1811                 }
1812                 else if(str=="OP:"){
1813                   getline(fin,str,' ');
1814                   v+=str;
1815                 }
1816                 else if(str=="NEGATION")
1817                     str+='!';
1818                 else if(str=="AND")
1819                     str+='&';
1820                 else if(str=="OR")
1821                     str+='^';
1822                 else if(str=="CLOSEB") v+=')';
1823                 else if(str=="OPENB") v+='(';
1824                 else{
1825                  if(vec[n].dat=="alpha"){
1826                     if(str=="LIT:"){
1827                        getline(fin,str,' ');
1828                        if(str=="") v+=tonumber(" ");
1829                        else
1830                        v+=tonumber(str);
1831                        v+=' ';
1832                     }
1833                     else if(str=="NUMB:"){
1834                       getline(fin,str,' ');
1835                       v+=(str);
1836                       v+=' ';
1837                     }
1838                  }
1839                  else if(vec[n].dat=="num"){
1840                     if(str=="NUMB:"){
1841                        getline(fin,str,' ');
1842                        v+=str;
1843                        v+=' ';
1844                     }
1845                     else if(str=="LIT:"){
1846                        getline(fin,str,' ');
1847                        if(str=="") v+=tonumber(" ");
1848                        else
1849                        v+=tonumber(str);
1850                        v+=' ';
1851                     }
```

```
1852                    }
1853                }
1854            }
1855        }
1856        if(vec[n+ind].dat=="alpha")
1857          vec[n+ind].val=toalpha(infix(v));
1858        else {
1859              vec[n+ind].val=infix(v);
1860              if(vec[n+ind].val=="") vec[n].val="0";
1861            }
1862        }
1863      }
1864      fin.close();
1865      DeleteFile("lexer.txt");
1866      cout<<"\n\n";
1867      system("pause");
1868  }
```

SCREEN OUTPUTS:

SAMPLE PROGRAM 1:

abs02.TXT - Notepad

File Edit Format View Help

```
cmt(TO SHOW HOW THE OUT STATEMENT WORKS!!);
out("HELLO WORLD!!");
en;
out("HELLO",' ',"WORLD!!",' ',1,2,3,'!');
en;
en;

cmt(TO SHOW HOW THE SINGLE VARIABLE DECLARATION AND ASSIGNING WORKS);
num a=1 + 2  ;
num b=3 2 + a -,c;
out("ENTER VALUE OF c: ");
in(c);
out(3 2+,' ',a+b*2-c);
en;
en;

cmt(TO SHOW HOW ARRAYS AND DECLARATION WORK);
num d='a'+2,arr#(a+2)(1,2,3);
alpha al=65,str#(arr#2*10+2)("empty");
out("ENTER TWO VALUES TO FILL THE ARRAY:");
in(arr#3,arr#4);
out("ENTER YOUR NAME:");
in(str#);
out("YOUR NAME: ",str#,en);
out("VALUE OF al: ",al,en);


cmt(TO SHOW HOW BASIC LOOPS WORK);
num cnt=0;
iter(cnt<5){
    out("VALUE",cnt+1,": ",arr#cnt,en);
    cnt=cnt+1;
}

cmt(END OF PROGRAM);
```

C:\Users\NAVIN\Desktop\IDE - new\executable\execute.exe

```
HELLO WORLD!!
HELLO WORLD!! 123!

ENTER VALUE OF c:3
5 178

ENTER TWO VALUES TO FILL THE ARRAY:4 5
ENTER YOUR NAME:ABISHEK
YOUR NAME:  ABISHEK
VALUE OF al:  A
VALUE1 :  1
VALUE2 :  2
VALUE3 :  3
VALUE4 :  4
VALUE5 :  5


Press any key to continue . . .
```

SAMPLE PROGRAM 2: (BUBBLE SORT)

abs01.txt - Notepad

File Edit Format View Help

```
cmt( BUBBLE SORT );
num a=0,b;
out("enter number of elements");
in(b);
num ar#b;

cmt(INPUT);

iter(a<b){
  in(ar#a);
  a=a+1;
}

cmt(SORTING);

num i=0;
iter(i<b){
  num j=0;
  iter(j<b-i-1){
   check(ar#j>ar#(j+1)){
     num n=ar#(j+1);
     ar#(j+1)=ar#j;
     ar#j=n;
   }
   j=j+1;
  }
  i=i+1;
}

cmt(OUTPUT);
i=0;
iter(i<b){
  out(ar#i,' ');
  i=i+1;
}
```

C:\Users\NAVIN\Desktop\IDE - new\executable\execute.exe

```
enter number of elements5
6 5 4 3 2
2 3 4 5 6

Press any key to continue . . .
```

SAMPLE PROGRAM 3:

abs03.txt - Notepad

File Edit Format View Help

```
cmt(TO CHECK THE CATEGORY OF THE CITIZEN);

num age;
alpha name#50;
out("enter the name of person");
in(name#);
out(en,"enter the age");
in(age);

check(age<18) { out("child"); }
or(age>18 & age<50) { out("adult"); }
orelse { out("senior citizen"); }
```

C:\Users\NAVIN\Desktop\IDE\executable\execute.exe

```
enter the name of person abishek

enter the age 16
child

Press any key to continue . . .
```

C:\Users\NAVIN\Desktop\IDE\executable\execute.exe

```
enter the name of person ahmed

enter the age 30
adult

Press any key to continue . . .
```

C:\Users\NAVIN\Desktop\IDE\executable\execute.exe

```
enter the name of person naren

enter the age 60
senior citizen

Press any key to continue . . .
```