

# Final Assignment

CSE-0302 Summer 2021

Prachurja Kanti Banrman Praggo  
Department of Computer Science and Engineering  
State University of Bangladesh (SUB)  
Dhaka, Bangladesh  
prachurjapraggo@gmail.com

**Abstract—Final Assignment**  
**Index Terms—code in c/c++**

## I. INTRODUCTION

This assignment is given by Compiler design course. The assignment is done with c and c++ code.

## II. CODES

```
Assignment=1 include<stdio.h> include<string.h>
int i,j,k,l,m,n=0,o,p,nv,z=0,t,x=0; char
str[10],temp[20],temp2[20],temp3[20];
struct prod char lhs[10],rhs[10][10]; int n; pro[10];
void findter() for(k=0;k<n;k++) if(temp[i]==pro[k].lhs[0])
for(t=0;t<n;t++) for(l=0;l<20;l++) temp2[l]="";
for(l=i+1;l<strlen(temp);l++) temp2[l-i-1]=temp[l];
for(l=i;l<20;l++) temp[l]=""; for(l=0;l<strlen(pro[k].rhs[t]);l++)
temp[i+l]=pro[k].rhs[t][l]; strcat(temp,temp2);
if(str[i]==temp[i]) return; else if(str[i]!=temp[i] temp[i]<65
temp[i]>90) break; break; if(temp[i]<65 temp[i]>90)
findter();
int main() FILE *f; // clrscr();
for(i=0;i<10;i++) pro[i].n=0;
f=fopen("in.txt","r"); while(!feof(f)) fscanf(f,"%c",&n) if(
strcmp(pro[n].lhs,pro[n-1].lhs) == 0 ) pro[n].lhs[0]="";
fscanf(f,"%c",&pro[n-1].n++; continue; fscanf(f,"%c",&n++; n++;
n--;
printf("GRAMMAR IS AS FOLLOWS"); for(i=0;i<n;i++)
for(j=0;j<pro[i].n;j++) printf("
while(1) for(l=0;l<10;l++) str[0]=NULL;
printf("ANY STRING ( 0 for EXIT ) : ");
scanf("%c",&str[0]); if(str[0]=='0') break;
for(j=0;j<pro[0].n;j++) for(l=0;l<20;l++) temp[l]=NULL;
strcpy(temp,pro[0].rhs[j]);
m=0; for(i=0;i<strlen(str);i++) if(str[i]==temp[i]) m++;
else if(str[i]!=temp[i] temp[i]<65 temp[i]>90) findter();
if(str[i]==temp[i]) m++; else if( str[i]!=temp[i] (temp[i]<65
—— temp[i]>90) ) break;
if(m==strlen(str) strlen(str)==strlen(temp))
printf("STRING can be PARSED !!!"); break;
if(j==pro[0].n) printf("STRING can NOT be PARSED
!!!");
// cin.ignore(numeric_limits< streamsize >::max(),'');
```

## III. CODES

```
include <stdio.h> include <stdlib.h>
FILE *fp , *fp2;
void check_comment(char*charx;
if( a == '/' ) //checking if the character starts with '/', it will
be a comment if((x=fgetc(fp))== '*') check_block_comment();
else if( x == '/' ) // else if the next character '/', it is the
beginning of single line comment check_single_comment();
else // when both the cases fail then it is not a comment
fputc(a,fp2); fputc(x,fp2);
// when all the conditions are false, add the character as it
is in the new file. else fputc(a,fp2);
// function for block comments void check_block_comment()
char x,y;
while((x=fgetc(fp))!=EOF) // the block comment has
started
if(x=='*') y=fgetc(fp); // check if it ends
if(y=='/') return;
// function for single line comments void
check_single_comment()char x,y;
while((x=fgetc(fp))!=EOF)
if(x=='') return; // if the comment ends return from the
function
int main(void) char c;
fp = fopen ("testfile.txt","r") ; // first file in read mode fp2
= fopen ("solved.txt","w") ; // second file in write mode
while( (c=fgetc(fp))!=EOF)
check_comment(c); //checking forthebeginningofacomment
// closing both files fclose(fp); fclose(fp2);
return 0;
```

## IV. CONCLUSION AND FUTURE WORK

None

## ACKNOWLEDGMENT

I would like to thank my honourable **Khan Md. Hasib Sir** for his time, generosity and critical insights into this project.

## REFERENCES

- [1] Wilhelm, R., Maurer, D. (1995). Compiler design. Reading: Addison-Wesley Publishing Company.
- [2] Grune, D., Van Reeuwijk, K., Bal, H. E., Jacobs, C. J., Langendoen, K. (2012). Modern compiler design. Springer Science Business Media.

- [3] Muchnick, S. (1997). Advanced compiler design implementation. Morgan kaufmann.
- [4] Hoare, C. A. R., Jifeng, H., Sampaio, A. (1993). Normal form approach to compiler design. *Acta informatica*, 30(8), 701-739.
- [5] Hoare, C. A. R., He Jifeng, and Augusto Sampaio. "Normal form approach to compiler design." *Acta informatica* 30.8 (1993): 701-739.
- [6] Bozkus, Z., Choudhary, A., Fox, G., Haupt, T., Ranka, S. (1993, November). Fortran 90D/HPF compiler for distributed memory MIMD computers: Design, implementation, and performance results. In *Supercomputing'93: Proceedings of the 1993 ACM/IEEE Conference on Supercomputing* (pp. 351-360). IEEE.
- [7] Bozkus, Zeki, et al. "Fortran 90D/HPF compiler for distributed memory MIMD computers: Design, implementation, and performance results." *Supercomputing'93: Proceedings of the 1993 ACM/IEEE Conference on Supercomputing*. IEEE, 1993.
- [8] Millstein, R. E. (1971). Compiler Design for the ILLIAC 4. MASSACHUSETTS COMPUTER ASSOCIATES INC WAKEFIELD.

Assignment - 6

1. Find the first and follow, FIRST and FOLLOW.

sets of each of the non terminals:—

Here,  $\gamma \rightarrow b$  and  $\gamma \rightarrow \epsilon$

So, we can write  $\gamma \rightarrow b | \epsilon$

Same for  $z$   $z \rightarrow c | x$  and  $z \rightarrow \epsilon$

So,  $z \rightarrow c | x | \epsilon$

So, we have  $s \rightarrow axd$ ,  $x \rightarrow \gamma z$ ,  $\gamma \rightarrow b | \epsilon$ ,  $z \rightarrow c | x$

Determining FIRST sets of each of non terminals.

$FIRST(s) = \{a\}$

$FIRST(x) = FIRST(\gamma)$   
 $= \{b, \epsilon\}$

$FIRST(\gamma) = \{b, \epsilon\}$

$FIRST(z) = \{c, \epsilon\}$

Fig. 1. Code

Determining FOLLOW sets of each

of the non-terminals

$$\text{FOLLOW}(S) = \{ \$ \}$$

$$\text{FOLLOW}(X) = \{ \$, d, \text{FOLLOW}(Z) \}$$

$$= \{ \$, d, \text{FOLLOW}(X) \}$$

$$= \{ \$, d \}$$

$$\text{FOLLOW}(Y) = \{ \text{FIRST}(Z) \}$$

$$= \{ a, c, \epsilon \}$$

$$= \{ \$, c, \text{FOLLOW}(X) \}$$

$$= \{ \$, c, d \}$$

$$\text{FOLLOW}(Z) = \{ \$, \text{FOLLOW}(X) \}$$

$$= \{ \$, d \}$$

	FIRST	FOLLOW
$S \rightarrow aXd$	$\{ a \}$	$\{ \$ \}$
$X \rightarrow YZ$	$\{ b, \epsilon \}$	$\{ \$, d \}$
$Y \rightarrow b c$	$\{ b, \epsilon \}$	$\{ \$, c, d \}$
$Z \rightarrow cX c$	$\{ c, \epsilon \}$	$\{ \$, d \}$

Fig. 2. Code

2. Construct the predictive parsing table for LL(1) method.

Non-terminals	a	b	c	\$	d
S	S → a x d				
X		X → x z	X → x z	X → x z	X → x z
Y		Y → b	Y → c	Y → c	Y → c
Z			Z → c x	Z → c	Z → c

3. The input string is,  
w = abed.

Stack	Input	Output
\$	abed	S → a x d
\$ d x a	abcd	X → x z
\$ d x	abcd	Y → b
\$ d z x	bed	Z → c x
\$ d z b	bcd	X → x z
\$ d z	cd	Y → c
\$ d x c	cd	X → x z
\$ d x	d	Y → c
\$ d z x	d	Z → c
\$ d z	d	X → x z
\$	\$	

Fig. 3. Code

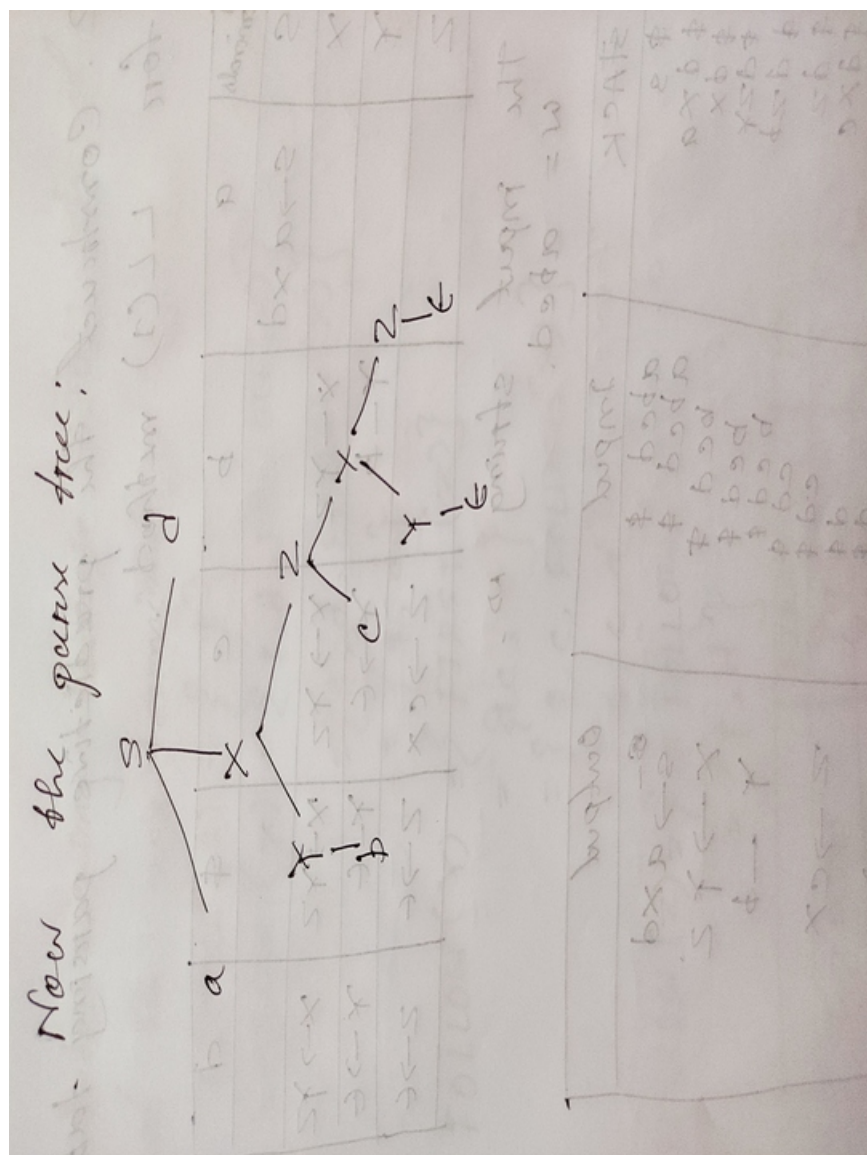
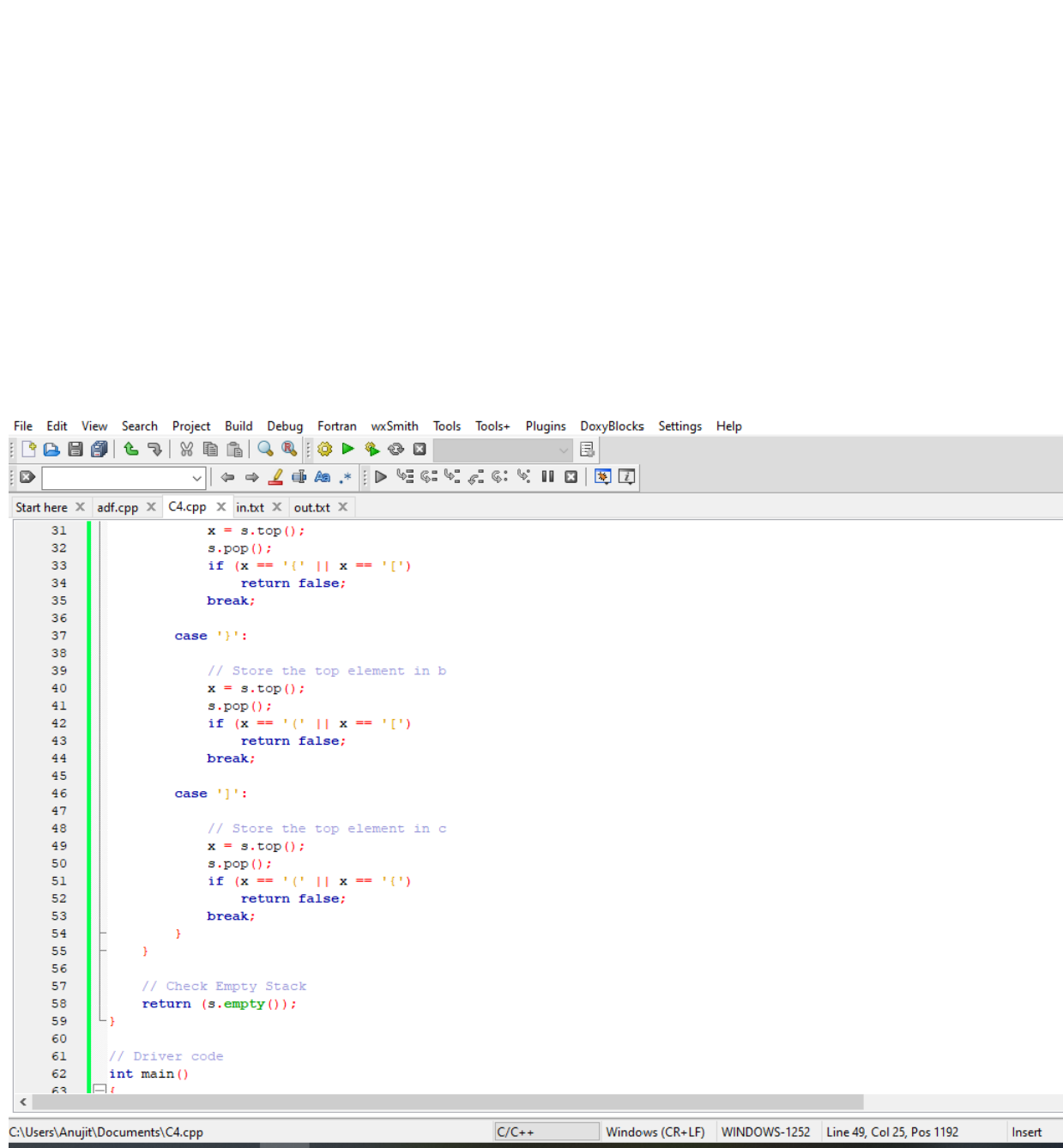


Fig. 4. Code

The image shows a screenshot of a code editor window. The title bar includes a menu (File, Edit, View, Search, Project, Build, Debug, Fortran, wxSmith, Tools, Tools+, Plugins, DoxyBlocks, Settings, Help) and a toolbar with various icons. The editor has several tabs open: 'Start here', 'adf.cpp', 'C4.cpp', 'in.txt', and 'out.txt'. The 'adf.cpp' tab is active, displaying C++ code for a function named 'areBracketsBalanced'. The code uses a stack to check if brackets in a string are balanced. It includes comments in blue and uses color-coded syntax (green for keywords, red for strings, blue for comments). The code is as follows:

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  // function to check if brackets are balanced
5  bool areBracketsBalanced(string expr)
6  {
7      stack<char> s;
8      char x;
9
10     // Traversing the Expression
11     for (int i = 0; i < expr.length(); i++)
12     {
13         if (expr[i] == '(' || expr[i] == '['
14             || expr[i] == '{')
15         {
16             // Push the element in the stack
17             s.push(expr[i]);
18             continue;
19         }
20
21         // IF current current character is not opening
22         // bracket, then it must be closing. So stack
23         // cannot be empty at this point.
24         if (s.empty())
25             return false;
26
27         switch (expr[i]) {
28             case ')':
29
30                 // Store the top element in a
31                 x = s.top();
32                 s.pop();
33                 if (x == '(' || x == '[' || x == '{')
```

Fig. 5. Code



```
31         x = s.top();
32         s.pop();
33         if (x == '(' || x == '[')
34             return false;
35         break;
36
37     case ')':
38
39         // Store the top element in b
40         x = s.top();
41         s.pop();
42         if (x == '(' || x == '[')
43             return false;
44         break;
45
46     case ']':
47
48         // Store the top element in c
49         x = s.top();
50         s.pop();
51         if (x == '(' || x == '[')
52             return false;
53         break;
54     }
55 }
56
57 // Check Empty Stack
58 return (s.empty());
59 }
60
61 // Driver code
62 int main()
63 {
```

Fig. 6. Code



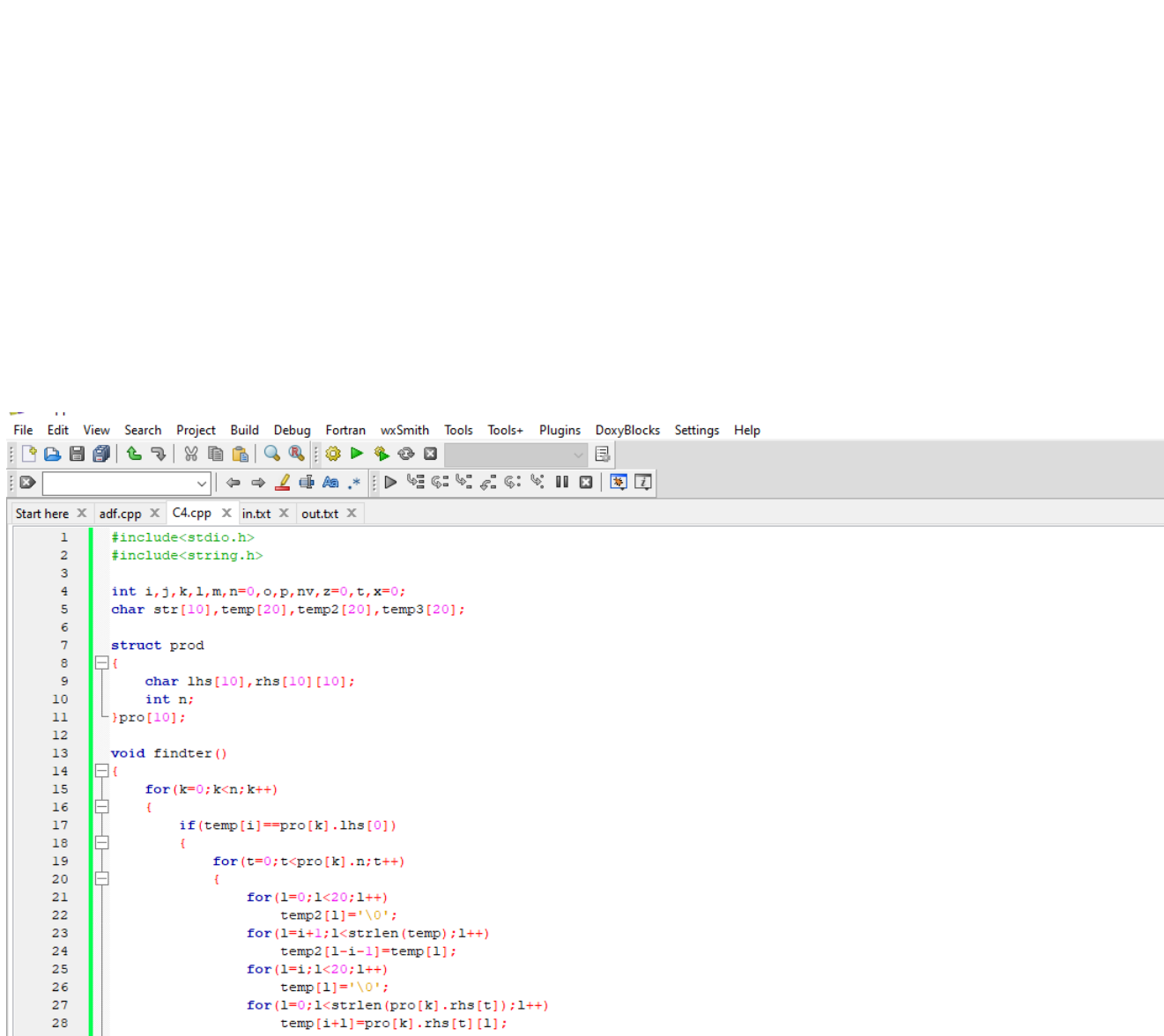
The image shows a C++ IDE with a menu bar (File, Edit, View, Search, Project, Build, Debug, Fortran, wxSmith, Tools, Tools+, Plugins, DoxyBlocks, Settings, Help) and a toolbar. The editor has tabs for 'Start here', 'adf.cpp', 'C4.cpp', 'in.txt', and 'out.txt'. The 'C4.cpp' tab is active, displaying the following code:

```
42         if (x == '(' || x == '[')
43             return false;
44         break;
45
46     case ']':
47
48         // Store the top element in c
49         x = s.top();
50         s.pop();
51         if (x == '(' || x == '[')
52             return false;
53         break;
54     }
55 }
56
57 // Check Empty Stack
58 return (s.empty());
59 }
60
61 // Driver code
62 int main()
63 {
64     string expr = "{()}[]";
65
66     // Function call
67     if (areBracketsBalanced(expr))
68         cout << "Your given input is
69     else
70         cout << "Your given input is
```

Overlaid on the bottom right is a terminal window titled 'C:\Users\Anujit\Documents\C4.exe'. It contains the following output:

```
Your given input is Balanced and also don't have any error
Process returned 0 (0x0)   execution time : 0.062 s
Press any key to continue.
```

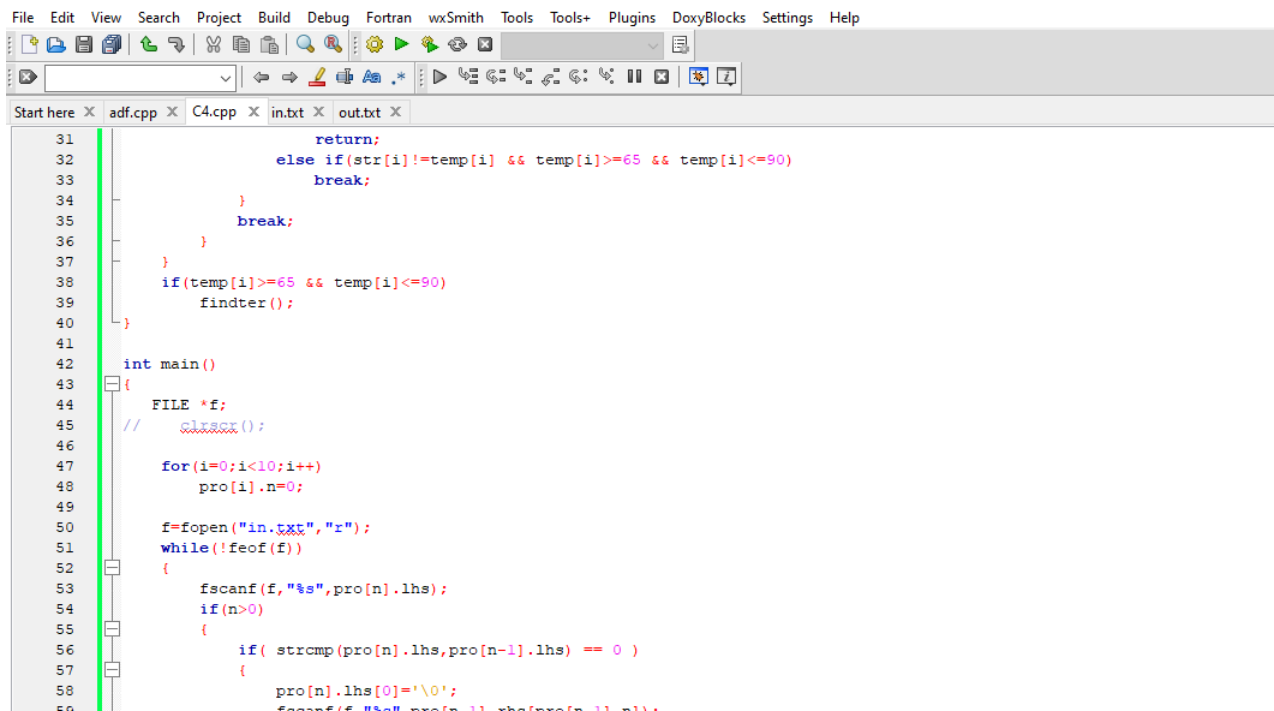
Fig. 7. Code



The screenshot displays the wxSmith IDE interface. The top menu bar includes File, Edit, View, Search, Project, Build, Debug, Fortran, wxSmith, Tools, Tools+, Plugins, DoxyBlocks, Settings, and Help. Below the menu is a toolbar with various icons for file operations, editing, and execution. The main workspace shows a project named 'Start here' with several files: adf.cpp, C4.cpp, in.txt, and out.txt. The 'adf.cpp' file is open, showing C++ code. The code includes standard headers, declares variables, defines a struct, and implements a function named 'findter'.

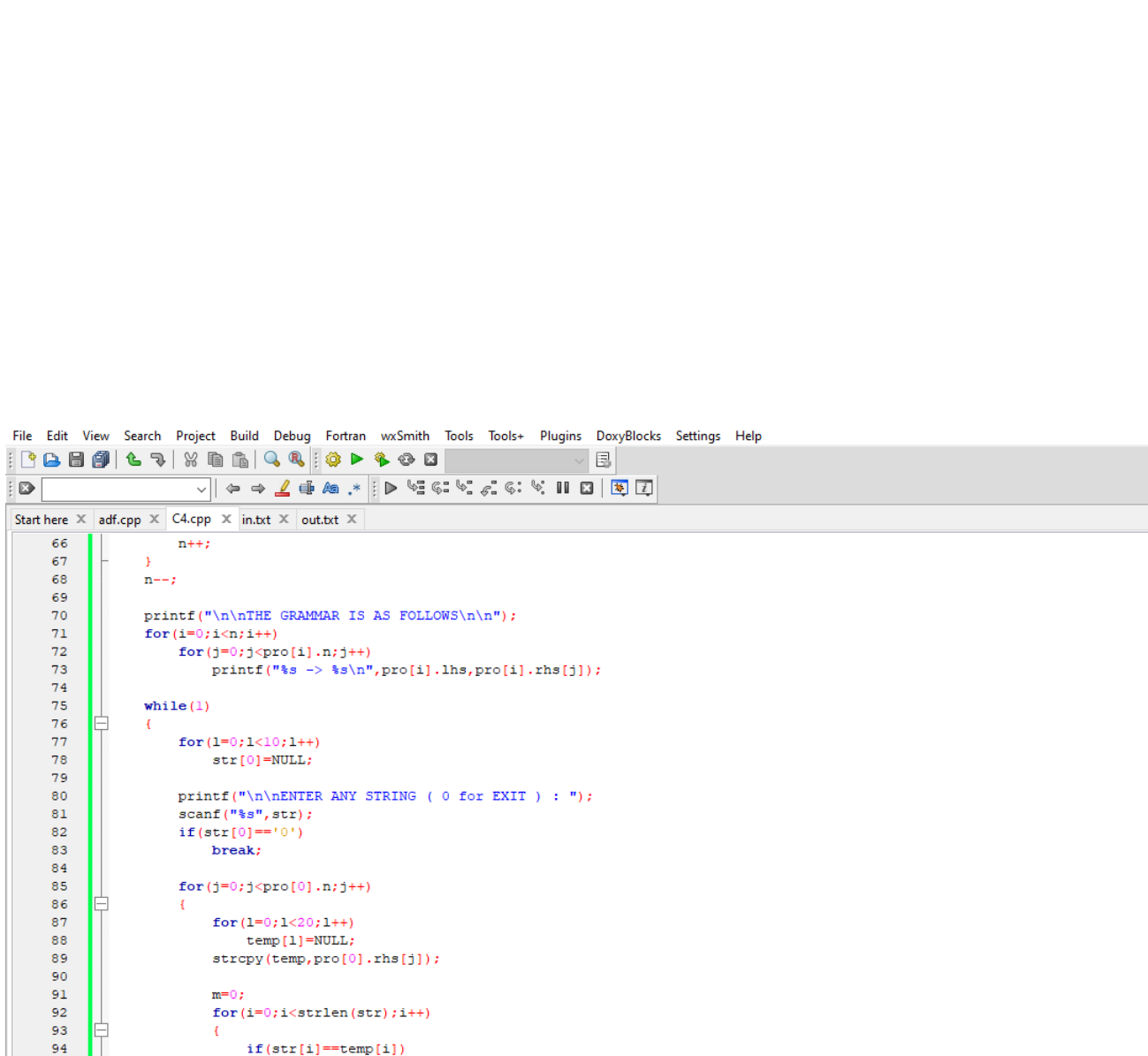
```
1  #include<stdio.h>
2  #include<string.h>
3
4  int i,j,k,l,m,n=0,o,p,nv,z=0,t,x=0;
5  char str[10],temp[20],temp2[20],temp3[20];
6
7  struct prod
8  {
9      char lhs[10],rhs[10][10];
10     int n;
11 }pro[10];
12
13 void findter()
14 {
15     for(k=0;k<n;k++)
16     {
17         if(temp[i]==pro[k].lhs[0])
18         {
19             for(t=0;t<pro[k].n;t++)
20             {
21                 for(l=0;l<20;l++)
22                     temp2[l]='\0';
23                 for(l=i+1;l<strlen(temp);l++)
24                     temp2[l-i-1]=temp[l];
25                 for(l=i;l<20;l++)
26                     temp[l]='\0';
27                 for(l=0;l<strlen(pro[k].rhs[t]);l++)
28                     temp[i+1]=pro[k].rhs[t][l];
```

Fig. 8. Code



```
31         return;
32         else if(str[i]!=temp[i] && temp[i]>=65 && temp[i]<=90)
33             break;
34     }
35     break;
36 }
37 }
38 if(temp[i]>=65 && temp[i]<=90)
39     findter();
40 }
41
42 int main()
43 {
44     FILE *f;
45     // scanf();
46
47     for(i=0;i<10;i++)
48         pro[i].n=0;
49
50     f=fopen("in.txt","r");
51     while(!feof(f))
52     {
53         fscanf(f,"%s",pro[n].lhs);
54         if(n>0)
55         {
56             if( strcmp(pro[n].lhs,pro[n-1].lhs) == 0 )
57             {
58                 pro[n].lhs[0]='\0';
59                 fscanf(f,"%s",pro[n].lhs);
60             }
61         }
62     }
63 }
```

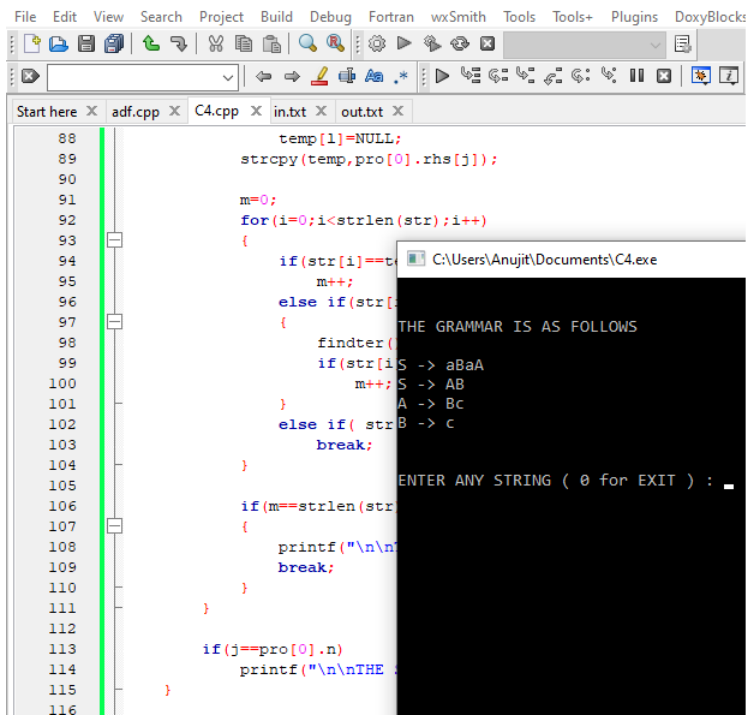
Fig. 9. Code



The image shows a screenshot of a C++ IDE with a menu bar (File, Edit, View, Search, Project, Build, Debug, Fortran, wxSmith, Tools, Tools+, Plugins, DoxyBlocks, Settings, Help) and a toolbar. The active window is 'adf.cpp', which contains the following C++ code:

```
66     n++;
67 }
68 n--;
69
70 printf("\n\nTHE GRAMMAR IS AS FOLLOWS\n\n");
71 for(i=0;i<n;i++)
72     for(j=0;j<pro[i].n;j++)
73         printf("%s -> %s\n",pro[i].lhs,pro[i].rhs[j]);
74
75 while(1)
76 {
77     for(l=0;l<10;l++)
78         str[0]=NULL;
79
80     printf("\n\nENTER ANY STRING ( 0 for EXIT ) : ");
81     scanf("%s",str);
82     if(str[0]=='0')
83         break;
84
85     for(j=0;j<pro[0].n;j++)
86     {
87         for(l=0;l<20;l++)
88             temp[l]=NULL;
89         strcpy(temp,pro[0].rhs[j]);
90
91         m=0;
92         for(i=0;i<strlen(str);i++)
93         {
94             if(str[i]==temp[i])
```

Fig. 10. Code



The image shows a screenshot of a code editor with a menu bar (File, Edit, View, Search, Project, Build, Debug, Fortran, wxSmith, Tools, Tools+, Plugins, DoxyBlocks) and a toolbar. The editor has several tabs: 'Start here', 'adf.cpp', 'C4.cpp', 'in.txt', and 'out.txt'. The 'C4.cpp' tab is active, displaying C++ code. The code includes a loop that reads input from 'in.txt' and processes it. A terminal window is overlaid on the right side of the code editor, showing the output of the program. The terminal text is as follows:

```
C:\Users\Anujit\Documents\C4.exe
THE GRAMMAR IS AS FOLLOWS
S -> aBaA
S -> AB
A -> Bc
B -> c
ENTER ANY STRING ( 0 for EXIT ) : 
```

```
88      temp[1]=NULL;
89      strcpy(temp,pro[0].rhs[j]);
90
91      m=0;
92      for(i=0;i<strlen(str);i++)
93      {
94          if(str[i]==temp[m])
95              m++;
96          else if(str[i]==temp[m])
97          {
98              findter(
99                  if(str[i]==temp[m])
100                      m++;
101                  }
102          else if(str[i]==temp[m])
103              break;
104      }
105      if(m==strlen(temp))
106      {
107          printf("\n\n");
108          break;
109      }
110  }
111
112
113  if(j==pro[0].n)
114      printf("\n\nTHE ");
115  }
116
```

Fig. 11. Code

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4  vector<string>sp,ke,ri;
5  map<string,string>mp,mpp;
6  string ans;
7
8  bool isTERMINAL(char a){
9      if(a>='A' && a<='Z') return true;
10     return false;
11 }
12
13 void FIRST(string key){
14
15     string val = mp[key];
16
17     if(isTERMINAL(val[0])){
18         string p = "";
19         p += val[0];
20         FIRST(p);
21     }
22     else{
23         ans += val[0];
24         ans += ",";
25         int flag = 0;
26         for(int i=0;i<val.size();i++){
27             if(val[i]=='|'){
28                 flag = 1;
29                 continue;
```

Fig. 12. Code

The image shows a screenshot of a C++ IDE with a menu bar (File, Edit, View, Search, Project, Build, Debug, Fortran, wxSmith, Tools, Tools+, Plugins, DoxyBlocks, Settings, Help) and a toolbar. The active window is 'adf.cpp'. The code is as follows:

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4  vector<string>sp,ke,ri;
5  map<string,string>mp,mpp;
6  string ans;
7
8  bool isTERMINAL(char a){
9      if(a>='A' && a<='Z') return true;
10     return false;
11 }
12
13 void FIRST(string key){
14
15     string val = mp[key];
16
17     if(isTERMINAL(val[0])){
18         string p = "";
19         p += val[0];
20         FIRST(p);
21     }
22     else{
23         ans += val[0];
24         ans += ",";
25         int flag = 0;
26         for(int i=0;i<val.size();i++){
27             if(val[i]=='|'){
28                 flag = 1;
29                 break;
30             }
31         }
32     }
33 }
```

Fig. 13. Code

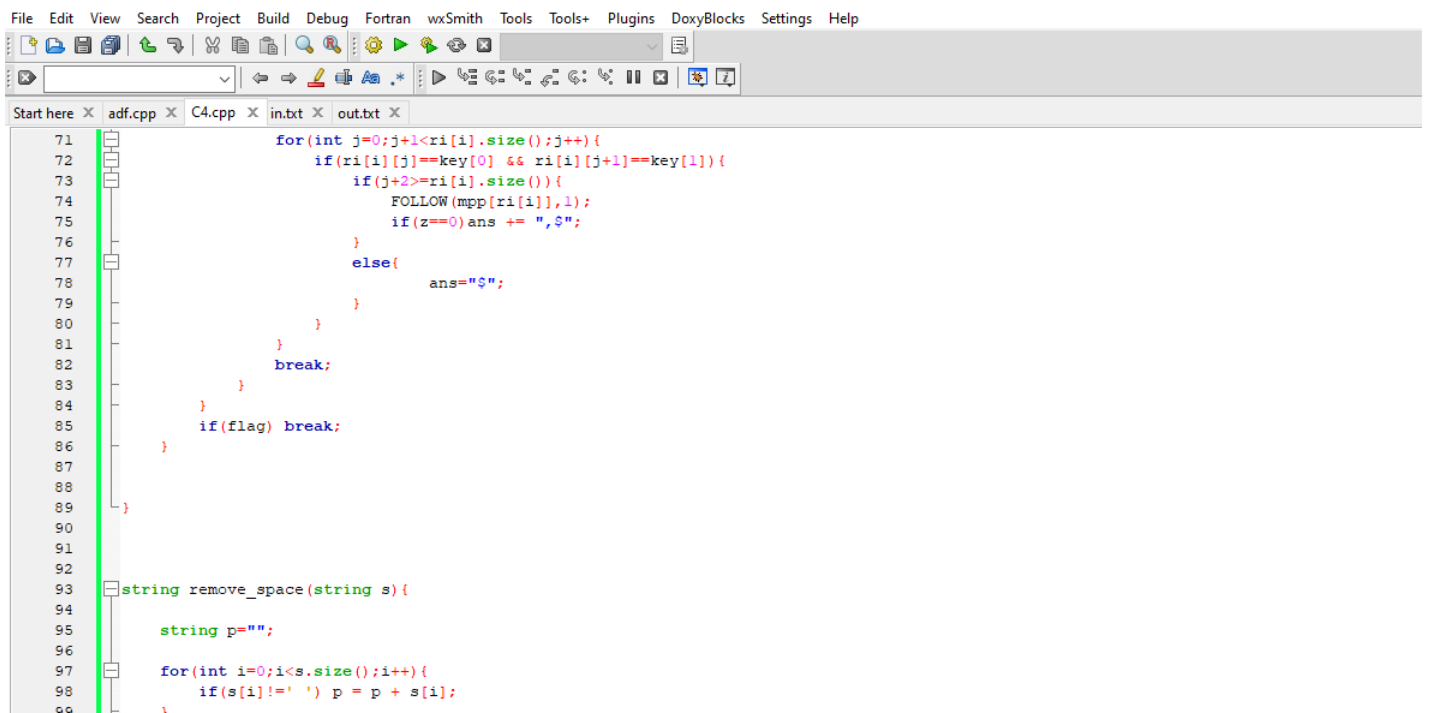
The image shows a screenshot of a C++ IDE with the following components:

- Menu Bar:** File, Edit, View, Search, Project, Build, Debug, Fortran, wxSmith, Tools, Tools+, Plugins, DoxyBlocks, Settings, Help.
- Toolbar:** Contains icons for file operations (open, save, print), editing (undo, redo, copy, paste), and development (run, debug, breakpoints).
- Tab Bar:** Shows four open files: 'Start here', 'adf.cpp', 'C4.cpp', 'in.txt', and 'out.txt'.
- Code Editor:** Displays the source code for 'adf.cpp'. The code defines a function 'FOLLOW' that processes a string 'key' and an integer 'z' by iterating through a vector 'ri' and its elements, performing string manipulations and conditional checks.

```
36     }
37 }
38
39 void FOLLOW(string key,int z){
40
41     int flag = 0;
42
43     for(int i=0;i<ri.size();i++){
44         if (ri[i].find(key) != string::npos) {
45             if(key.size()==1){
46                 for(int j=0;j<ri[i].size();j++){
47                     if(ri[i][j]==key[0]){
48                         if(j+1<ri.size() && ri[i][j+1]!='\'){
49                             flag = 1;
50                             if(isTERMINAL(ri[i][j+1])==false){
51                                 if(z==0)ans += "$,";
52                                 ans += ri[i][j+1];
53                             }
54                         }
55                     }
56                     else{
57                         string g = ri[i];
58                         g.erase(0,1);
59                         FIRST(g);
60                         if(z==0)ans += "$,";
61                         FOLLOW(mpp[ri[i]],1);
62                     }
63                 }
64                 break;
65             }
66         }
67     }
68 }
```

Fig. 14. Code

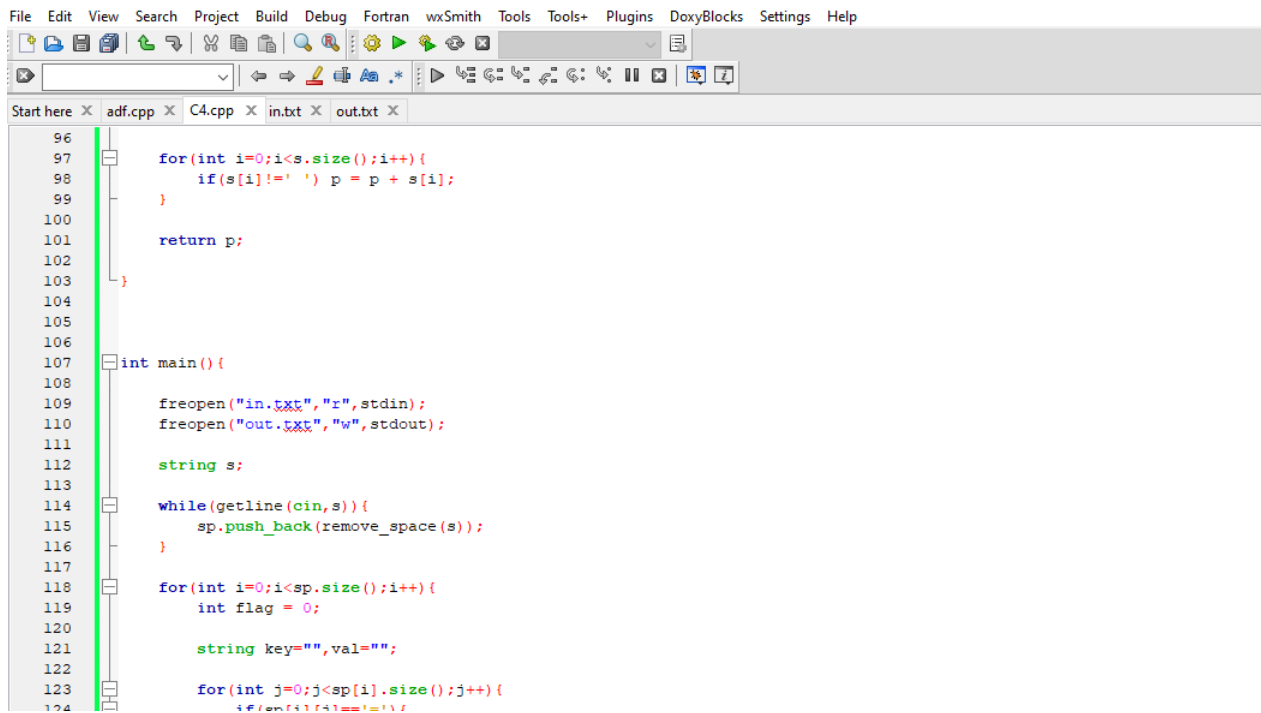




The image shows a screenshot of a code editor with a menu bar and a toolbar. The menu bar includes File, Edit, View, Search, Project, Build, Debug, Fortran, wxSmith, Tools, Tools+, Plugins, DoxyBlocks, Settings, and Help. The toolbar contains various icons for file operations, editing, and execution. The editor has several tabs open: Start here, adf.cpp, C4.cpp, in.txt, and out.txt. The active tab is in.txt, which contains C++ code. The code is as follows:

```
71         for(int j=0;j+1<ri[i].size();j++){
72             if(ri[i][j]==key[0] && ri[i][j+1]==key[1]){
73                 if(j+2>ri[i].size()){
74                     FOLLOW(mpp[ri[i]],1);
75                     if(z==0) ans += ", ";
76                 }
77             }
78             else{
79                 ans="$";
80             }
81         }
82         break;
83     }
84 }
85 if(flag) break;
86 }
87
88
89 }
90
91
92
93 string remove_space(string s){
94
95     string p="";
96
97     for(int i=0;i<s.size();i++){
98         if(s[i]!=' ') p = p + s[i];
99     }
```

Fig. 15. Code



The image shows a screenshot of a code editor with a menu bar (File, Edit, View, Search, Project, Build, Debug, Fortran, wxSmith, Tools, Tools+, Plugins, DoxyBlocks, Settings, Help) and a toolbar. The editor has several tabs open: Start here, adf.cpp, C4.cpp, in.txt, and out.txt. The code is written in C++ and is displayed in a monospaced font with syntax highlighting. The code is as follows:

```
96
97     for(int i=0;i<s.size();i++){
98         if(s[i]!=' ') p = p + s[i];
99     }
100
101     return p;
102 }
103
104
105
106
107 int main() {
108
109     freopen("in.txt","r",stdin);
110     freopen("out.txt","w",stdout);
111
112     string s;
113
114     while(getline(cin,s)){
115         sp.push_back(remove_space(s));
116     }
117
118     for(int i=0;i<sp.size();i++){
119         int flag = 0;
120
121         string key="",val="";
122
123         for(int j=0;j<sp[i].size();j++){
124             if(sp[i][j]!=' '){
```

Fig. 16. Code

```

for(int i=0;i<ke.size();i++){

    string val = mp[ke[i]];
    string v = "";

    for(int j=0;j<val.size();j++){
        if(val[j]!='|') break;
        v += val[j];
    }

    mp[ke[i]] = v;
    mpp[v] = ke[i];
    ri.push_back(v);
}

cerr<<"\nFOLLOW: \n\n";
cout<<"\nFOLLOW: \n\n";

for(int i=0;i<ke.size();i++){
    ans = "";

    FOLLOW(ke[i],0);
    cerr<<"FOLLOW ("<<ke[i]<<") "<<" = {"<<ans<<"}\n";
    cout<<"FOLLOW ("<<ke[i]<<") "<<" = {"<<ans<<"}\n";
}

```

```

C:\Users\Anujit\Documents\C4.exe
FIRST:
FIRST(S) = {a,}
FIRST(X) = {b,}
FIRST(Y) = {b,}
FIRST(Z) = {c,}

FOLLOW:
FOLLOW(S) = {}
FOLLOW(X) = {$,d}
FOLLOW(Y) = {c,$,d}
FOLLOW(Z) = {$, }

Process returned 0 (0x0)   execution time : 0.065 s
Press any key to continue.

```

Fig. 17. Code