

Central University Of Rajasthan

Ajmer, Rajasthan



COMPILER DESIGN LAB

SUBMITTED BY

Ritvik Shukla

2020BTCSE023

Ques-1

```
1  #include <stdio.h>
2  int fa[10][10][10], states[2][10], curr, row = 0, col = 0, sr = 0, sc = 0, th = 0, in;
3  char *str;
4  int nfa(char *string, int state)
5  {
6      int i, j;
7      for (i = 0; i <= row; i++)
8      {
9          if (*string)
10         {
11             curr = fa[state][*string - 97][i];
12             if (curr == -1)
13                 break;
14             if (nfa(string + 1, curr))
15                 return 1;
16         }
17         else
18         {
19             if (states[1][i] == -1)
20                 break;
21             if (state == states[1][i])
22                 return 1;
23         }
24     }
25     return 0;
26 }
27 int main()
28 {
29     FILE *fp;
30     int i, j, k, flag = 0;
31     char c, ch;
32
33     fp = fopen("Nfa_ip.txt", "r");
34     for (i = 0; i < 2; i++)
35         for (j = 0; j < 10; j++)
36             states[i][j] = -1;
37     for (i = 0; i < 10; i++)
38         for (j = 0; j < 10; j++)
39             for (k = 0; k < 10; k++)
```

```
40         fa[i][j][k] = -1;
41     while (fscanf(fp, "%d", &in) != EOF)
42     {
43         fscanf(fp, "%c", &c);
44         if (flag)
45         {
46             states[sr][sc++] = in;
47             if (c == '\n')
48             {
49                 sr++;
50                 sc = 0;
51             }
52         }
53         else if (c == '#')
54         {
55             flag = 1;
56             fa[row][col][th] = in;
57             printf("\nfa[%d][%d][%d]=%d", row, col, th, fa[row][col][th]);
58         }
59         else if (!flag)
60         {
61             fa[row][col][th] = in;
62             printf("\nfa[%d][%d][%d]=%d", row, col, th, fa[row][col][th]);
63             if (c == ',')
64             {
65                 th++;
66             }
67             else if (c == '\n')
68             {
69                 col = 0;
70                 row++;
71                 th = 0;
72             }
73             else if (c != ',')
74             {
75                 col++;
76                 th = 0;
77             }
78         }
79     }
```

```
79 }
80 printf("\n\nEnter the string : \n");
81 scanf("%s", str);
82 if (nfa(str, states[0][0]))
83     printf("\nString Is Accepted");
84 else
85     printf("\nString Not Accepted");
86 getch();
87 return 0;
88 }
89
```

Ques-2

```

1  #include <stdio.h>
2
3  int Fa[10][10][10], states[2][10], row = 0, col = 0, sr = 0, sc = 0, th = 0,
4      in, stat, new_state[10][10], max_inp = -1, no_stat;
5  FILE *fp;
6
7  int search(int search_var)
8  {
9
10     int i;
11     for (i = 0; i < no_stat; i++)
12         if (search_var == states[1][i])
13             return 1;
14     return 0;
15 }
16
17 int sort(int *arr, int count)
18 {
19     int temp, i, j;
20     for (i = 0; i < count - 1; i++)
21     {
22         for (j = i + 1; j < count; j++)
23         {
24             if (arr[i] >= arr[j])
25             {
26                 temp = arr[i];
27                 arr[i] = arr[j];
28                 arr[j] = temp;
29             }
30         }
31     }
32     return 0;
33 }
34
35 int checkcon(int *arr, int *count)
36 {
37     int i, temp, j, k, c, t, m;
38     for (i = 0; i < *count; i++)
39     {
40         . . . . .

```

```

40         if (arr[i] > row)
41         {
42             temp = arr[i];
43             c = 0;
44             t = 0;
45             while (new_state[arr[i]][t] != -1)
46             {
47                 t++;
48                 c++;
49             }
50             for (k = 0; k <= c - 2; k++)
51             {
52                 for (j = 9; j >= i + 1 + k; j--)
53                 {
54                     arr[j] = arr[j - 1];
55                 }
56             }
57
58             t = 0;
59             for (j = i; j < c; j++)
60             {
61                 arr[j] = new_state[temp][t];
62                 t++;
63             }
64         }
65         c = 0;
66         for (i = 0; arr[i] != -1; i++)
67             c++;
68         *count = c;
69         return 0;
70     }
71 }
72
73 int remove_duplicate(int *arr, int *count)
74 {
75     int i, j = 0;
76     for (i = 1; i < *count; i++)
77     {
78         if (arr[i] != arr[j])

```

```

79     {
80         j++;
81         arr[j] = arr[i];
82     }
83 }
84 *count = j + 1;
85 return 0;
86 }
87
88 int check(int i, int j, int c, int *name)
89 {
90     int t, l, f;
91     for (l = 0; l <= stat; l++)
92     {
93         t = 0;
94         f = 0;
95         while (Fa[i][j][t] != -1)
96         {
97             if (Fa[i][j][t] == new_state[l][t])
98                 t++;
99             else
100             {
101                 f = 1;
102                 break;
103             }
104         }
105         if ((t == c) && !f)
106         {
107             *name = l;
108             return 1;
109         }
110     }
111     return 0;
112 }
113
114 int trans(int i, int j, int t, int c, int *count, int *arr)
115 {
116     int k = 0, co, temp;
117     *count = 0;
118
119     for (k = 0; k < c; k++)
120     {
121         temp = Fa[i][j][k];
122         co = 0;
123         while (Fa[temp][t][co] != -1)
124         {
125             arr[*count] = Fa[temp][t][co++];
126             (*count)++;
127         }
128     }
129     return 0;
130 }
131
132 int nfa2dfa(int start, int end)
133 {
134     int j, t, c, i, k, count, arr[10], name, l;
135     for (i = start; i <= end; i++)
136     {
137         for (j = 0; j <= max_inp; j++)
138         {
139             c = 0;
140             t = 0;
141             while (Fa[i][j][t] >= 0)
142             {
143                 t++;
144                 c++;
145             }
146             if (c > 1)
147             {
148                 if (check(i, j, c, &name) == 0)
149                 {
150                     for (k = 0; k < c; k++)
151                     {
152                         new_state[stat][k] = Fa[i][j][k];
153                         for (l = 0; states[l][l] != -1; l++)
154                             if (new_state[stat][k] == states[l][l] && !search(stat))
155                                 states[l][no_stat++] = stat;
156                     }

```

```

157         for (t = 0; t <= max_inp; t++)
158         {
159             count = 0;
160             for (k = 0; k < 10; k++)
161                 arr[k] = -1;
162             trans(i, j, t, c, &count, arr);
163
164             checkcon(arr, &count);
165
166             sort(arr, count);
167             remove_duplicate(arr, &count);
168
169             for (k = 0; k < count; k++)
170                 Fa[stat][t][k] = arr[k];
171         }
172         Fa[i][j][0] = stat++;
173         for (t = 1; t < c; t++)
174             Fa[i][j][t] = -1;
175     }
176     else
177     {
178         Fa[i][j][0] = name;
179         for (t = 1; t < c; t++)
180             Fa[i][j][t] = -1;
181     }
182 }
183 }
184 }
185 return 0;
186 }
187 int main()
188 {
189     int i, j, k, flag = 0, start, end;
190     char c, ch;
191     fp = fopen("Nfa_ip.txt", "r+");
192
193     for (i = 0; i < 2; i++)
194         for (j = 0; j < 10; j++)

```

```

196             states[i][j] = -1;
197
198     for (i = 0; i < 10; i++)
199         for (j = 0; j < 10; j++)
200             new_state[i][j] = -1;
201
202     for (i = 0; i < 10; i++)
203         for (j = 0; j < 10; j++)
204             for (k = 0; k < 10; k++)
205                 Fa[i][j][k] = -1;
206
207     while (fscanf(fp, "%d", &in) != EOF)
208     {
209         fscanf(fp, "%c", &c);
210
211         if (flag)
212         {
213             states[sr][sc++] = in;
214             if (c == '\n')
215             {
216                 sr++;
217                 sc = 0;
218             }
219         }
220         else if (c == '#')
221         {
222             flag = 1;
223             Fa[row][col][th] = in;
224         }
225         else if (!flag)
226         {
227             Fa[row][col][th] = in;
228             if (c == ',')
229             {
230                 th++;
231             }
232             else if (c == '\n')
233             {
234                 if (max_inp < col)

```

```

235         max_inp = col;
236         col = 0;
237         row++;
238         th = 0;
239     }
240     else if (c != ',')
241     {
242         col++;
243         th = 0;
244     }
245     }
246 }
247 no_stat = 0;
248 i = 0;
249 while (states[1][i++] != -1)
250     no_stat++;
251 stat = row + 1;
252 start = 0;
253 end = row;
254 while (1)
255 {
256     nfa2dfa(start, end);
257     start = end + 1;
258     end = row;
259     if (start > end)
260         break;
261 }
262
263 printf("\n\nDFA IS : \n\n");
264 for (i = 0; i <= max_inp; i++)
265     printf("\t%d", i);
266 printf("\n");
267 printf("-----\n");
268
269 for (i = 0; i < stat; i++)
270 {
271     printf("%d-> |", i);
272     for (j = 0; j <= max_inp; j++)
273     {
274         printf("%2d", Fa[i][j][0]);
275     }
276     printf("\n");
277 }
278 printf("\n\n");
279 printf("Total Number Of State Is : %d \n\n", stat);
280 printf("Final States Are : ");
281 for (i = 0; states[1][i] != -1; i++)
282     printf("%d ", states[1][i]);
283
284 printf("\n\n");
285 return 0;
286 }

```

Output

```

ritvik@ritvik-G3-3500:~/OpenCv$ gcc cd.c
ritvik@ritvik-G3-3500:~/OpenCv$ ./a.out

```

DFA IS :

	0	1
0->	3	1
1->	-1	2
2->	-1	-1
3->	-1	2

Total Number Of State Is : 4

Final States Are : 2 3

Ques-3

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<string.h>
4
5  //Global Variables
6  int z = 0, i = 0, j = 0, c = 0;
7
8  // Modify array size to increase
9  // length of string to be parsed
10 char a[16], ac[20], stk[15], act[10];
11
12 // This Function will check whether
13 // the stack contain a production rule
14 // which is to be Reduce.
15 // Rules can be E->2E2 , E->3E3 , E->4
16 void check()
17 {
18     // Copying string to be printed as action
19     strcpy(ac,"REDUCE TO E -> ");
20
21     // c=length of input string
22     for(z = 0; z < c; z++)
23     {
24         //checking for producing rule E->4
25         if(stk[z] == '4')
26         {
27             printf("%s4", ac);
28             stk[z] = 'E';
29             stk[z + 1] = '\0';
30
31             //printing action
32             printf("\n%s\t%s\t", stk, a);
33         }
34     }
35
36     for(z = 0; z < c - 2; z++)
37     {
38         //checking for another production
39         if(stk[z] == '2' && stk[z + 1] == 'E' &&

```

```

40         stk[z + 2] == '2')
41     {
42         printf("%s2E2", ac);
43         stk[z] = 'E';
44         stk[z + 1] = '\0';
45         stk[z + 2] = '\0';
46         printf("\n%s\t%s\t", stk, a);
47         i = i - 2;
48     }
49 }
50
51
52 for(z=0; z<c-2; z++)
53 {
54     //checking for E->3E3
55     if(stk[z] == '3' && stk[z + 1] == 'E' &&
56         stk[z + 2] == '3')
57     {
58         printf("%s3E3", ac);
59         stk[z]='E';
60         stk[z + 1]='\0';
61         stk[z + 1]='\0';
62         printf("\n%s\t%s\t", stk, a);
63         i = i - 2;
64     }
65 }
66 return ; //return to main
67 }
68
69 //Driver Function
70 int main()
71 {
72     printf("GRAMMAR is -\nE->2E2 \nE->3E3 \nE->4\n");
73
74     // a is input string
75     strcpy(a,"32423");
76
77     // strlen(a) will return the length of a to c
78     c=strlen(a);

```

```

79
80 // "SHIFT" is copied to act to be printed
81 strcpy(act,"SHIFT");
82
83 // This will print Labels (column name)
84 printf("\nstack \t input \t action");
85
86 // This will print the initial
87 // values of stack and input
88 printf("\n\t%s\t", a);
89
90 // This will Run upto length of input string
91 for(i = 0; j < c; i++, j++)
92 {
93     // Printing action
94     printf("%s", act);
95
96     // Pushing into stack
97     stk[i] = a[j];
98     stk[i + 1] = '\0';
99
100    // Moving the pointer
101    a[j]=' ';
102
103    // Printing action
104    printf("\n\t%s\t%s\t", stk, a);
105
106    // Call check function ..which will
107    // check the stack whether its contain
108    // any production or not
109    check();
110 }
111
112 // Rechecking last time if contain
113 // any valid production then it will
114 // replace otherwise invalid
115 check();
116
117 // if top of the stack is E(starting symbol)

```

```

118 // then it will accept the input
119 if(stk[0] == 'E' && stk[1] == '\0')
120     printf("Accept\n");
121 else //else reject
122     printf("Reject\n");
123
124 }

```

Output

```

• ritvik@ritvik-G3-3500:~/OpenCv$ gcc cd.c
• ritvik@ritvik-G3-3500:~/OpenCv$ ./a.out
GRAMMAR is -
E->2E2
E->3E3
E->4

stack  input  action
$      32423$ SHIFT
$3     2423$  SHIFT
$32    423$   SHIFT
$324   23$    REDUCE TO E -> 4
$32E   23$    SHIFT
$32E2  3$     REDUCE TO E -> 2E2
$3E    3$     SHIFT
$3E3   $      REDUCE TO E -> 3E3
$E     $      Accept
• ritvik@ritvik-G3-3500:~/OpenCv$ █

```

Ques-4


```

C cd.c > main()
1  #include<stdio.h>
2  #include<string.h>
3
4  char *input;
5  int i=0;
6  char lasthandle[6],stack[50],handles[][5]={"}E(", "E"E", "E+E", "i", "E^E");
7  //(E) becomes )E( when pushed to stack
8
9  int top=0,l;
10 char prec[9][9]={
11
12     | | | | | | | | | | /*input*/
13
14     /*stack + - * / ^ i ( ) $ */
15
16     /* + */ '>', '>', '<', '<', '<', '<', '<', '>', '>',
17
18     /* - */ '>', '>', '<', '<', '<', '<', '<', '>', '>',
19
20     /* * */ '>', '>', '>', '>', '<', '<', '<', '>', '>',
21
22     /* / */ '>', '>', '>', '>', '<', '<', '<', '>', '>',
23
24     /* ^ */ '>', '>', '>', '>', '<', '<', '<', '>', '>',
25
26     /* i */ '>', '>', '>', '>', '>', '>', 'e', 'e', '>', '>',
27
28     /* ( */ '<', '<', '<', '<', '<', '<', '<', '>', 'e',
29
30     /* ) */ '>', '>', '>', '>', '>', '>', 'e', 'e', '>', '>',
31
32     /* $ */ '<', '<', '<', '<', '<', '<', '<', '<', '>',
33
34     };
35
36 int getIndex(char c)
37 {
38     switch(c)
39     {
40

```

```

C cd.c > main()
40     case '+':return 0;
41     case '-':return 1;
42     case '*':return 2;
43     case '/':return 3;
44     case '^':return 4;
45     case 'i':return 5;
46     case '(':return 6;
47     case ')':return 7;
48     case '$':return 8;
49     }
50 }
51
52
53 int shift()
54 {
55     stack[++top]=*(input+i++);
56     stack[top+1]='\0';
57 }
58
59
60 int reduce()
61 {
62     int i,len,found,t;
63     for(i=0;i<5;i++)//selecting handles
64     {
65         len=strlen(handles[i]);
66         if(stack[top]==handles[i][0]&&top+1>=len)
67         {
68             found=1;
69             for(t=0;t<len;t++)
70             {
71                 if(stack[top-t]!=handles[i][t])
72                 {
73                     found=0;
74                     break;
75                 }
76             }
77             if(found==1)
78             {

```

```

C cd.c > main()
79     stack[top-t+1]='E';
80     top=top-t+1;
81     strcpy(lasthandle,handles[i]);
82     stack[top+1]='\0';
83     return 1;//successful reduction
84 }
85 }
86 }
87 return 0;
88 }
89
90
91
92 void dispstack()
93 {
94     int j;
95     for(j=0;j<=top;j++)
96         printf("%c",stack[j]);
97 }
98
99
100
101 void dispinput()
102 {
103     int j;
104     for(j=i;j<=l;j++)
105         printf("%c",*(input+j));
106 }
107
108
109
110 void main()
111 {
112     int j;
113
114     input=(char*)malloc(50*sizeof(char));
115     printf("\nEnter the string\n");
116     scanf("%s",input);
117     input=strcat(input,"$");

```

```

C cd.c > main()
118     l=strlen(input);
119     strcpy(stack,"$");
120     printf("\nSTACK\tINPUT\tACTION");
121     while(i<=l)
122     {
123         shift();
124         printf("\n");
125         dispstack();
126         printf("\t");
127         dispinput();
128         printf("\tShift");
129         if(prec[getIndex(stack[top])][getIndex(input[i])]=='>')
130         {
131             while(reduce())
132             {
133                 printf("\n");
134                 dispstack();
135                 printf("\t");
136                 dispinput();
137                 printf("\tReduced: E->%s",lasthandle);
138             }
139         }
140     }
141
142     if(strcmp(stack,"$E$")==0)
143         printf("\nAccepted;");
144     else
145         printf("\nNot Accepted;");
146 }

```

Output

```
ritvik@ritvik-G3-3500:~/OpenCv$ ./a.out
```

```
Enter the string
i+(i*i)+i
```

STACK	INPUT	ACTION
\$i	+(i*i)+i\$	Shift
\$E	+(i*i)+i\$	Reduced: E->i
\$E+	(i*i)+i\$	Shift
\$E+(i*i)+i\$	Shift
\$E+(i	*i)+i\$	Shift
\$E+(E	*i)+i\$	Reduced: E->i
\$E+(E*	i)+i\$	Shift
\$E+(E*i)i\$	Shift
\$E+(E*iE)i\$	Reduced: E->i
\$E+(E)i\$	Reduced: E->E*i
\$E+(E)	+i\$	Shift
\$E+E	+i\$	Reduced: E->)E(
\$E	+i\$	Reduced: E->E+E
\$E+	i\$	Shift
\$E+i	\$	Shift
\$E+E	\$	Reduced: E->i
\$E	\$	Reduced: E->E+E
\$E\$		Shift
\$E\$		Shift

Ques-5

```
C cd.c > F()
1  #include <stdio.h>
2  #include <string.h>
3  #include <ctype.h>
4
5  char input[10];
6  int i, error;
7  void E();
8  void T();
9  void Eprime();
10 void Tprime();
11 void F();
12 main()
13 {
14     i = 0;
15     error = 0;
16     printf("Enter an arithmetic expression : "); // Eg: a+a*a
17     gets(input);
18     E();
19     if (strlen(input) == i && error == 0)
20         printf("\nAccepted....!!\n");
21     else
22         printf("\nRejected....!!\n");
23 }
24
25 void E()
26 {
27     T();
28     Eprime();
29 }
30 void Eprime()
31 {
32     if (input[i] == '+')
33     {
34         i++;
35         T();
36         Eprime();
37     }
38 }
39 void T()
40 {
```

```
C cd.c > F()
40 {
41     F();
42     Tprime();
43 }
44 void Tprime()
45 {
46     if (input[i] == '*')
47     {
48         i++;
49         F();
50         Tprime();
51     }
52 }
53 void F()
54 {
55     if (isalnum(input[i]))
56         i++;
57     else if (input[i] == '(')
58     {
59         i++;
60         E();
61         if (input[i] == ')')
62             i++;
63         else
64             error = 1;
65     }
66     else
67         error = 1;
68 }
69 }
70 }
```

Output

```
• ritvik@ritvik-G3-3500:~/OpenCv$ ./a.out
Enter an arithmetic expression : a+a*a

Accepted....!!
○ ritvik@ritvik-G3-3500:~/OpenCv$
```

Ques-6

```
C cd.c × C nfa.c Preview CHANGELOG.md Go for VS Code launch.json Nfa_lp.txt
C cd.c > ...
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 struct op
5 {
6     char l;
7     char r[20];
8 } op[10], pr[10];
9 void main()
10 {
11     int a, i, k, j, n, z = 0, m, q;
12     char *p, *l;
13     char temp, t;
14     char *tem;
15     printf("enter no of values");
16     scanf("%d", &n);
17     for (i = 0; i < n; i++)
18     {
19         printf("\tleft\t");
20         op[i].l = getche();
21         printf("\tright\t");
22         scanf("%s", op[i].r);
23     }
24     printf("intermediate Code\n");
25     for (i = 0; i < n; i++)
26     {
27         printf("%c=", op[i].l);
28         printf("%s\n", op[i].r);
29     }
30     for (i = 0; i < n - 1; i++)
31     {
32         temp = op[i].l;
33         for (j = 0; j < n; j++)
34         {
35             p = strchr(op[j].r, temp);
36             if (p)
37             {
38                 pr[z].l = op[i].l;
39                 strcpy(pr[z].r, op[i].r);
40                 ...
41             }
42         }
43     }
44 }
```

```

C cd.c x C nfa.c Preview CHANGELOG.md Go for VS Code launch.json Nfa_lp.txt
C cd.c > ...
40     }
41     }
42     }
43 }
44 pr[z].l = op[n - 1].l;
45 strcpy(pr[z].r, op[n - 1].r);
46 z++;
47 printf("\nafter dead code elimination\n");
48 for (k = 0; k < z; k++)
49 {
50     printf("%c\t=", pr[k].l);
51     printf("%s\n", pr[k].r);
52 }
53 // sub expression elimination
54 for (m = 0; m < z; m++)
55 {
56     temp = pr[m].r;
57     for (j = m + 1; j < z; j++)
58     {
59         p = strstr(temp, pr[j].r);
60         if (p)
61         {
62             t = pr[j].l;
63             pr[j].l = pr[m].l;
64             for (i = 0; i < z; i++)
65             {
66                 l = strchr(pr[i].r, t);
67                 if (l)
68                 {
69                     a = l - pr[i].r;
70                     // printf("pos: %d", a);
71                     pr[i].r[a] = pr[m].l;
72                 }
73             }
74         }
75     }
76 }
77 printf("eliminate common expression\n");
78 for (i = 0; i < z; i++)

```

```

C cd.c x C nfa.c Preview CHANGELOG.md Go for VS Code launch.json Nfa_lp.txt
C cd.c > ...
79     }
80     printf("%c\t=", pr[i].l);
81     printf("%s\n", pr[i].r);
82 }
83 // duplicate production elimination
84 for (i = 0; i < z; i++)
85 {
86     for (j = i + 1; j < z; j++)
87     {
88         q = strcmp(pr[i].r, pr[j].r);
89         if ((pr[i].l == pr[j].l) && !q)
90         {
91             pr[i].l = '\0';
92             strcpy(pr[i].r, '\0');
93         }
94     }
95     printf("optimized code");
96     for (i = 0; i < z; i++)
97     {
98         if (pr[i].l != '\0')
99         {
100             printf("%c=", pr[i].l);
101             printf("%s\n", pr[i].r);
102         }
103     }
104     getch();
105 }
106 }

```

Output

```

enter no of values3
      left   1      right:  1
      left   1      right:  2
      left   3      right:  4
intermediate Code
1=1
1=2
3=4

after dead code elimination
1      =1
1      =2
3      =4
eliminate common expression
1      =1
1      =2
3      =4
optimized code1=1
1=2
3=4

-----
Process exited after 36.74 seconds with return value 4
Press any key to continue . . .

```

Ques-7

```

C cd.c x C nfa.c Preview CHANGELOG.md Go for VS Code launch.json Nfa_ip.txt
C cd.c > ...
1 #include <stdio.h>
2 #include <csd.h>
3 #include <string.h>
4 char op[2], arg1[5], arg2[5], result[5];
5 void main()
6 {
7     FILE *fp1, *fp2;
8     fp1 = fopen("input.txt", "r");
9     fp2 = fopen("output.txt", "w");
10    while (!feof(fp1))
11    {
12
13        fscanf(fp1, "%s%s%s", op, arg1, arg2, result);
14        if (strcmp(op, "+") == 0)
15        {
16            fprintf(fp2, "\nMOV R0,%s", arg1);
17            fprintf(fp2, "\nADD R0,%s", arg2);
18            fprintf(fp2, "\nMOV %s,R0", result);
19        }
20        if (strcmp(op, "*") == 0)
21        {
22            fprintf(fp2, "\nMOV R0,%s", arg1);
23            fprintf(fp2, "\nMUL R0,%s", arg2);
24            fprintf(fp2, "\nMOV %s,R0", result);
25        }
26        if (strcmp(op, "-") == 0)
27        {
28            fprintf(fp2, "\nMOV R0,%s", arg1);
29            fprintf(fp2, "\nSUB R0,%s", arg2);
30            fprintf(fp2, "\nMOV %s,R0", result);
31        }
32        if (strcmp(op, "/") == 0)
33        {
34            fprintf(fp2, "\nMOV R0,%s", arg1);
35            fprintf(fp2, "\nDIV R0,%s", arg2);
36            fprintf(fp2, "\nMOV %s,R0", result);
37        }
38        if (strcmp(op, "=") == 0)
39        {
40            fprintf(fp2, "\nMOV R0,%s", arg1);
41            fprintf(fp2, "\nMOV %s,R0", result);
42        }
43    }
44    fclose(fp1);
45    fclose(fp2);
46    return;
47 }
48

```

Output

Open

input.txt

Save

```
1 + a b t1
2 + c d t2
3 - t1 t2 t
4 = t ? x
5
```

Open

output.txt

Save

```
1 MOV R0,a
2 ADD R0,b
3 MOV t1,R0
4 MOV R0,c
5 MUL R0,d
6 MOV t2,R0
7 MOV R0,t1
8 SUB R0,t2
9 MOV t,R0
10 MOV R0,t
11 MOV x,R0
12 MOV R0,t
13 MOV x,R0
14
```