

Name: Brijesh Rameshbhai Rohit

Admission number: U19CS009

SS-ASSIGNMENT-09

1. Write a lex program to identify identifiers, constants and keywords (int, float) used in c/c++ from a given input file.

Code:

```
%{
    #include<stdio.h>
}%

%%

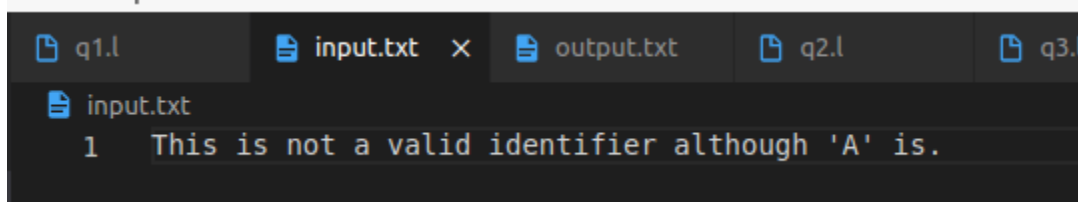
['']([a-zA-z0-9])[''] {fprintf(yyout,"%s\t: Constant\n",yytext);};
([0-9])+ {fprintf(yyout,"%s\t: Constant\n",yytext);};
int {fprintf(yyout,"%s\t: Keyword\n",yytext);};
float {fprintf(yyout,"%s\t: Keyword\n",yytext);};
[a-zA-Z_]([a-zA-Z0-9])* {fprintf(yyout,"%s\t: Identifier\n",yytext);};
([a-zA-Z0-9])* {fprintf(yyout,"%s\t: Invalid\n",yytext);};

%%

int main()
{
    extern FILE * yyin,*yyout;
    yyin=fopen("input.txt","r");
    yyout=fopen("output.txt","w");
    yylex();
    return 0;
}
```

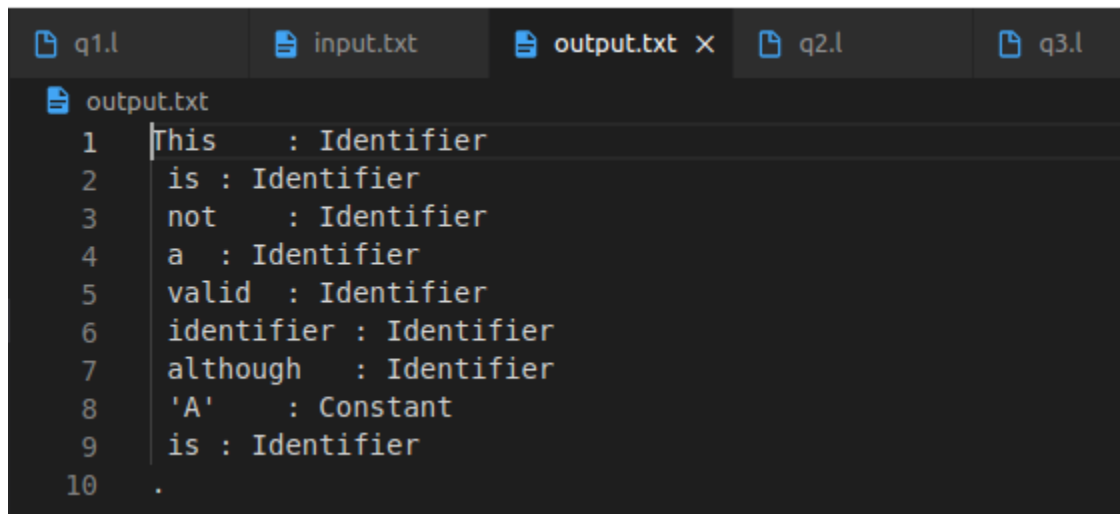
Output:

Input-file :



```
q1.l  input.txt  x  output.txt  q2.l  q3.l
input.txt
1 This is not a valid identifier although 'A' is.
```

Output-file :



```
q1.l  input.txt  output.txt x  q2.l  q3.l
output.txt
1  This      : Identifier
2  is : Identifier
3  not      : Identifier
4  a  : Identifier
5  valid   : Identifier
6  identifier : Identifier
7  although : Identifier
8  'A'     : Constant
9  is : Identifier
10 .
```

2. Write a lex Program to find octal and hexadecimal numbers.

Code:

```
%{
    #include<stdio.h>
}%

%%

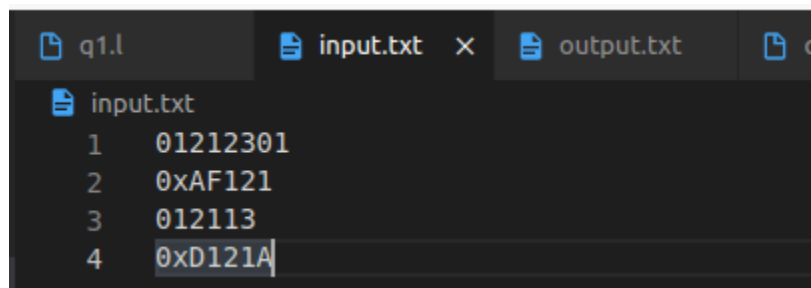
[0]([0-7])+ {fprintf(yyout,"%s : Octal Number\n",yytext);};
[0][x]([0-9a-fA-F])+ {fprintf(yyout,"%s : Hexadecimal Number\n",yytext);};
([0-9])+ {fprintf(yyout,"%s : Decimal Number\n",yytext);};

%%

int main()
{
    extern FILE * yyin,*yyout;
    yyin=fopen("input.txt","r");
    yyout=fopen("output.txt","w");
    yylex();
    return 0;
}
```

Output:

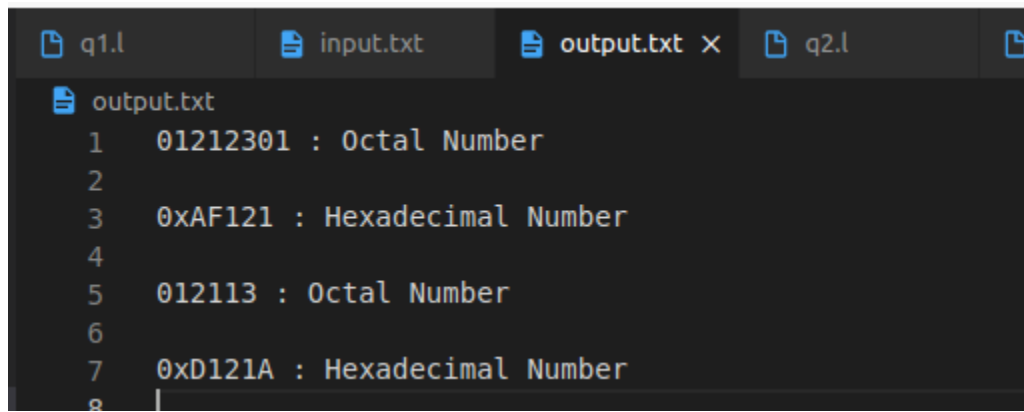
Input-file :



The screenshot shows a text editor window with three tabs: 'q1.l', 'input.txt', and 'output.txt'. The 'input.txt' tab is active, displaying the following content:

Line	Content
1	01212301
2	0xAF121
3	012113
4	0xD121A

Output-file :



The image shows a terminal window with a dark background. At the top, there are four tabs: 'q1.l', 'input.txt', 'output.txt x', and 'q2.l'. The 'output.txt' tab is active. Below the tabs, the text 'output.txt' is displayed. The main content of the terminal shows a list of numbers and their corresponding base representations, numbered 1 through 8. The text is as follows:

```
1 01212301 : Octal Number
2
3 0xAF121 : Hexadecimal Number
4
5 012113 : Octal Number
6
7 0xD121A : Hexadecimal Number
8
```

3. Write a lex program to count and display Single line and multiline comments.

Code:

```
%{
    #include<stdio.h>
    int scount=0 ,mcount=0;
}%

%%

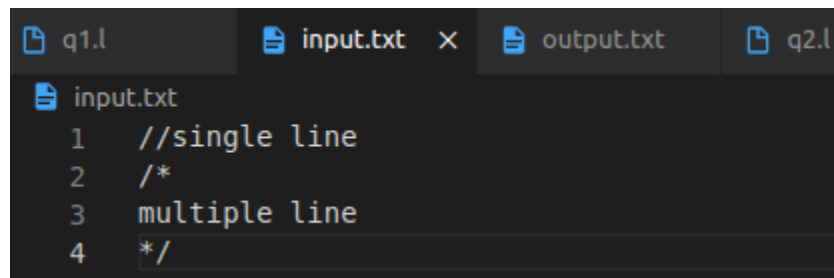
[//][/]( [a-zA-Z0-9 \t] ) * [ \n ] {fprintf(yyout, "\nSingle line Comment
:\n%s", yytext); scount++;};
[/][*]( [a-zA-Z0-9 \t \n] ) * [*][/] {fprintf(yyout, "\nMultiline Comment
:\n%s", yytext); mcount++;};
end {fprintf(yyout, "\nSingle line comment count : %d\nMultiple Line
comment count : %d", scount, mcount);};

%%

int main()
{
    extern FILE * yyin, *yyout;
    yyin=fopen("input.txt", "r");
    yyout=fopen("output.txt", "w");
    yylex();
    return 0;
}
```

Output:

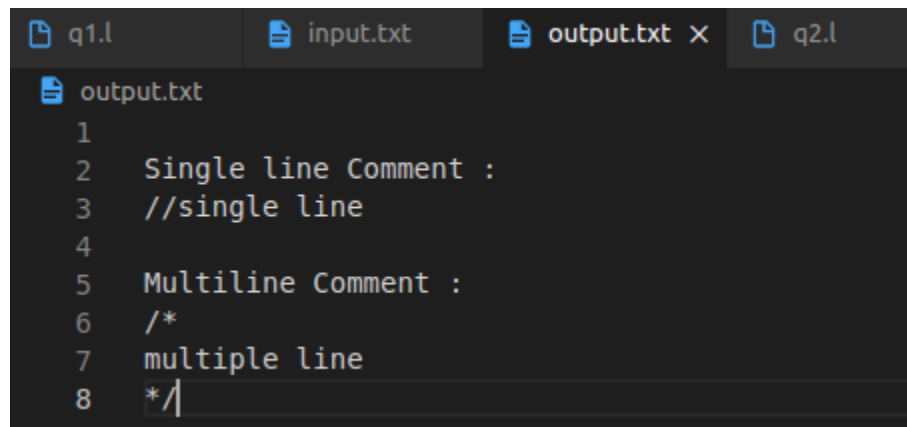
Input-file :



The screenshot shows a code editor with four tabs: q1.l, input.txt, output.txt, and q2.l. The 'input.txt' tab is active, displaying the following content:

```
1 //single line
2 /*
3 multiple line
4 */
```

Output-file :



```
1
2  Single line Comment :
3  //single line
4
5  Multiline Comment :
6  /*
7  multiple line
8  */
```

4. Write a lex program to count no of negative, positive and zero numbers.

Code:

```
%{
    #include<stdio.h>
    int nn=0,pn=0,zn=0;
}%

%%

[-] [1-9] ([0-9])* {nn++;};
[+] [1-9] ([0-9])* {pn++;};
[1-9] ([0-9])* {pn++;};
([0])+ {zn++;};

%%

int main()
{
    yylex();
    printf("\nNumber of Positive Numbers: %d",pn);
    printf("\nNumber of Negative Numbers: %d",nn);
    printf("\nNumber of Zero Numbers: %d\n",zn);
    return 0;
}
```

Output:

```
brijesh@brijesh-VirtualBox:~/Documents/ss-assign09$ lex q4.l
brijesh@brijesh-VirtualBox:~/Documents/ss-assign09$ gcc lex.yy.c -lfl
brijesh@brijesh-VirtualBox:~/Documents/ss-assign09$ ./a.out
1
0
-1
1
-1
0
2
-4
6
10
-11
Number of Positive Numbers: 5
Number of Negative Numbers: 4
Number of Zero Numbers: 2
```


5. Write a Lex program to accept strings that start with aa and end with bcd.

Code:

```
%{
    #include<stdio.h>
}%

%%

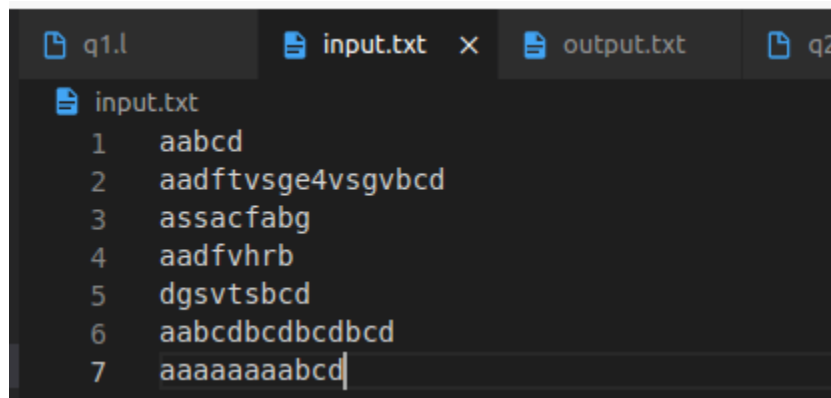
aa([a-zA-Z])*bcd {fprintf(yyout,"%s : Accepted",yytext);};
([a-zA-Z])+ {fprintf(yyout,"%s : Declined",yytext);};

%%

int main()
{
    extern FILE * yyin,*yyout;
    yyin=fopen("input.txt","r");
    yyout=fopen("output.txt","w");
    yylex();
    return 0;
}
```

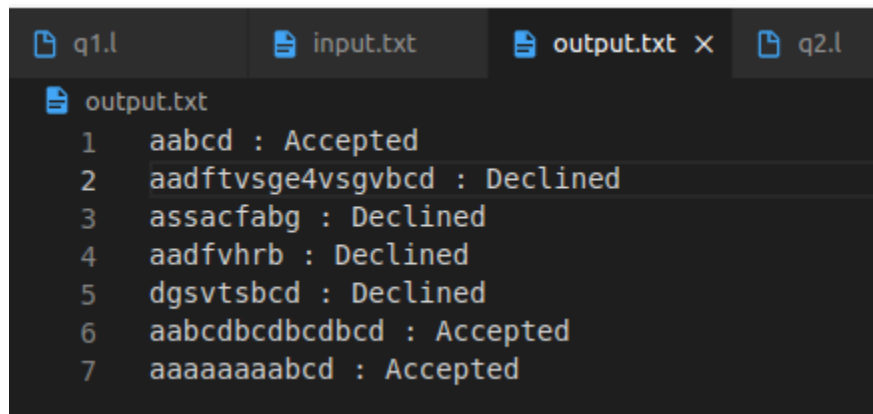
Output:

Input-file :



```
q1.l  input.txt  output.txt  q2
input.txt
1 aabcd
2 aadftvsge4vsgvbcd
3 assacfabg
4 aadfvlrb
5 dgsvtsbcd
6 aabcbcbcbcbcd
7 aaaaaaabcd
```

Output-file :



The image shows a terminal window with four tabs: q1.l, input.txt, output.txt (active), and q2.l. The active tab displays the contents of output.txt, which lists seven lines of input strings and their corresponding acceptance status.

```
output.txt
1 aabcd : Accepted
2 aadftvsge4vsgvbcd : Declined
3 assacfabg : Declined
4 aadfvrhb : Declined
5 dgsvtsbcd : Declined
6 aabcbcbcbcbcd : Accepted
7 aaaaaaabcd : Accepted
```