SS ASSIGNMENT – 9

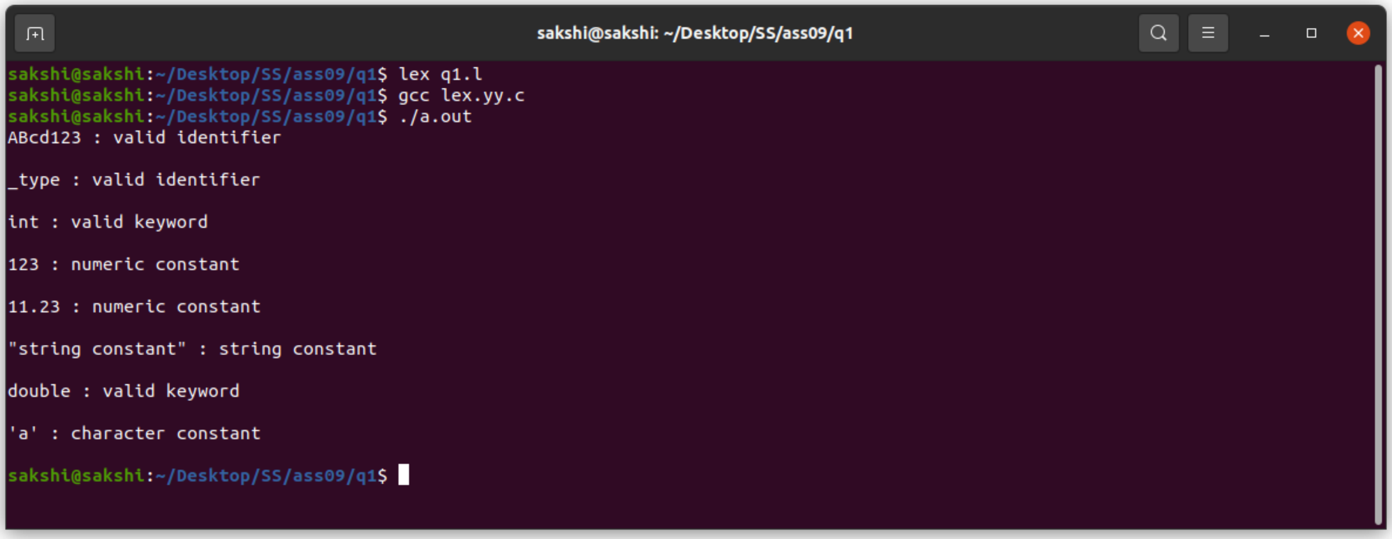
1. **Write a lex program to identify identifiers, constants and keywords (int, float) used in c/c++ from a given input file.**

Source Code:

```
%{
%}

alphabet[a-zA-Z]
digit[0-9]

%%
int|float|double|char|while|for|if|else|switch|break { printf("%s : valid keyword\n",yytext);
}

([_]|{alphabet})({alphabet}|{digit}|[_])* { printf("%s : valid identifier\n",yytext); }

({digit})+|({digit})+[.]({digit})* { printf("%s : numeric constant\n",yytext); }

["].*["] { printf("%s : string constant\n",yytext); }

[']{alphabet}['] { printf("%s : character constant\n",yytext); }
%%

int yywrap(){}

int main(){
    yyin=fopen("input1.txt","r");
    yylex();
    return 0;
}
```

Input.txt:

```
ABcd123
_type
int
123
11.23
"string constant"
double
'a'
```

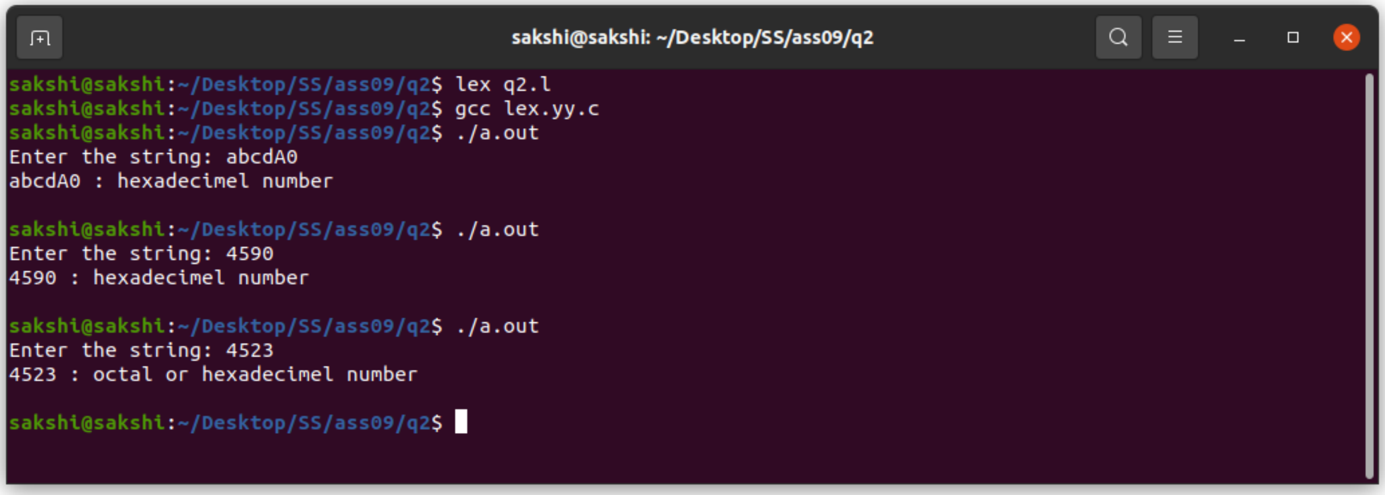


```
sakshi@sakshi:~/Desktop/SS/ass09/q1$ lex q1.l
sakshi@sakshi:~/Desktop/SS/ass09/q1$ gcc lex.yy.c
sakshi@sakshi:~/Desktop/SS/ass09/q1$ ./a.out
ABcd123 : valid identifier

_type : valid identifier

int : valid keyword

123 : numeric constant

11.23 : numeric constant

"string constant" : string constant

double : valid keyword

'a' : character constant

sakshi@sakshi:~/Desktop/SS/ass09/q1$
```

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

Source Code:

```
%{

%}

octal [0-7]
hex [0-9ABCDEFabcdef]


%%
{octal}+ { printf("%s : octal or hexadecimel number\n",yytext); }

{hex}+ { printf("%s : hexadecimel number\n",yytext); }
%%

int yywrap(void) {
    return 1;
}

int main(){
    printf("Enter the string: ");
    yylex();
    return 0;
}
```

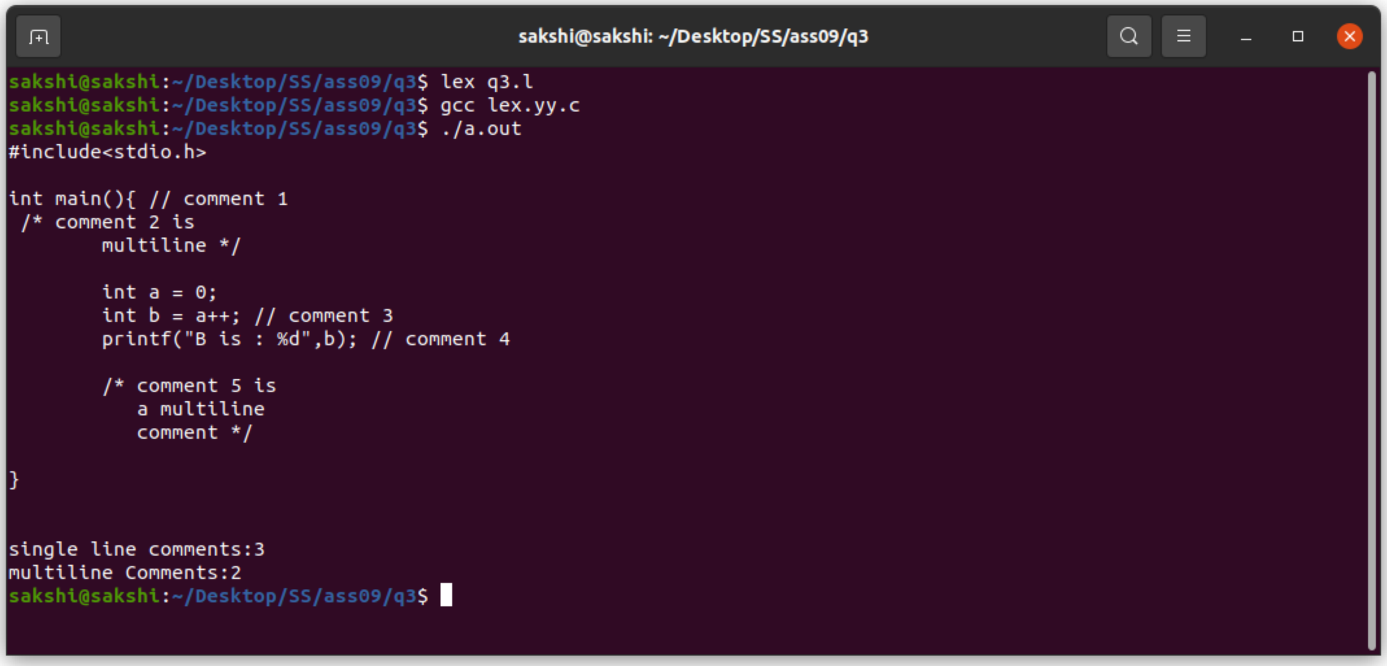


```
sakshi@sakshi:~/Desktop/SS/ass09/q2$ lex q2.l
sakshi@sakshi:~/Desktop/SS/ass09/q2$ gcc lex.yy.c
sakshi@sakshi:~/Desktop/SS/ass09/q2$ ./a.out
Enter the string: abcdA0
abcdA0 : hexadecimel number

sakshi@sakshi:~/Desktop/SS/ass09/q2$ ./a.out
Enter the string: 4590
4590 : hexadecimel number

sakshi@sakshi:~/Desktop/SS/ass09/q2$ ./a.out
Enter the string: 4523
4523 : octal or hexadecimel number

sakshi@sakshi:~/Desktop/SS/ass09/q2$
```

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

Source Code:

```
%{
    int singleComment=0;
    int multilineComment=0;
    int flag=0;
%}

%%
"/*" {flag=1;printf("%s",yytext);}

"*/" { printf("%s",yytext);if(flag==1){ multilineComment++; flag=0;} }

[/][/].*\n { printf("%s",yytext);if(flag==0)singleComment++; }
%%

int yywrap() {}

int main(){
    yyin=fopen("input.txt","r");
    yylex();
    printf("\n\nsingle line comments:%d\n",singleComment);
    printf("multiline Comments:%d\n",multilineComment);
    return 0;
}
```

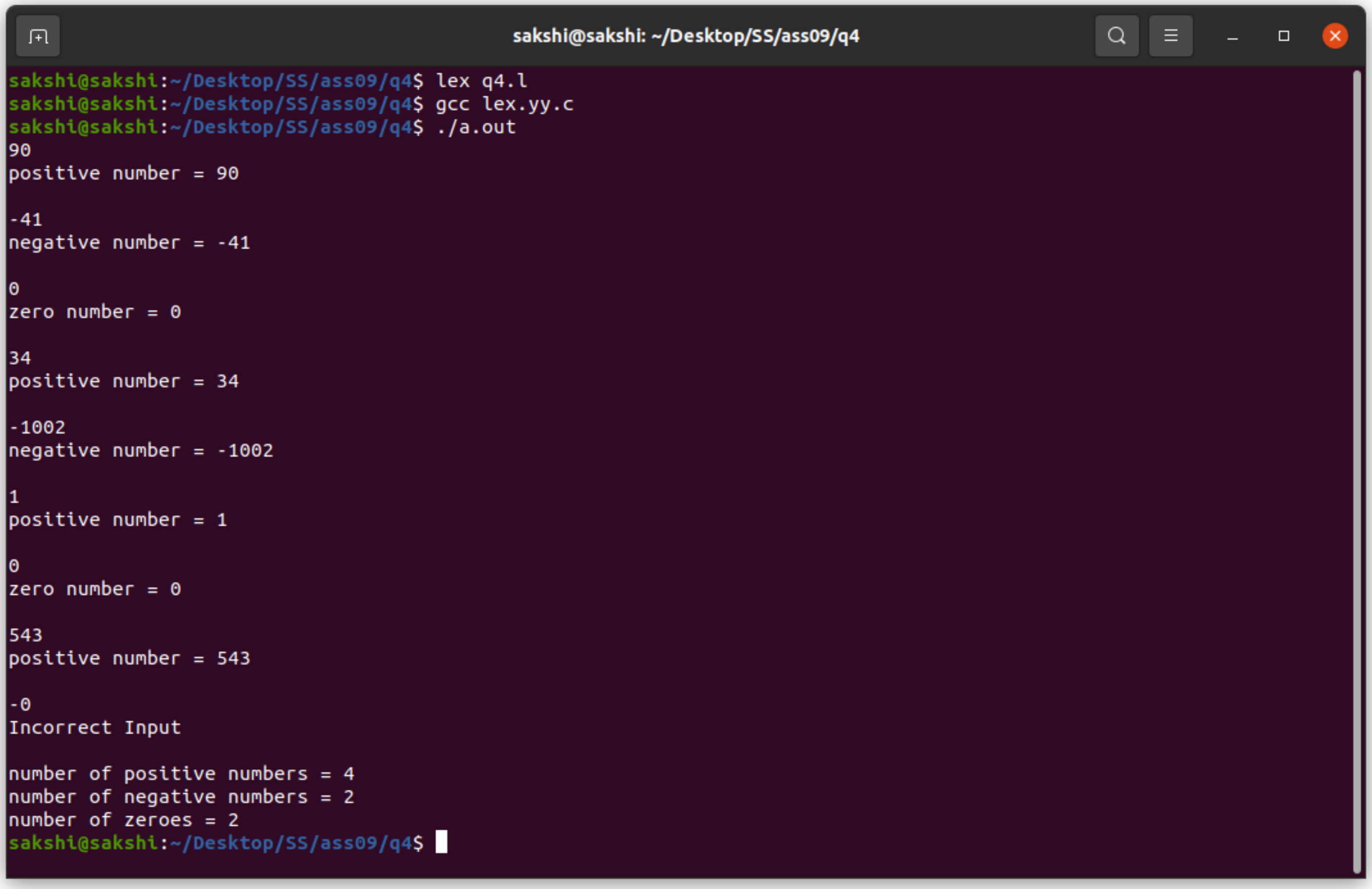

sakshi@sakshi: ~/Desktop/SS/ass09/q3

```
sakshi@sakshi:~/Desktop/SS/ass09/q3$ lex q3.l
sakshi@sakshi:~/Desktop/SS/ass09/q3$ gcc lex.yy.c
sakshi@sakshi:~/Desktop/SS/ass09/q3$ ./a.out
#include<stdio.h>

int main(){ // comment 1
 /* comment 2 is
        multiline */

        int a = 0;
        int b = a++; // comment 3
        printf("B is : %d",b); // comment 4

        /* comment 5 is
           a multiline
           comment */

}


single line comments:3
multiline Comments:2
sakshi@sakshi:~/Desktop/SS/ass09/q3$ 
```

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

Source Code:

```
%{
    int positive_no = 0, negative_no = 0, zero_no = 0;
%}

/* Rules for identifying and counting
positive and negative numbers*/

%%
^[-][1-9][0-9]+ {negative_no++;
            printf("negative number = %s\n",
                yytext);} // negative number

[0] {zero_no++;
        printf("zero number = %s\n",
                yytext);} // zero number

[0-9]+ {positive_no++;
        printf("positive number = %s\n",
                yytext);} // positive number

.* {printf("Incorrect Input\n");}

%%


int yywrap(){}
int main(){
    yylex();
    printf ("number of positive numbers = %d\n"
        "number of negative numbers = %d\n"
        "number of zeroes = %d\n",
                    positive_no,negative_no,zero_no);
    return 0;
}
```



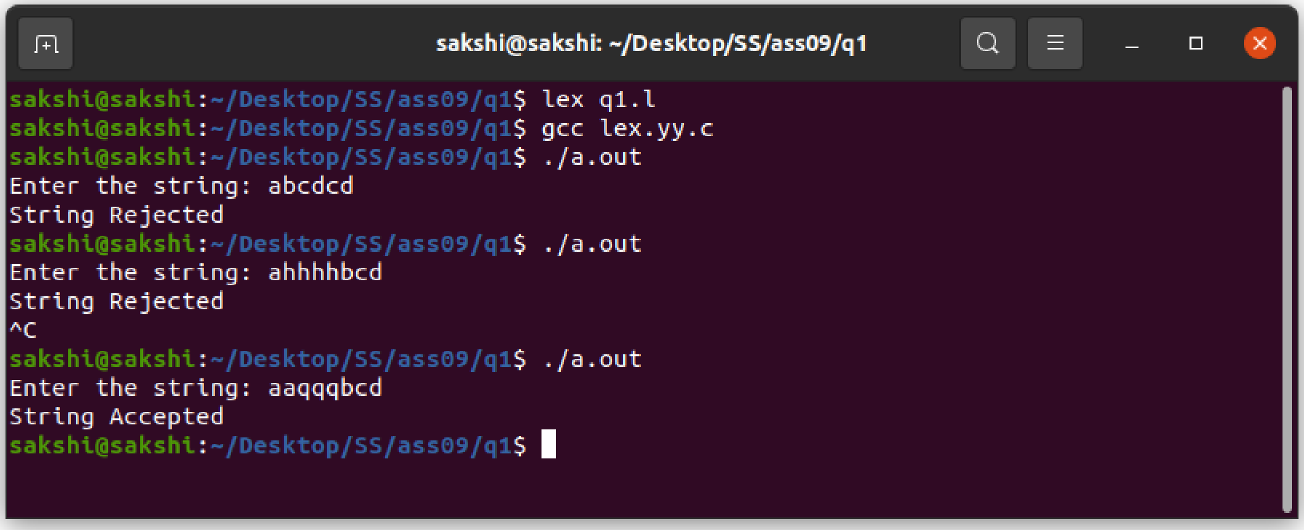
```
sakshi@sakshi:~/Desktop/SS/ass09/q4$ lex q4.l
sakshi@sakshi:~/Desktop/SS/ass09/q4$ gcc lex.yy.c
sakshi@sakshi:~/Desktop/SS/ass09/q4$ ./a.out
90
positive number = 90

-41
negative number = -41

0
zero number = 0

34
positive number = 34

-1002
negative number = -1002

1
positive number = 1

0
zero number = 0

543
positive number = 543

-0
Incorrect Input

number of positive numbers = 4
number of negative numbers = 2
number of zeroes = 2
sakshi@sakshi:~/Desktop/SS/ass09/q4$ 
```

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

Source Code:

```
%{

%}

str (aa).*(bcd)

%%
```

```
{str} printf("String Accepted");

.* printf("String Rejected");

%%

int yywrap(void) {
    return 1;
}

int main(){
    printf("Enter the string: ");
    yylex();
    return 0;
}
```



```
sakshi@sakshi:~/Desktop/SS/ass09/q1$ lex q1.l
sakshi@sakshi:~/Desktop/SS/ass09/q1$ gcc lex.yy.c
sakshi@sakshi:~/Desktop/SS/ass09/q1$ ./a.out
Enter the string: abcdcd
String Rejected
sakshi@sakshi:~/Desktop/SS/ass09/q1$ ./a.out
Enter the string: ahhhhbcd
String Rejected
^C
sakshi@sakshi:~/Desktop/SS/ass09/q1$ ./a.out
Enter the string: aaqqqbcd
String Accepted
sakshi@sakshi:~/Desktop/SS/ass09/q1$ 
```