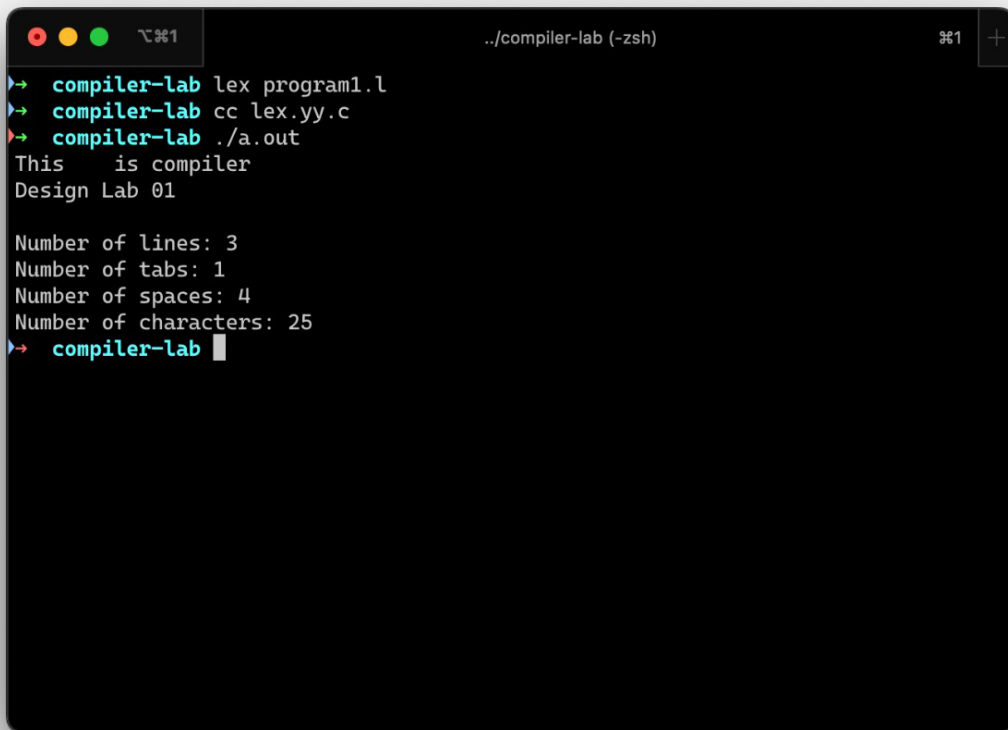


1. Write a lex program to count the number of spaces, lines, tabs and characters.

```
%{
#include<stdio.h>
int nl=0,tc=0,sc=0,ch=0;
%}
%%
\n {nl++;}
\t {tc++;}
[ ] {sc++;}
. {ch++;}
%%
int yywrap() {
    return 1;
}

int main() {
    yylex();
    printf("Number of lines: %d\n",nl);
    printf("Number of tabs: %d\n",tc);
    printf("Number of spaces: %d\n",sc);
    printf("Number of characters: %d\n",ch);
    return 0;
}
```



The image shows a terminal window titled `../compiler-lab (-zsh)`. The user has entered the following commands:

```
→ compiler-lab lex program1.l
→ compiler-lab cc lex.yy.c
→ compiler-lab ./a.out
```

The output of the program is:

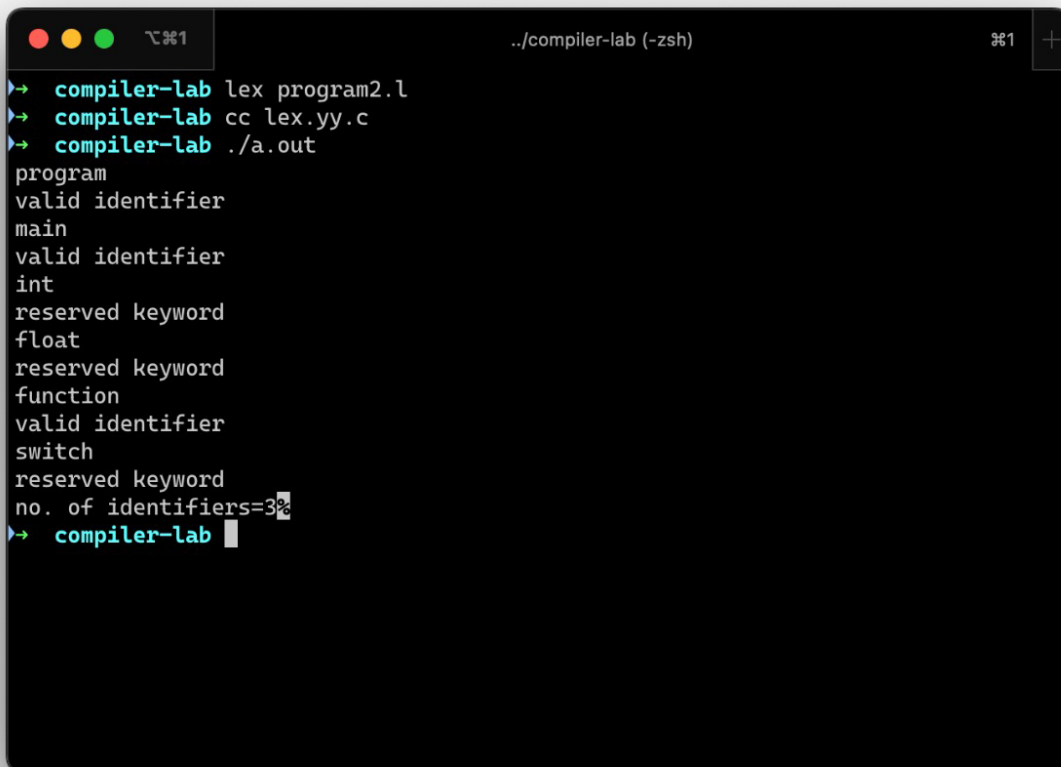
```
This is compiler
Design Lab 01

Number of lines: 3
Number of tabs: 1
Number of spaces: 4
Number of characters: 25
→ compiler-lab
```

2. Write a lex program to count the number of valid identifiers.

```
%{
#include<stdio.h>
int num=0;
%}
%%
auto|else|long|switch|break|enum|register|typedef|case|extern|return|union|char|float|short|
unsigned|const|for|signed|void|continue|goto|sizeof|volatile|default|if|static|while|do|int|
struct|Packed {printf("reserved keyword");}
([a-zA-Z_][a-zA-Z0-9_]*) {num++;printf("valid identifier");}
^([0-9a-zA-Z])* {printf("not a identifier");}
%%
int yywrap() {
    return 1;
}

int main() {
    yylex();
    printf("no. of identifiers=%d",num);
}
```



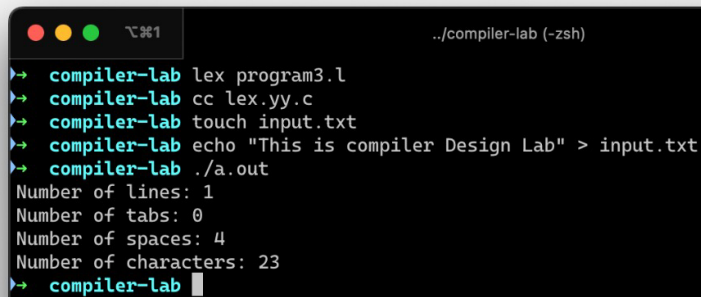
The image shows a terminal window titled `../compiler-lab (-zsh)` with a tab labeled `~%1`. The terminal displays the following commands and output:

```
> compiler-lab lex program2.l
> compiler-lab cc lex.yy.c
> compiler-lab ./a.out
program
valid identifier
main
valid identifier
int
reserved keyword
float
reserved keyword
function
valid identifier
switch
reserved keyword
no. of identifiers=3%
> compiler-lab
```

3. Write a lex program to count the number of spaces, lines, tabs and characters using file handling for input.

```
%{
#include<stdio.h>
int nl=0,tc=0,sc=0,ch=0;
%}
%%
\n {nl++;}
\t {tc++;}
[ ] {sc++;}
. {ch++;}
%%
int yywrap() {
    return 1;
}

int main() {
    extern FILE *yyin;
    yyin=fopen("input.txt","r");
    yylex();
    printf("Number of lines: %d\n",nl);
    printf("Number of tabs: %d\n",tc);
    printf("Number of spaces: %d\n",sc);
    printf("Number of characters: %d\n",ch);
    return 0;
}
```



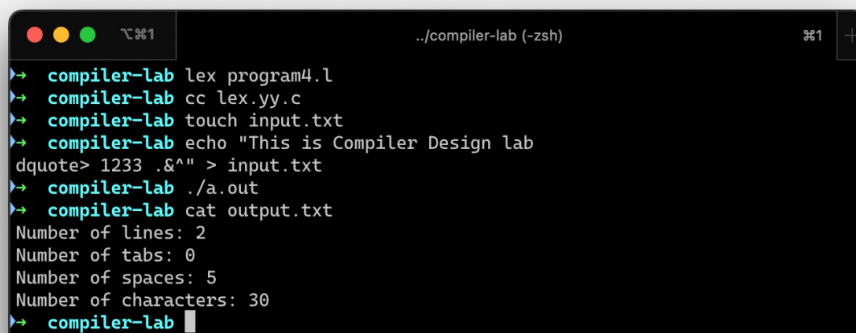
A terminal window titled `../compiler-lab (-zsh)` showing the following commands and output:

```
> compiler-lab lex program3.l
> compiler-lab cc lex.yy.c
> compiler-lab touch input.txt
> compiler-lab echo "This is compiler Design Lab" > input.txt
> compiler-lab ./a.out
Number of lines: 1
Number of tabs: 0
Number of spaces: 4
Number of characters: 23
> compiler-lab
```

4. Write a lex program to count the number of spaces, lines, tabs and characters using file handling for input and output.

```
%{
#include<stdio.h>
int nl=0,tc=0,sc=0,ch=0;
%}
%%
\n {nl++;}
\t {tc++;}
[ ] {sc++;}
. {ch++;}
%%
int yywrap(){
return 1;
}

int main() {
extern FILE *yyin, *yyout;
yyin=fopen("input.txt", "r");
yyout=fopen("output.txt", "w");
yylex();
fprintf(yyout, "Number of lines: %d\n", nl);
fprintf(yyout, "Number of tabs: %d\n", tc);
fprintf(yyout, "Number of spaces: %d\n", sc);
fprintf(yyout, "Number of characters: %d\n", ch);
return 0;
}
```



A terminal window titled `../compiler-lab (-zsh)` showing the execution of a Lex program. The user enters several commands: `lex program4.l`, `cc lex.yy.c`, `touch input.txt`, `echo "This is Compiler Design lab dquote> 1233 .&^" > input.txt`, `./a.out`, and `cat output.txt`. The output of the program is displayed in the terminal, showing the counts for lines, tabs, spaces, and characters.

```
➤ compiler-lab lex program4.l
➤ compiler-lab cc lex.yy.c
➤ compiler-lab touch input.txt
➤ compiler-lab echo "This is Compiler Design lab
dquote> 1233 .&^" > input.txt
➤ compiler-lab ./a.out
➤ compiler-lab cat output.txt
Number of lines: 2
Number of tabs: 0
Number of spaces: 5
Number of characters: 30
➤ compiler-lab
```

5. Write a lex program to count the number of identifiers, separators, keywords, operators, and decimals using file handling.

```
%{
#include<stdio.h>
int nk=0,sp=0,op=0,id=0,integer=0,dec=0;
%}
%%
auto|else|long|switch|break|enum|register|typedef|case|extern|return|union|char|float|short|
unsigned|const|for|signed|void|continue|goto|sizeof|volatile|default|if|static|while|do|int|
struct|Packed {nk++;}
"+"|"*"|"/"|"="|"%"|"("|")"|"{"|"}" {op++;}
"."|";"|","|"-" {sp++;}
([a-zA-Z_][a-zA-Z0-9_]*) {id++;}
[^\.[0-9][^.] {integer++;}
([0-9][.][0-9]) {dec++;}
%%
int yywrap() {
    return 1;
}

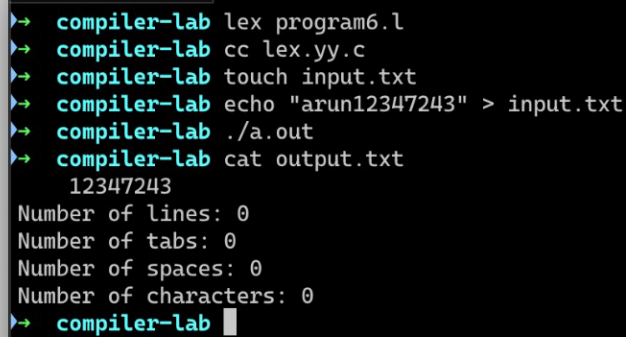
int main()
{
    extern FILE *yyin,*yyout;
    yyin = fopen("input.txt","r");
    yyout = fopen("output.txt","w");
    yylex();
    fprintf(yyout," No. of keyword: %d",nk);
    fprintf(yyout,"\n No. of separator: %d",sp);
    fprintf(yyout,"\n No. of operator: %d",op);
    fprintf(yyout,"\n No. of identifier: %d",id);
    fprintf(yyout,"\n No. of integer: %d",integer);
    fprintf(yyout,"\n No. of decimal: %d",dec);
    return 0;
}
```

```

>→ compiler-lab lex program5.l
>→ compiler-lab cc lex.yy.c
>→ compiler-lab echo "int main ;* . 987 55.09" > input.txt
>→ compiler-lab ./a.out
>→ compiler-lab cat output.txt
    7 No. of keyword: 1
    No. of separator: 3
    No. of operator: 1
    No. of identifier: 1
    No. of integer: 3
    No. of decimal: 0%
>→ compiler-lab
```

6. Write a lex program to replace the occurrence with whitespaces using file handling.

```
%{
#include<stdio.h>
int nl=0,tc=0,sc=0,ch=0;
%}
%%
([a-zA-Z]) {fprintf(yyout," ");}
%%
int yywrap() {
    return 1;
}
int main() {
    extern FILE *yyin, *yyout;
    yyin=fopen("input.txt","r");
    yyout=fopen("output.txt","w");
    yylex();
    fprintf(yyout, "Number of lines: %d\n", nl);
    fprintf(yyout, "Number of tabs: %d\n", tc);
    fprintf(yyout, "Number of spaces: %d\n", sc);
    fprintf(yyout, "Number of characters: %d\n", ch);
    return 0;
}
```



A terminal window titled `../compiler-lab (-zsh)` showing the execution of a Lex program. The user enters the following commands:

- `lex program6.l`
- `cc lex.yy.c`
- `touch input.txt`
- `echo "arun12347243" > input.txt`
- `./a.out`
- `cat output.txt`

The output of the program is displayed as follows:

```
12347243
Number of lines: 0
Number of tabs: 0
Number of spaces: 0
Number of characters: 0
```

The prompt `compiler-lab` is visible at the bottom of the terminal.