**Practical-2**

1. **Write a separate Lex programs for following.**
2. **Write RE that accept zero or one(at most one) occurrence of ‘a’.**

**Code:**

%{

#include<stdio.h>

%}

%%

ba? printf("Valid");

.\* printf("Invalid");

%%

int yywrap(void)

{

return 1;

}

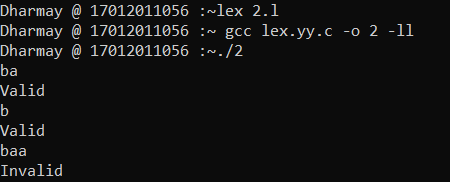
int main(void)

{

yylex();

}

**Output:**



1. **Write RE that accept either ‘a’ or ‘b’.**

**Code:**

%{

#include<stdio.h>

%}

%%

a|b printf("Valid");

.\* printf("Invalid");

%%

int yywrap(void)

{

return 1;

}

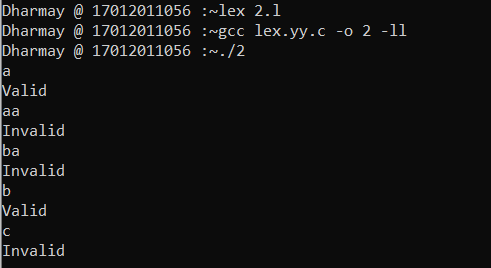
int main(void)

{

yylex();

}

**Output:**



1. **Write RE that accept either ‘a’ or ‘b’ or ‘c’ without using |.**

**Code:**

%{

#include<stdio.h>

%}

%%

[a,b,c] printf("Valid");

.\* printf("Invalid");

%%

int yywrap(void)

{

return 1;

}

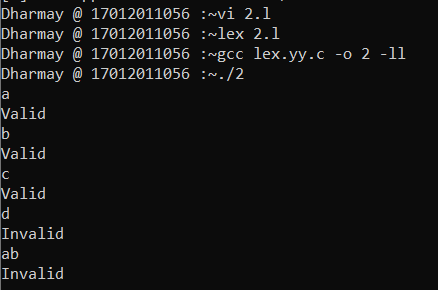
int main(void)

{

yylex();

}

**Output:**



1. **Write RE that accept that accept zero or more occurrences of ‘a’ and single occurrences of ‘b’.**

**Code:**

%{

#include<stdio.h>

%}

%%

a\*ba\* printf("Valid");

.\* printf("Invalid");

%%

int yywrap(void)

{

return 1;

}

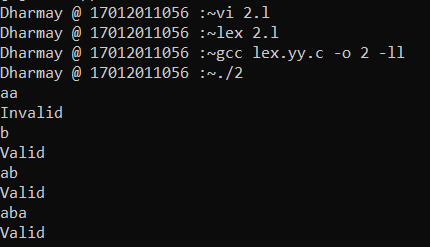
int main(void)

{

yylex();

}

**Output:**



1. **Write RE that accepts all the strings which ends with ‘b’.**

**Code:**

%{

#include<stdio.h>

%}

%%

.\*b printf("Valid");

.\* printf("Invalid");

%%

int yywrap(void)

{

return 1;

}

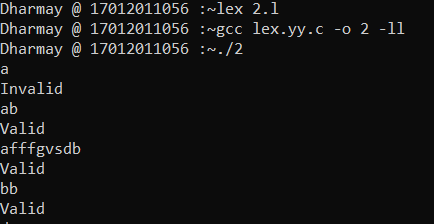
int main(void)

{

yylex();

}

**Output:**



1. **Write RE for new line.**

**Code:**

%{

#include<stdio.h>

%}

%%

[\n] printf("Valid\n");

.\* printf("Invalid\n");

%%

int yywrap(void)

{

return 1;

}

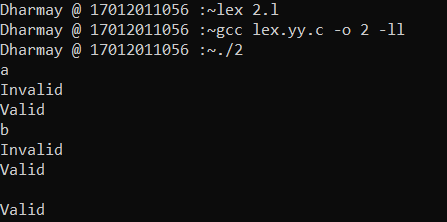
int main(void)

{

yylex();

}

**Output:**



1. **Write RE that accepts ‘\n’.**

**Code:**

%{

#include<stdio.h>

%}

%%

\\n printf("Valid");

.\* printf("Invalid");

%%

int yywrap(void)

{

return 1;

}

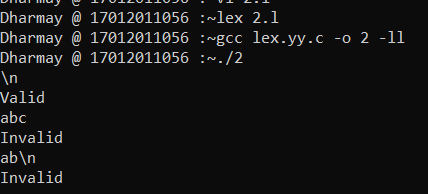
int main(void)

{

yylex();

}

**Output:**



1. **Write a string that accepted by the given lex program and justify the output.**

|  |  |
| --- | --- |
| **1.**  **%{**  **#include<stdio.h>**  **%}**  **%%**  **[\n] printf("valid");**  **.\* printf("invalid");**  **%%**  **int yywrap()**  **{**  **return 1;**  **}**  **int main()**  **{**  **yylex();**  **return 0;**  **}** | **2.**  **%{**  **#include<stdio.h>**  **%}**  **%%**  **[\\n] printf("valid");**  **.\* printf("invalid");**  **%%**  **int yywrap(void)**  **{**  **return 1;**  **}**  **int main()**  **{**  **yylex();**  **return 0;**  **}** |

**Code1:**

%{

#include<stdio.h>

%}

%%

[\n] printf("valid");

.\* printf("invalid");

%%

int yywrap()

{

return 1;

}

int main()

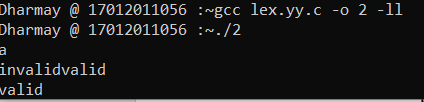
{

yylex();

return 0;

}

**Output1:**



**Code2:**

%{

#include<stdio.h>

%}

%%

[\\n] printf("valid");

.\* printf("invalid");

%%

int yywrap(void)

{

return 1;

}

int main()

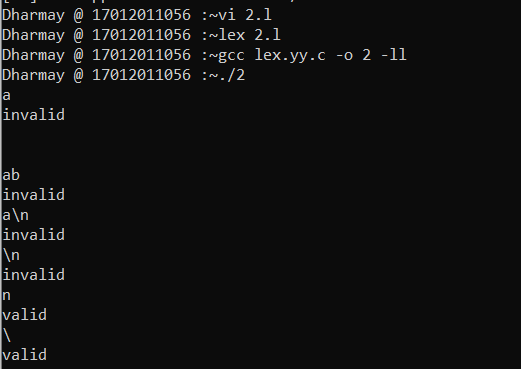
{

yylex();

return 0;

}

**Output2:**



1. **Write a RE that accepts any character except ‘\’ and ‘n’.**

**Code:**

%{

#include<stdio.h>

%}

%%

[\\n] printf("invalid");

.\* printf("valid");

%%

int yywrap(void)

{

return 1;

}

int main()

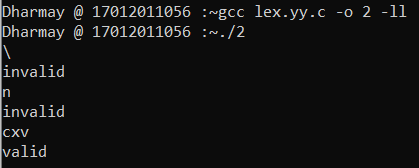
{

yylex();

return 0;

}

**Output:**



1. **Write all the strings which are accepted by [a|b|c\*].**

**Code:**

%{

#include<stdio.h>

%}

%%

[a|b|c\*] printf("valid");

.\* printf("Invalid");

%%

int yywrap(void)

{

return 1;

}

int main()

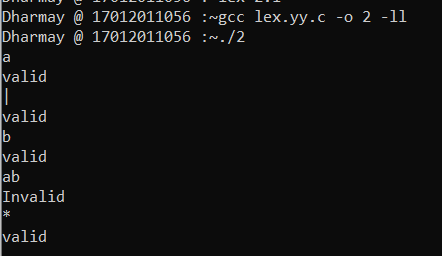
{

yylex();

return 0;

}

**Output:**



1. **Write a RE that accept any character except ‘a’ and ‘b’.**

**Code:**

%{

#include<stdio.h>

%}

%%

a|b printf("Invalid");

.\* printf("valid");

%%

int yywrap(void)

{

return 1;

}

int main()

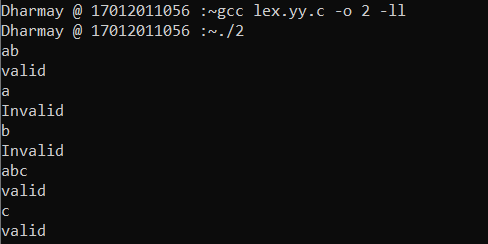
{

yylex();

return 0;

}

**Output:**



1. **Write more than one RE that accepts string ‘abc’.**

**Code(1):**

%{

#include<stdio.h>

%}

%%

abc printf("valid");

.\* printf("Invalid");

%%

int yywrap(void)

{

return 1;

}

int main()

{

yylex();

return 0;

}

**Code(2):**

%{

#include<stdio.h>

%}

%%

abc printf("valid");

.\* printf("Invalid");

%%

int yywrap(void)

{

return 1;

}

int main()

{

yylex();

return 0;

}

**Code(3):**

%{

#include<stdio.h>

%}

%%

(abc) printf("valid");

.\* printf("Invalid");

%%

int yywrap(void)

{

return 1;

}

int main()

{

yylex();

return 0;

}

**Code(4):**

%{

#include<stdio.h>

%}

%%

[a][b][c] printf("valid");

.\* printf("Invalid");

%%

int yywrap(void)

{

return 1;

}

int main()

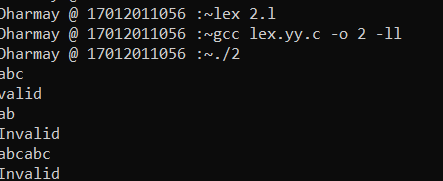
{

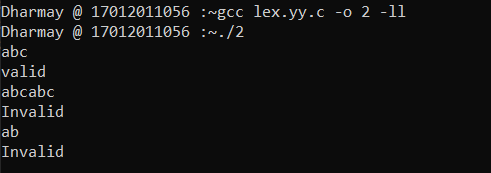
yylex();

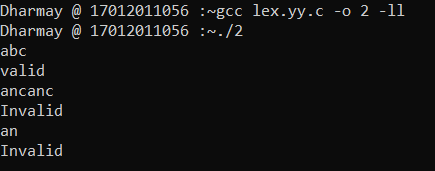
return 0;

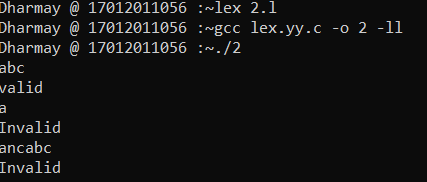
}

**Output:**









1. **Is there any difference between ‘abc’ and “abc”? Justify your answer.**

**Code(1):**

%{

#include<stdio.h>

%}

%%

'abc' printf("valid");

.\* printf("Invalid");

%%

int yywrap(void)

{

return 1;

}

int main()

{

yylex();

return 0;

}

**Code(2):**

%{

#include<stdio.h>

%}

%%

"abc" printf("valid");

.\* printf("Invalid");

%%

int yywrap(void)

{

return 1;

}

int main()

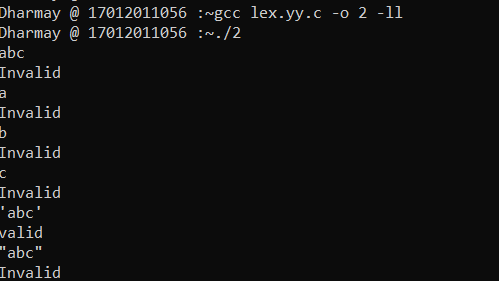
{

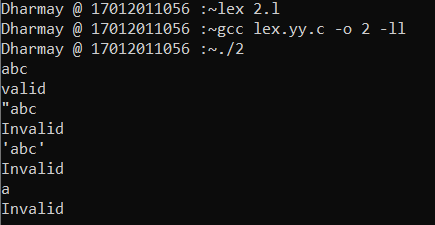
yylex();

return 0;

}

**Output:**





1. **Which are the strings accepted by (“abc”)\*.**

**Code:**

**%{**

#include<stdio.h>

%}

%%

("abc")\* printf("valid");

.\* printf("Invalid");

%%

int yywrap(void)

{

return 1;

}

int main()

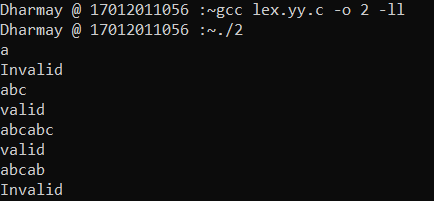
{

yylex();

return 0;

}

**Output:**



1. **Write the RE that accepts zero or more occurrences of digit and capital letters.**

**Code:**

%{

#include<stdio.h>

%}

%%

[A-Z0-9]\* printf("valid");

.\* printf("Invalid");

%%

int yywrap(void)

{

return 1;

}

int main()

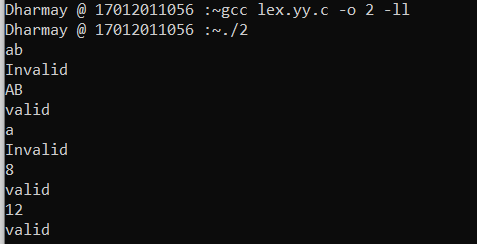
{

yylex();

return 0;

}

**Output:**



1. **Write valid and invalid strings accepted by following regular expressions.**
2. **(a-b)?[0-9 A-Z]\***

%{

#include <stdio.h>

%}

%%

(a-b)?[A-Z0-9]\* printf( "valid");

.\* printf("invalid");

%%

int yywrap(void)

{

return 1;

}

int main()

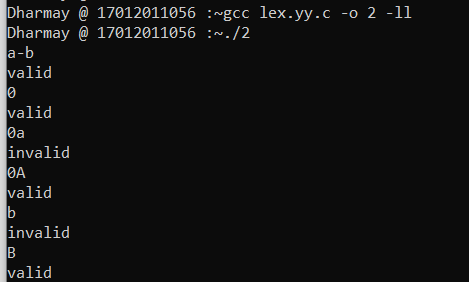
{

yylex();

return 0;

}

**Output:**



1. **[^ab][0-9]\***

**Code:**

%{

#include <stdio.h>

%}

%%

[^ab][0-9]\* printf( "valid\n");

.\* printf("invalid\n");

%%

int yywrap(void)

{

return 1;

}

int main()

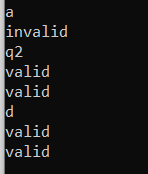
{

yylex();

return 0;

}

**Output:**



1. **^[ab][0-9 A-Z]+**

**Code:**

%{

#include <stdio.h>

%}

%%

^[ab][0-9A-Z]+ printf( "valid");

.\* printf("invalid");

%%

int yywrap(void)

{

return 1;

}

int main()

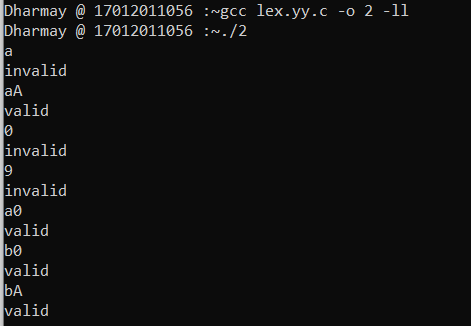
{

yylex();

return 0;

}

**Output:**



1. **[0-9][A-Z]$**

**Code:**

%{

#include <stdio.h>

%}

%%

[0-9][A-Z]$ printf( "valid");

.\* printf("invalid");

%%

int yywrap(void)

{

return 1;

}

int main()

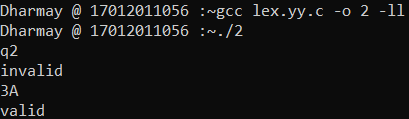
{

yylex();

return 0;

}

**Output:**



1. **[A-Z a-z]{6}**

**Code:**

%{

#include <stdio.h>

%}

%%

[A-Za-z]{6} {printf( "valid");}

.\* {printf("invalid");}

%%

int yywrap(void)

{

return 1;

}

int main()

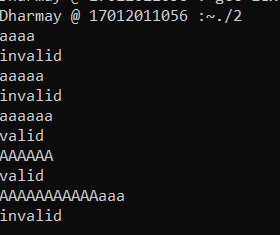
{

yylex();

return 0;

}

**Output:**



1. **[a+b]{6}**

**Code:**

%{

#include <stdio.h>

%}

%%

[a+b]{6} printf( "valid");

.\* printf("invalid");

%%

int yywrap(void)

{

return 1;

}

int main()

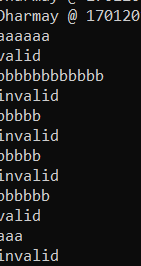
{

yylex();

return 0;

}

**Output:**



1. **[a+b]**

**Code:**

%{

#include <stdio.h>

%}

%%

[a+b] printf( "valid");

.\* printf("invalid");

%%

int yywrap()

{

return 1;

}

int main()

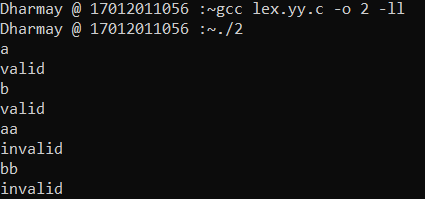
{

yylex();

return 0;

}

**Output:**



1. **Demonstrate the use of lex predefined variables (yytext, yyleng, yyin) with the help of program.**

**Code:**

%{

#include<stdio.h>

%}

%%

[a-zA-Z] {printf("%s is character",yytext);}

[a-zA-Z]\* {printf("%s is string",yytext);}

.\* {printf("%s special symbols",yytext);}

%%

int yywrap()

{

return 1;

}

int main()

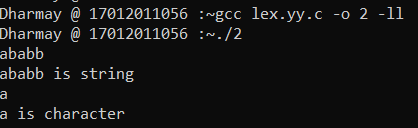
{

yylex();

return 0;

}

**Output:**



1. **Write a lex program to recognize character, string and special symbols from given input.**

**Code:**

%{

#include <stdio.h>

%}

%%

[a-zA-Z0-9] printf( "%s is Character.",yytext);

. printf("%s is Symbol.",yytext);

.\* printf("%s is String.",yytext);

%%

int yywrap()

{

return 1;

}

int main()

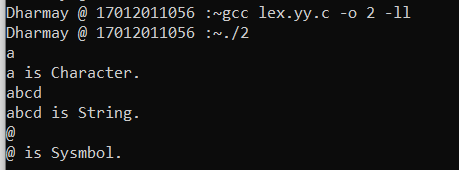
{

yylex();

return 0;

}

**Output:**



1. **Write a lex program to validate mobile number. (i.e Number having length of 10 is valid)**

**Code:**

%{

#include <stdio.h>

%}

%%

[0-9]{10} printf("Valid");

.\* printf("Invalid");

%%

int yywrap(void)

{

return 1;

}

int main()

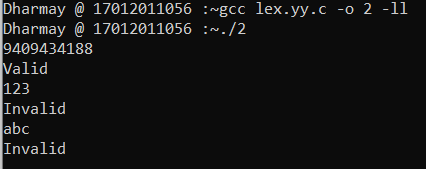
{

yylex();

return 0;

}

**Output:**



1. **Write a lex program to differentiate mobile number and land line number.**

**Code:**

%{

#include <stdio.h>

%}

%%

[0-6]+[7-9]+[0-9]{8} printf("It is a Landline Number");

[6-9][0-9]{9} printf("It is a Mobile Number");

.\* printf("Invalid input.!!");

%%

int yywrap(void)

{

return 1;

}

int main()

{

yylex();

return 0;

}

**Output:**

