|  |
| --- |
| Counting the number of capital characters |
| **Code-**  /\*\*\* Definition Section has one variable  which can be accessed inside yylex()  and main() \*\*\*/  %{  int count = 0;  %}    /\*\*\* Rule Section has three rules, first rule  matches with capital letters, second rule  matches with any character except newline and  third rule does not take input after the enter\*\*\*/  %%  [A-Z] {printf("%s capital letter\n", yytext);  count++;}  . {printf("%s not a capital letter\n", yytext);}  \n {return 0;}  %%    /\*\*\* Code Section prints the number of  capital letter present in the given input\*\*\*/  int yywrap(){}  int main(){  // Explanation:  // yywrap() - wraps the above rule section  /\* yyin - takes the file pointer  which contains the input\*/  /\* yylex() - this is the main flex function  which runs the Rule Section\*/  // yytext is the text in the buffer    // Uncomment the lines below  // to take input from file  // FILE \*fp;  // char filename[50];  // printf("Enter the filename: \n");  // scanf("%s",filename);  // fp = fopen(filename,"r");  // yyin = fp;    yylex();  printf("\nNumber of Captial letters "  "in the given input - %d\n", count);    return 0;  } |
| **Output Screenshot-** |

|  |
| --- |
| Counting the number of lines and characters |
| **Code-**  /\* Decalring two counters one for number  of lines other for number of characters \*/  %{  int no\_of\_lines = 0;  int no\_of\_chars = 0;  %}    /\*\*\*rule 1 counts the number of lines,  rule 2 counts the number of characters  and rule 3 specifies when to stop  taking input\*\*\*/  %%  \n ++no\_of\_lines;  . ++no\_of\_chars;  end return 0;  %%    /\*\*\* User code section\*\*\*/  int yywrap(){}  int main(int argc, char \*\*argv)  {    yylex();  printf("number of lines = %d, number of chars = %d\n",  no\_of\_lines, no\_of\_chars );    return 0;  } |
| **Output Screenshot-** |

|  |
| --- |
| Is the given number positive or negative |
| **Code-**  %%  [+]?[0-9]+ {printf("positive integer\n");}  [-]?[0-9]+ {printf("negative integer\n");}  .  %%  int yywrap()  {  return 1;  }  int main()  {  printf("positive and negative integer recognition\n");  yylex();    return 0;  } |
| **Output Screenshot -** |

|  |
| --- |
| Identifying tokens |
| **Code-**  %{  int n = 0 ;  %}  %%  "while"|"if"|"else" {n++;printf("\t keywords : %s", yytext);}  "int"|"float" {n++;printf("\t keywords : %s", yytext);}  "<="|"=="|"="|"++"|"-"|"\*"|"+" {n++;printf("\t operator : %s", yytext);}  [a-zA-Z\_][a-zA-Z0-9\_]\* {n++;printf("\t identifier : %s", yytext);}  [(){}|, ;] {n++;printf("\t separator : %s", yytext);}  [0-9]\*"."[0-9]+ {n++;printf("\t float : %s", yytext);}  [0-9]+ {n++;printf("\t integer : %s", yytext);}  . ;  %%  int yywrap()  {  return 1;  }  int main()  {  yylex();  printf("\n total no. of token = %d\n", n);  return 0;  } |
| **Output Screenshot -** |

|  |
| --- |
| Identifying characters |
| **Code-**  %{  #include <math.h>  /\*Inclusive start condition\*/  #undef yywrap  #define yywrap() 1  %}  %s expect  %%  expect-floats BEGIN(expect);  <expect>[0-9]+.[0-9]+ {  printf( "found a float, = %f\n",  atof( yytext ) );  }  <expect>\n {  /\* that's the end of the line, so  \* we need another "expect-number"  \* before we'll recognize any more  \* numbers  \*/  BEGIN(INITIAL);  }  [0-9]+ {  printf( "found an integer, = %d\n",  atoi( yytext ) );  }  "." printf( "found a dot\n" );    %%  int main()  {  yylex();  } |
| **Output Screenshot -** |

|  |
| --- |
| Showcasing ECHO & REJECT |
| **Code-**  %{      /\*USE OF REJECT STATEMENT\*/  #undef yywrap  #define yywrap() 1    %}      %%      [a-z]+ {  printf("\ncontains only lowercase letters = ");  ECHO;    }    [a-zA-Z]+ {  printf("\ncontains both uppercase and lowercase letters = ");  ECHO;  REJECT;    }    . {  printf("\ncontains mixed letters = ");  ECHO;    }    %%        int main()  {  yylex();  } |
| **Output Screenshot –** |

|  |
| --- |
| Inclusives |
| **Code-**  %{    /\*Inclusive start condition\*/  #undef yywrap  #define yywrap() 1    %}      %s SM SMBG      %%      # BEGIN(SM);  ## BEGIN(SMBG);    [0-9]+ {  printf("Contains only digits");  }    <SMBG>[A-Z]+ {  printf("Contains uppercase letters");  }    <SM>. {  printf("Exiting from # start condition");  BEGIN(INITIAL);  }    <SM,SMBG>[a-z]+ {  printf("Contains lowercase letters");    }    <SMBG>.+ {  printf("Exiting from ## start condition");  BEGIN(INITIAL);  }    .+ {  printf("No action exexuted");    }    %%      int main()  {  printf("Enter # when expecting digits or lowercase letters");  printf(" Enter ## when expecting only lowercase and uppercase letters");  yylex();  } |
| **Output Screenshot –** |