

COMPILER CONSTRUCTION

GROUP ASSIGNMENT 1



GROUP MEMBERS

<u>NAME</u>	<u>REGISTRATION NUMBER</u>
YVONNE KIMANI	SCT212-0475/2017
MARK MUNENE	SCT 212-0224-2017
GABRIEL WAINAINA MWANGI	SCT212-0480/2017
IAN MWANGI	SCT212-0066/2017
STANLEY NGUGI	SCT212-0065/2017
DENNIS GACHOMO	SCT212-9218/2015

1) Write a LEX specification files to:

a) Count the number of words in a file and their total size

Solution

```
%{
    #include <stdio.h>
    int words=0, c_letters=0, total=0;
}%

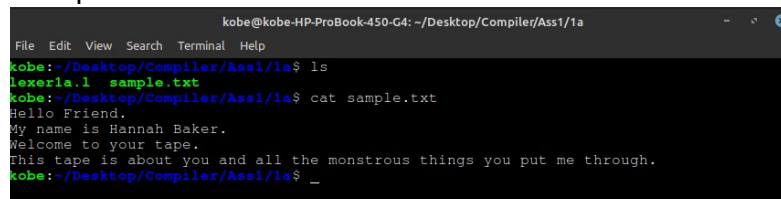
%%

\n {words++;}
[\t ' '] words++;
[a-zA-Z] c_letters++;
%%

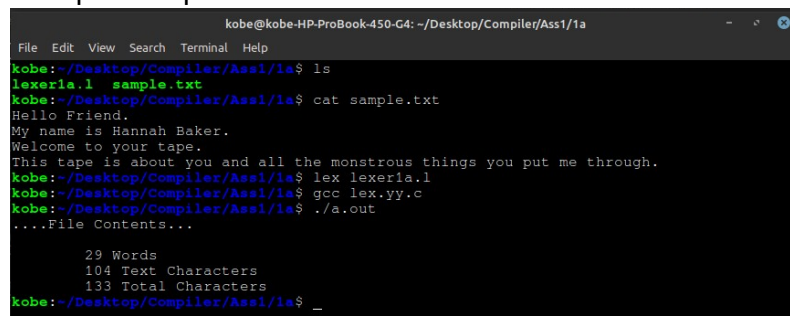
int main(){
    yyin=fopen("sample.txt","r");
    yylex();
    total = c_letters+words;
    printf("File Contents...\n");
    printf("\n\t%d Words",words);
    printf("\n\t%d Text Characters",c_letters);
    printf("\n\t%d Total Characters\n",total);
    return 0;
}
int yywrap(){
    return 1;
}
```

Result

Sample Text File Contents:

A terminal window with a dark background and light green text. The prompt is 'kobe@kobe-HP-ProBook-450-G4: ~/Desktop/Compiler/Ass1/1a'. The user enters 'ls', showing 'lexeria.l' and 'sample.txt'. Then they enter 'cat sample.txt', displaying the contents of the file: 'Hello Friend.', 'My name is Hannah Baker.', 'Welcome to your tape.', and 'This tape is about you and all the monstrous things you put me through.'

Sample Output

A terminal window with a dark background and light green text. The prompt is 'kobe@kobe-HP-ProBook-450-G4: ~/Desktop/Compiler/Ass1/1a'. The user enters 'ls', showing 'lexeria.l' and 'sample.txt'. Then they enter 'cat sample.txt', displaying the contents of the file. After that, they enter 'lex lexeria.l', 'gcc lex.yy.c', and './a.out'. The output of the program is displayed: '....File Contents...', '29 Words', '104 Text Characters', and '133 Total Characters'.

b) Counts the number of different words in an input

Solution

```
%{
int words = 0; // will be counter for our different words

char *cache = NULL; // the unique words will be put in the 'cache' variable
%}

%%
[a-zA-Z0-9]+ {
    int ret_code = 0; //retrival code: code to check if input matches
    regular expression
    char *target = NULL;
    char *found = NULL;
    // add a 'space' before and after the word for words matching and
    word boundaries.
    ret_code = asprintf(&target, " %s ", yytext);
    if (ret_code < 0) { //returns No<0 if input does not match reg exp
        exit(1);
    }

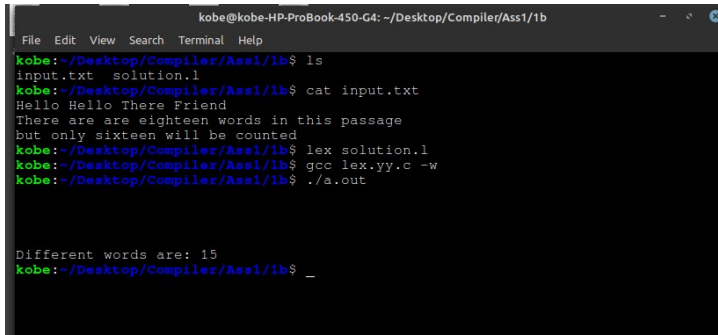
    // check for NULL in cache var
    if (NULL != cache) {
        found = strstr(cache, target); //checks for first occurence of a word
    }
    // if the word has NOT been found
    if (NULL == found) {
        words++;
        // store this new different word into the cache
        ret_code = asprintf(&cache, "%s%s", cache, target);
        if (ret_code < 0) {
            exit(1);
        }
    }
}

. //ignore other words not defined above
%%

int main(int argc, char **argv) {
    yyin = fopen("input.txt", "r"); //open and read input file
    yylex(); //opening and reading input stream
    printf("\nDifferent words are: %d\n", words);
    return 0;
}
```

```
}
int yywrap() { return(1); } //shows end of input
```

Result



```
kobe@kobe-HP-ProBook-450-G4: ~/Desktop/Compiler/Ass1/1b
File Edit View Search Terminal Help
kobe:~/Desktop/Compiler/Ass1/1b$ ls
input.txt  solution.l
kobe:~/Desktop/Compiler/Ass1/1b$ cat input.txt
Hello Hello There Friend
There are are eighteen words in this passage
but only sixteen will be counted
kobe:~/Desktop/Compiler/Ass1/1b$ lex solution.l
kobe:~/Desktop/Compiler/Ass1/1b$ gcc lex.yy.c -w
kobe:~/Desktop/Compiler/Ass1/1b$ ./a.out

Different words are: 15
kobe:~/Desktop/Compiler/Ass1/1b$ _
```

- c) **Accepts the English language words (without bothering for the meaning) and replaces each occurrence of the string “abc” in it to “ABC”.**

Solution

```
%{
#include<stdio.h>
#include<string.h>
int i;
}%

%%

[a-zA-Z]*    {
                for(i=0;i<=yyleng;i++)
                {
                    if((yytext[i]=='a')&&(yytext[i+1]=='b')&&(yytext[i+2]=='c'))
                    {
                        yytext[i]='A';
                        yytext[i+1]='B';
                        yytext[i+2]='C';
                    }
                }
                printf("%s",yytext);
            }

.*           {ECHO;}
\n           {printf("%s",yytext);}

%%
```

Result

```
kobe@kobe-HP-ProBook-450-G4: ~/Desktop/Compiler/Ass1/1c
File Edit View Search Terminal Help
kobe:~/Desktop/Compiler/Ass1/1c$ ls
lexer1c.1
kobe:~/Desktop/Compiler/Ass1/1c$ lex lexer1c.1
kobe:~/Desktop/Compiler/Ass1/1c$ gcc lex.yy.c
kobe:~/Desktop/Compiler/Ass1/1c$ ./a.out
lexer1c.1
abc
ABC
abcbcabcb
ABCBABCB
abcbcabcbabcb
ABCBABCBABCB
--
```

2) The following is a listing of a set of verbs:

is am are were do does did will has have had go
was be being been would should can
could

Write a simple LEX specification to recognize these verbs

Solution

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

%%
```

```
[\t ' ']+ /* ignore whitespace */ ;
```

is |
am |
are |
were |
was |

```

be |
being |
been |
do |
does |
did |
will |
would |
should |
can |
could |
has |
have |
had |
go      { printf("%s: is a verb\n", yytext); }
[a-zA-Z]+ { printf("%s: is not a verb\n", yytext); }

.|\\n    { ECHO; /* normal default anyway */ }
%%

```

```

int main()
{
    yylex() ;
    return 0;
}

int yywrap(){
    return 1;
}

```

Result

```

kobe:~/Desktop/Compiler/Asst1/2$ ls
lexer2a.l
kobe:~/Desktop/Compiler/Asst1/2$ lex lexer2a.l
kobe:~/Desktop/Compiler/Asst1/2$ gcc lex.yy.c
kobe:~/Desktop/Compiler/Asst1/2$ ls
a.out lexer2a.l lex.yy.c
kobe:~/Desktop/Compiler/Asst1/2$ ./a.out
happy
happy: is not a verb

have
have: is a verb

I should be reading but I have to go to the beach.
I: is not a verb
should: is a verb
be: is a verb
reading: is not a verb
but: is not a verb
I: is not a verb
have: is a verb
to: is not a verb
go: is a verb
to: is not a verb
the: is not a verb
beach: is not a verb
.I: is not a verb

```