SS LAB 6

SAHIL BONDRE: U18CO021

1. Write a Lex program to count the number of lines, characters and words of the given input file.

2. Design a scanner to

   (a) Count number of single and multiple line comments from a C program available in *xyz.txt* file. [Note: You can create any txt file having sample C code which contains single and multiple line comments]

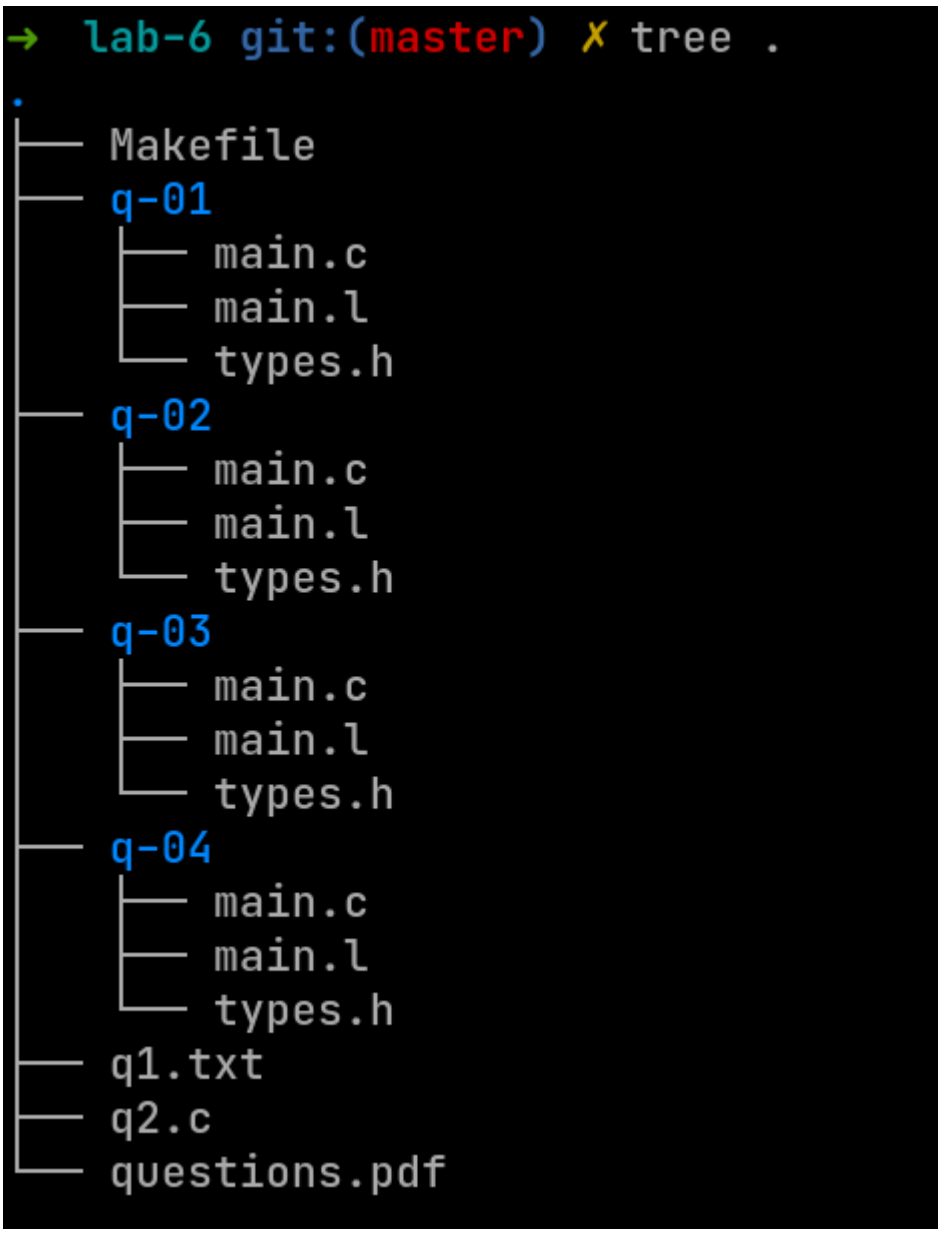
   (b) Remove comment lines from the C program.

3. Write a Lex program to check valid/invalid

   (a) Mobile number (considering 10-digit mobile number followed by country code +91)

   (b) Email address

4. Design a scanner to check whether a number is Armstrong number or not.



Makefile

```
default: $(TARGET)

q1:
	lex -o q-01/lex.yy.c q-01/main.l
	gcc q-01/main.c q-01/lex.yy.c

q2:
	lex -o q-02/lex.yy.c q-02/main.l
	gcc q-02/main.c q-02/lex.yy.c

q3:
	lex -o q-03/lex.yy.c q-03/main.l
```

```
	gcc q-03/main.c q-03/lex.yy.c

q4:
	lex -o q-04/lex.yy.c q-04/main.l
	gcc q-04/main.c q-04/lex.yy.c -lm


.PHONY: clean
clean:
	find . -name "*.out" -delete
	find . -name "*.yy.c" -delete
	find . -name "clean.c" -delete
```

Q1:

types.h

```
#define CARET_RETURN 1
#define NON_WHITE_SPACE 2
#define WHITE_SPACE 3
```

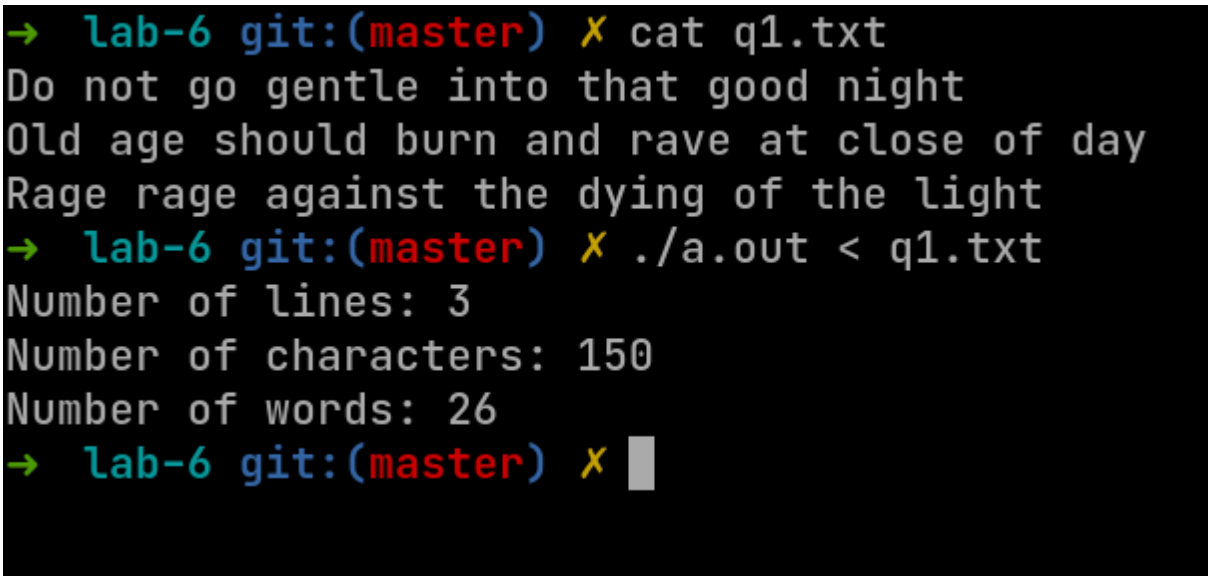
main.l

```
%{
  #include "types.h"
%}

%%
\n  return CARET_RETURN;
[^ \n\t]+ return NON_WHITE_SPACE;
. return WHITE_SPACE;
%%

int yywrap(void) {return 1;}
```

main.c

```
#include <stdio.h>

#include "types.h"

extern int yylex();
extern int yylineno;
extern char* yytext;
```

```
extern int yyleng;

int main(int argc, char const* argv[]) {
  int token = yylex();
  int lines = 0, chars = 0, words = 0;

  while (token) {
    switch (token)
    {
    case CARET_RETURN:
      ++lines;
      ++chars;
      break;
    case NON_WHITE_SPACE:
      chars += yyleng;
      ++words;
    case WHITE_SPACE:
      ++chars;
    default:
      break;
    }
    token = yylex();
  }

  printf("Number of lines: %d\n", lines);
  printf("Number of characters: %d\n", chars);
  printf("Number of words: %d\n", words);
  return 0;
}
```



Q2:

types.h

```
#define SINGLE_LINE_COMMENT 1
#define MULTI_LINE_COMMENT 2
#define NOT_COMMENT 3
#define CR 4
```

main.l

```
%{
  #include "types.h"

%}

%%
\/\/(.*) return SINGLE_LINE_COMMENT;
"/*"([^*]|\*+[^*/])*\*+"/" return MULTI_LINE_COMMENT;
\n return CR;
. return NOT_COMMENT;
%%

int yywrap(void) {return 1;}
```

main.c

```
%{
  #include "types.h"

%}

%%
\/\/(.*) return SINGLE_LINE_COMMENT;
"/*"([^*]|\*+[^*/])*\*+"/" return MULTI_LINE_COMMENT;
\n return CR;
. return NOT_COMMENT;
%%

int yywrap(void) {return 1;}include <stdio.h>
#include <stdlib.h>

#include "types.h"

extern int yylex();
extern int yylineno;
```

```c
extern char* yytext;
extern int yyleng;

int main(int argc, char const* argv[]) {
  int token = yylex();
  int single = 0, multi = 0;
  FILE* fp = fopen("clean.c", "w");

  while (token) {
    switch (token) {
      case SINGLE_LINE_COMMENT:
        ++single;
        break;
      case MULTI_LINE_COMMENT:
        ++multi;
        break;
      case CR:
        fprintf(fp, "\n");
      case NOT_COMMENT:
        fprintf(fp, "%s", yytext);
      default:
        break;
    }
    token = yylex();
  }
  fclose(fp);
  printf("Number of single line comments: %d\n", single);
  printf("Number of single multi line comments: %d\n", multi);

  return 0;
}
```

```
➜  lab-6 git:(master) ✗ cat q2.c
#include <stdio.h>

int main(int argc, char const *argv[]) {
  // Comment
  // Comment
  // Comment
  // Comment
  /*This
    is a long
  comment

  */
  printf("Hello World!\n");
  return 0;
}
  /*This
    is a long
  comment

  */%
➜  lab-6 git:(master) ✗ ./a.out < q2.c
Number of single line comments: 4
Number of single multi line comments: 2
```

```
➜  lab-6 git:(master) ✗ cat clean.c
#include <stdio.h>

int main(int argc, char const *argv[]) {
  printf("Hello World!\n");

  return 0;
}
➜  lab-6 git:(master) ✗
```

Q3:

types.h

```
#define PHONE 1
#define EMAIL 2
#define NONE 3
```

main.l

```
%{
  #include "types.h"
%}

%%
[1-9][0-9]{9} return PHONE;
[a-z . 0-9]+@[a-z]+[.][a-z]{2,4} return EMAIL;
.+ return NONE;
%%

int yywrap(void) {return 1;}
```

main.c

```
#include <stdio.h>
#include <stdlib.h>

#include "types.h"

extern int yylex();
extern int yylineno;
extern char* yytext;
extern int yyleng;

int main(int argc, char const* argv[]) {
  int token = yylex();

  while (token) {
    switch (token) {
      case PHONE:
      printf("Valid Phone\n");
        break;
      case EMAIL:
      printf("Valid Email\n");
        break;
      case NONE:
      printf("Invalid\n");
      default:
```

```
        break;
    }
    token = yylex();
  }

  return 0;
}
```

Q4:

types.h

```
#define NUMBER 1
#define NOT_A_NUM 2
```

main.l

```
%{
  #include "types.h"
%}

%%
```

```
[0-9]* return NUMBER;
.+ return NOT_A_NUM;
%%

int yywrap(void) {return 1;}
```

main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#include "types.h"

extern int yylex();
extern int yylineno;
extern char* yytext;
extern int yyleng;

int armstrong(char* a) {
  int len = strlen(a), i, num = 0;
  for (i = 0; i < len; i++) num = num * 10 + (a[i] - '0');

  int x = 0, y = 0, temp = num;
  while (num > 0) {
    y = pow((num % 10), len);
    x = x + y;
    num = num / 10;
  }

  return x == temp;
}

int main(int argc, char const* argv[]) {
  int token = yylex();

  while (token) {
    switch (token) {
      case NUMBER:
        if (armstrong(yytext))
          printf("Armstrong\n");
        else
          printf("Not Armstrong\n");
        break;
      case NOT_A_NUM:
```

```
        printf("Not a Number\n");
        break;
      default:
        break;
    }
    token = yylex();
  }

  return 0;
}
```

```
➜  lab-6 git:(master) ✗ ./a.out
sahil
Not a Number

123
Not Armstrong

351
Not Armstrong

371
Armstrong

153
Armstrong

22
Not Armstrong

^C
```