**COMPILER DESIGN (CST -309)**

**LEXICAL ANALYSER**

Submitted to – Dr. Dinesh Goplani

Submitted by

* Ashutosh Soni (2018ucp1505)
* Shubham Yadav (2018ucp1421)

**Aim:** Implement Lexical Analyser using LEX program and analyse the output.

**Introduction:**

Lexical analyser is also known as ‘Scanner’. Its main job is to convert given input source file into stream of tokens.

Programmatically, lex is a tool for automatically generating a scanner starting from lex specification.

Lexical Program Consists of three parts

1. Declarations
2. Transition Rules
3. Auxiliary procedures

Below is the code for its implementation

**Implementation of lexical Analyser using LEX program.**

**/\* Program for Lexical Analyser \*/**

**/\* Declaration Part \*/**

**%{**

**int comment = 0;**

**int count\_of\_comment = 0;**

**%}**

**identifier [a-zA-Z\_][a-zA-Z0-9]\***

**/\* Transition Rules \*/**

**%%**

**#.\* { printf("\n%s is a PREPROCESSOR DIRECTIVE \n",yytext); }**

**auto |**

**break |**

**case |**

**char |**

**continue |**

**do |**

**default |**

**const |**

**double |**

**else |**

**enum |**

**extern |**

**for |**

**if |**

**goto |**

**float |**

**int |**

**long |**

**register |**

**return |**

**signed |**

**static |**

**sizeof |**

**short |**

**struct |**

**typedef |**

**union |**

**void |**

**while |**

**volatile |**

**unsigned { printf(" %s is a KEYWORD \n",yytext); }**

**"/\*" { comment = 1; }**

**"\*/" { comment = 0;**

**count\_of\_comment++; }**

**{identifier}\( {**

**if(!comment)**

**printf("\nFUNCTION %s \n",yytext);**

**}**

**\{ {**

**if(!comment)**

**printf("\t BLOCK BEGINS \n");**

**}**

**\} {**

**if(!comment)**

**printf("\t BLOCK ENDS \n");**

**}**

**{identifier}(\[[0-9]\*\])? {**

**if(!comment)**

**printf("\t %s is a IDENTIFIER \n",yytext);**

**}**

**\".\*\" {**

**if(!comment)**

**printf("\t %s is a STRING \n",yytext);**

**}**

**[0-9]+ {**

**if(!comment)**

**printf("\t %s is a NUMBER \n",yytext);**

**}**

**\)(\;)? {**

**if(!comment)**

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

**ECHO;**

**}**

**\(**

**ECHO;**

**= {**

**if(!comment)**

**printf("\t %s is a ASSIGNMENT OPERATOR \n",yytext);**

**}**

**\ <= |**

**\ >= |**

**\ < |**

**\ == |**

**\ > {**

**if(!comment)**

**printf("\t %s is a RELATIONAL OPERATOR \n",yytext);**

**}**

**%%**

**int main(int argc,char \*argv[]){**

**if(argc!=2){**

**printf("Please give input file \n");**

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

**exit(0);**

**}**

**FILE \*file = NULL ;**

**file = fopen(argv[1],"r");**

**if(file == NULL){**

**printf("Error in opening file \n");**

**printf("Try again ... \n terminating ....\n");**

**exit(0);**

**}**

**yyin = file;**

**yylex();**

**printf(" \n Total number of comments in this file is %d \n",count\_of\_comment);**

**return 0;**

**}**

**int yywrap()**

**{**

**return 1;**

**}**

**Procedure to run Lex Program:**

Step 1: Install Flex in Ubuntu

sudo apt-get update

sudo apt-get install flex

Step 2: Run command on lexical complier

lex filename.l

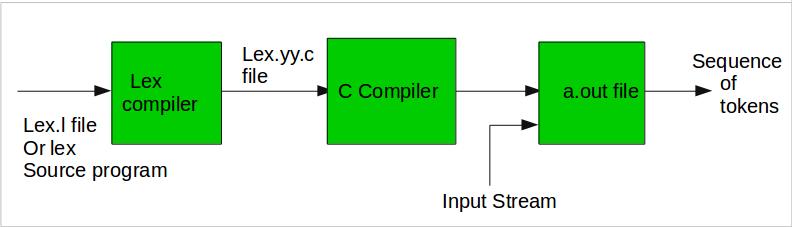
This will change the .l file to .yy.c file.

Step 3: Run command on C compiler

gcc lex.yy.c

As output of this we get a.out file

Step 4: ./a.out inputfile.cpp



**For our Analysis we use this input file:**

**Input File 1:**

**/\* Including Header file \*/**

**#include<bits/stdc++.h>**

**using namespace std;**

**/\* Swapping numbers \*/**

**void swap(int a,int b){**

**int c=a;**

**a=b;**

**b=c;**

**}**

**/\* Main Function \*/**

**int main(int argc, char \*argc[]){**

**int a=2;**

**int b=3;**

**printf("Number before swapping is a= %d and b= %d\n",a,b);**

**swap(a,b);**

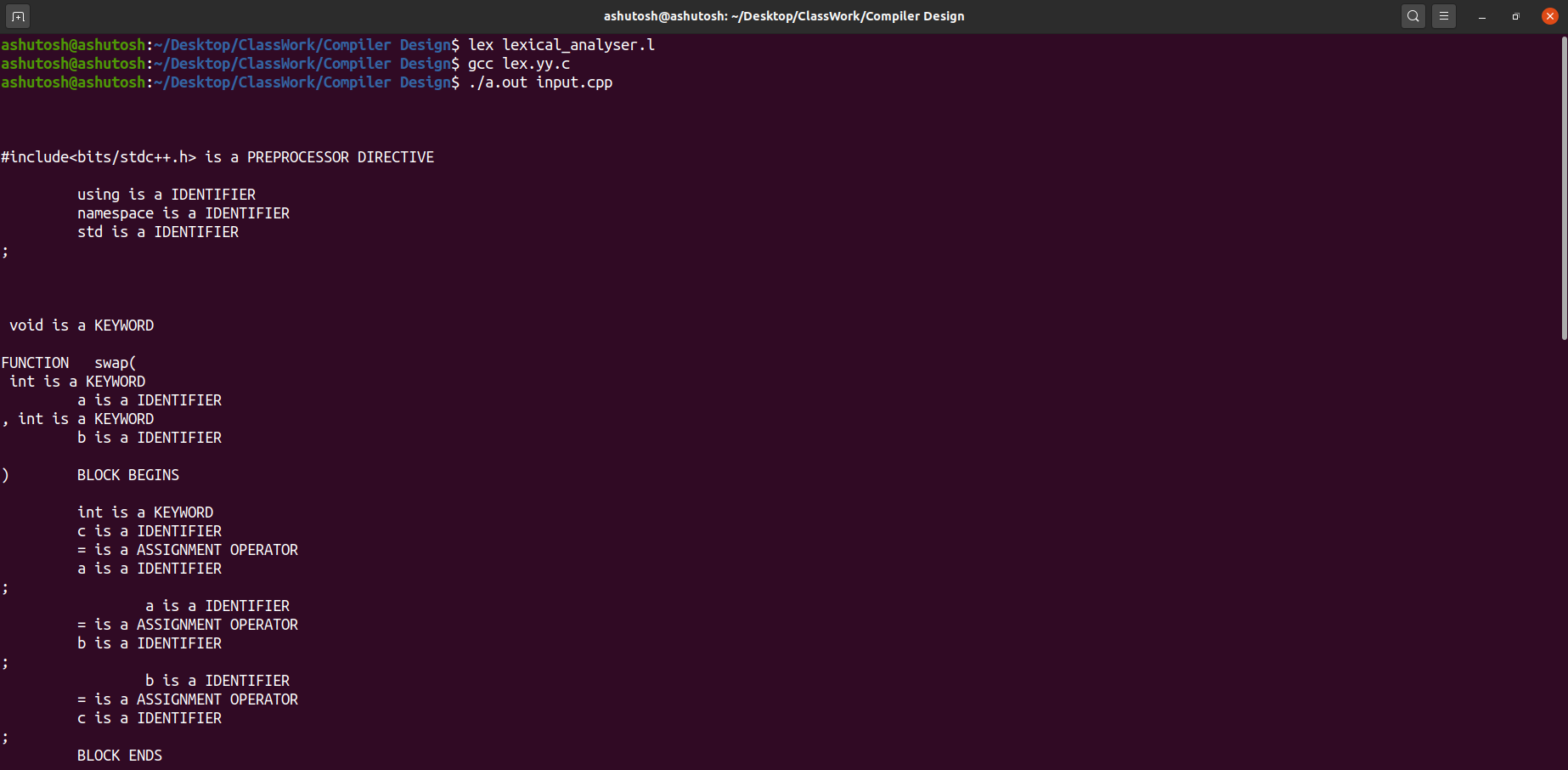
**printf("Number after swapping is a= %d and b= %d\n",a,b );**

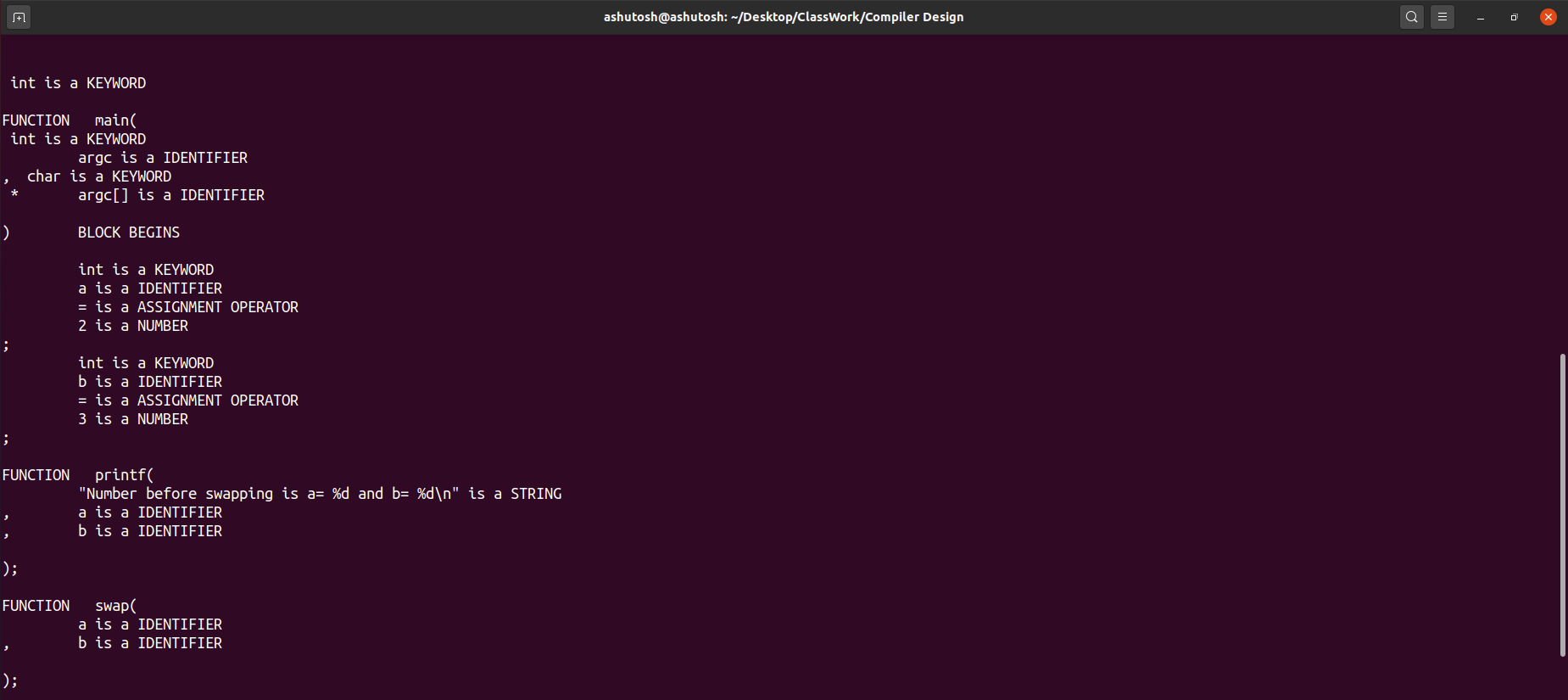
**printf("Sum of these number is %d\n",a+b );**

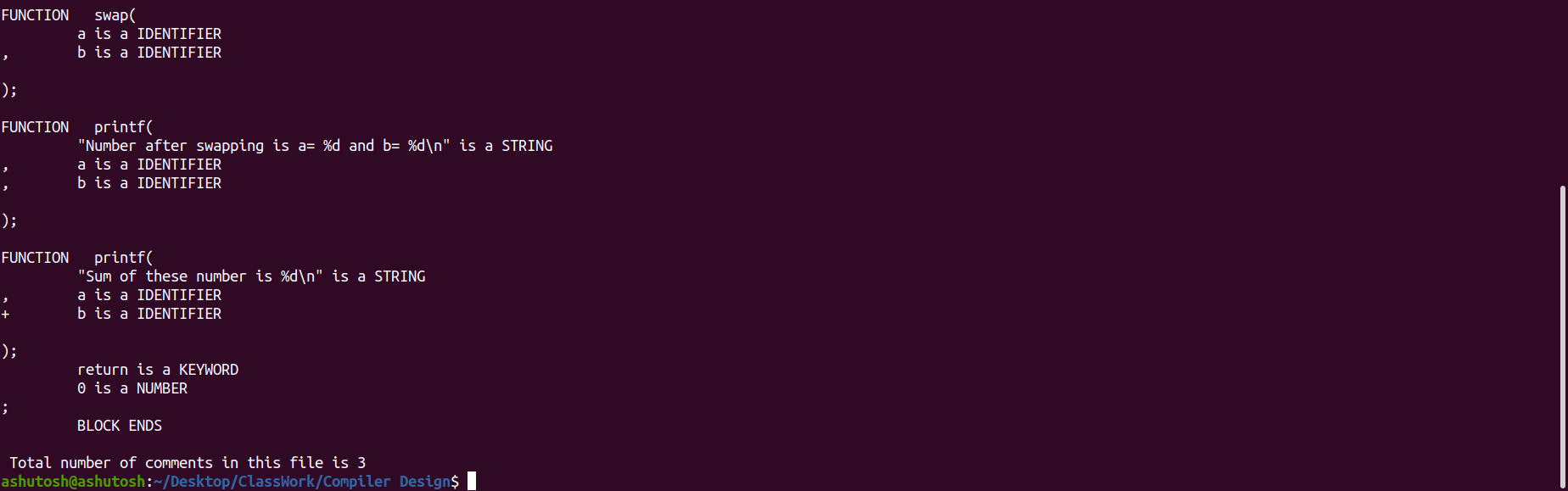
**return 0;**

**}**

**Result on Terminal:**

****





**Input File 2:**

**/\* Including Header File \*/**

**#include<stdio.h>**

**#include<conio.h>**

**#include<math.h>**

**#include<stdlib.h>**

**using namespace std;**

**/\* Sum Function\*/**

**int sum(int numberA,int numberB){**

**/\* Declaration of sum function \*/**

**int sum=0;**

**sum=numberA+numberB;**

**/\* Returning the value \*/**

**return sum;**

**}**

**/\* Main Function\*/**

**int main(int argv, char \*argc[]){**

**/\* Declaration on two numbers \*/**

**int numberA,numberB;**

**/\* Taking input of two numbers \*/**

**printf("Enter the two number which you want to add \n");**

**scanf("%d %d",&numberA,&numberB);**

**/\* Calling the function \*/**

**int result = sum(numberA,numberB);**

**/\* Output the result \*/**

**printf("Sum of these two number is %d\n",result );**

**return 0;**

**}**

**Result on Terminal:**

