**PASS 1**

**import java.util.\*;**

**public class pass1 {**

**public static void main(String args[]){**

**System.out.println("Enter no of statements: ");**

**Scanner sc = new Scanner(System.in);**

**int n = sc.nextInt();**

**System.out.println("Enter the mnemonic code: ");**

**String p[] = new String[n];**

**sc.nextLine();**

**for (int i = 0; i<n; i++){**

**p[i] = sc.nextLine();**

**}**

**System.out.println("Input: ");**

**for (int i = 0; i<n; i++){**

**System.out.println(p[i]);**

**}**

**HashMap<String,String> map = new HashMap<String,String>();**

**map.put("START","(AD,01)");**

**map.put("501", "(C, 501)");**

map.put("A","(S,0)");

map.put("DS","(DL,0)");

map.put("1","(C,1)");

map.put("B","(S,1)");

map.put("C","(S,2)");

map.put("READ","(IS,09)");

map.put("MOVER","(IS,04)");

map.put("AREG,","(RG,01)");

map.put("ADD","(IS,01)");

map.put("MOVEM","(IS,05)");

map.put("PRINT","(IS,10)");

map.put("END","(AD,02)");

**System.out.println("LC\tPASS1");**

**int j = 501;**

**for(int i = 0;i<n;i++){**

**if(i!=0){**

**System.out.print(j++);**

**}**

**String mnemonic[] = p[i].split(" ");**

**for(int k=0; k< mnemonic.length; k++){**

**System.out.print("\t"+map.get(mnemonic[k])+" ");**

**}**

**System.out.println();**

**}**

**}**

**}**

**PASS 2:**

**import java.util.\*;**

**public class pass2 {**

**public static void main(String args[]){**

**System.out.println("Enter no of statements: ");**

**Scanner sc = new Scanner(System.in);**

**int n = sc.nextInt();**

**System.out.println("Enter the mnemonic code: ");**

**String p[] = new String[n];**

**sc.nextLine();**

**for (int i = 0; i<n; i++){**

**p[i] = sc.nextLine();**

**}**

**System.out.println("Input: ");**

**for (int i = 0; i<n; i++){**

**System.out.println(p[i]);**

**}**

**HashMap<String,String> map = new HashMap<String,String>();**

**map.put("(AD,01) (c,501)", "-");**

**map.put("(S,0) (DL,0) (c,1)", "-");**

**map.put("(S,1) (DL,0) (c,1)", "-");**

**map.put("(S,2) (DL,0) (c,1)", "-");**

**map.put("(IS,09) (S,0)", "09 00 501");**

**map.put("(IS,09) (S,1)", "09 00 502");**

**map.put("(IS,04) (RG,01) (S,0)", "04 01 501");**

**map.put("(IS,01) (RG,01) (S,1)", "01 01 502");**

**map.put("(IS,05) (RG,01) (S,2)", "05 01 503");**

**map.put("(IS,05) (RG,01) (S,2)", "05 01 503");**

**map.put("(IS,10) (S,2)", "05 00 503");**

**map.put("(AD,02)", "-");**

**System.out.println("LC\tPASS1\t\t\tPASS2");**

**int j = 501;**

**for(int i = 0;i<n;i++){**

**if(i!=0){**

**System.out.print(j++);**

**}**

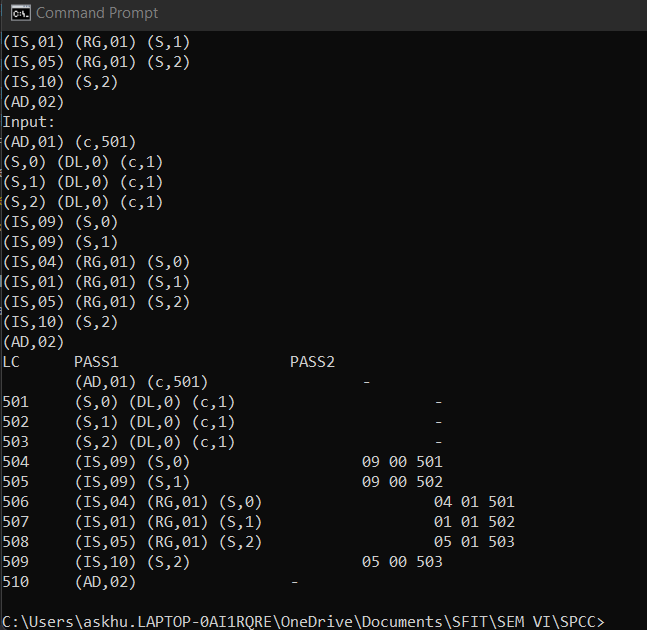
**System.out.print("\t"+p[i]+"\t\t\t"+map.get(p[i])+" ");**

**System.out.println();**

**}**

**}**

**}**

****

**Lex**

**To run:**

D:\FLEX>flex uplow.l

D:\FLEX>gcc lex.yy.c

D:\FLEX>a.exe

**To get out of code (o/p): ctrl C**

1. Write a LEX program for “Hello World” with output

%option noyywrap

%{

#include<stdio.h>

%} //delcare variables and stdio

%%

"hi" printf("Hello Word"); //rules

%%

int main()

{

printf("Enter hi to see the output");

yylex();

return 0;

}

1. Write a LEX program to count and identify uppercase and lowercase letters with output.

%option noyywrap

%{

#include<stdio.h>

int uppercase=0,lowercase=0;

%}

%%

[A-Z] {uppercase++; printf("Upper Case alphabet: %s\n", yytext);}

[a-z] {lowercase++; printf("Lower Case alphabet: %s\n", yytext);}

\n {printf("Total uppercase letters = %d \n Total lowercase letters = %d",uppercase,lowercase);}

%%

int main()

{

printf("Enter a string: ");

yylex();

return 0;

}

1. Write a LEX program to count and identify Vowels and consonants with output

%option noyywrap

%{

#include<stdio.h>

int consonant=0, vowel=0;

%}

%%

[aeiouAEIOU] {vowel++; printf("Vowel : %s\n", yytext);}

[^aeiouAEIOU\n] {consonant++; printf("Consonant: %s\n", yytext);}

\n {printf("Total Vowels = %d \nTotal Consonants = %d",vowel, consonant);}

%%

int main()

{

printf("Enter a string: ");

yylex();

return 0;

}

1. Write a LEX program to count and identify tokens with output

%option noyywrap

%{

#include<stdio.h>

int keyword=0, id=0, sp\_char=0, no=0, op=0;

%}

%%

"if"|"while"|"elif"|"break" {keyword++; printf("Keyword: %s\n", yytext);}

[A-z\_][A-z0-9\_]\* {id++; printf("Identifier: %s\n", yytext);} //variable

[0-9]+[0-9]\* {no++; printf("Number: %s\n", yytext);}

"="|"+"|"-"|"\*"|"/" {op++; printf("Operator: %s\n", yytext); }

"("|")"|"{"|"}"|";"|":" {sp\_char++; printf("Special Character: %s\n", yytext);}

\n printf("Total Keyword = %d \nTotal Identifier = %d \nTotal Special Character = %d \nTotal Number = %d \nTotal Operator = %d",keyword,id,sp\_char,no,op);

%%

int main()

{

printf("Enter expression: ");

yylex();

return 0;

}

1. Write a program to count the number of characters, words, sentences, lines, tabs, numbers and blank spaces present in input using LEX.

%option noyywrap

%{

#include<stdio.h>

int character = 0;

int word = 0;

int sentence = 0;

int line = 0;

int tab = 0;

int number = 0;

int space = 0;

int total\_character = 0;

%}

%%

[A-z]+[0-9]\* {word++; character=character+yyleng;}

[\n] {line++;}

[.] {sentence++; character=character+yyleng;}

[\t] {tab++; character=character+yyleng;}

[0-9] {number++; character=character+yyleng;}

" " {space++; character=character+yyleng;}

\n\n {printf("Characters Count = %d\nWords Count = %d\nSentences Count = %d\nLines Count = %d\nTabs Count = %d\nNumbers Count = %d\nSpaces Count = %d",character,word,sentence,line+1,tab,number,space);}

%%

int main()

{

printf("Enter Text : \n");

yylex();

return 0;

}

Write a Lex program to identify if the expression is valid or not. (Eg. 2+3 is valid expression and 4+ is invalid expression)

%option noyywrap

%{

#include<stdio.h>

#include<stdlib.h>

int var = 0, op = 0, b = 0;

%}

%%

[A-z0-9]+ {var++;}

"+"|"-"|"\*"|"/" {op++;}

"(" {b++;}

")" {b--;}

\n {if(b==0 && var>op){printf("Valid Expression");}else{printf("Invalid Expression");};exit(0);}

%%

int main()

{

printf("Enter expression: ");

yylex();

return 0;

}

##eleesaa

%{

#include<stdio.h>

#include<stdlib.h>

int c,d,bo=0,bc=0;

%}

operand [a-zA-Z0-9]+

operator [+\-\/\*]

%%

//the operand is one count higher than the operator if that fails then its is invalid eg a+b operand is two and operator is 1;

{operator} {d++;printf("%s is an operator \n",yytext);}

{operand} {c=d+1;printf("%s is an operand \n",yytext);}

"(" {if(bc<=bo)bo++;}

")" {bc++;}

\n {if(bo==bc&&c>d){printf("valid exp");}else {printf("invalid exp");};exit(0);}

%%

void main(){

yylex();

}

**Line count:**

%option noyywrap

%{

#include<stdio.h>

int line = 0;

int lala = 0;

%}

%%

[A-z][0-9]\* { lala++;}

[\n] {line++;}

\n\n {printf("Lines Count = %d",line+1);}

%%

int main()

{

printf("Enter Text : \n");

yylex();

return 0;

}

**Macros**

**Area.c**

#include<stdio.h>

#include "area.h"

int main()

{

int choice, n1, n2;

printf("Menu \n1.Square\n2.Rectangle\n3.Triangle\n4.Circle\n");

printf("Enter choice ");

scanf("%d",&choice);

switch (choice)

{

case 1:

printf("\nEnter length of the square\n");

scanf("%d", &n1);

printf("\nArea of Sauare is %d\n", sqArea(n1));

/\*break;\*/

case 2:

printf("Enter length rect\n");

scanf("%d" , &n1);

printf("Enter breadth rect\n");

scanf("%d" ,&n2);

printf("Area of Rect is %d\n", rectArea(n1,n2));

/\*break;\*/

case 3:

printf("Enter base of triangele\n");

scanf("%d", &n1);

printf("Enter height of triangele\n");

scanf("%d", &n2);

printf("Area of triangle is %lf\n", triArea(n1,n2));

/\*break;\*/

case 4:

printf("Enter radius of circle");

scanf("%d", &n1);

printf("Area of circle is %lf\n", circleArea(n1));

/\*break;\*/

default:

printf("\nEnter valid choice");

break;

}

return 0;

}

**Area.h**

#define sqArea(l) (l\*l)

#define rectArea(l, b) (l\*b)

#define triArea(b, h) (0.5\*b\*h)

#define circleArea(r) (3.14\*(r)\*(r))

**Conversion.c**

#include<stdio.h>

#include "conversion.h"

int main(){

float a;

int choice;

printf("\nMenu\n1.Celcius to fahrenheit\n2.fahrenheit to Celcius\n3.meter to feet\n4.feet to meter\n5.litre to cubic meter\n6cubic meter to liter\n");

printf("Enter choice");

scanf("%d", &choice);

switch(choice){

case 1:

printf("Enter the value in celcius");

scanf("%f", &a);

printf("The fahrenheit value is %f",ctof(a));

break;

case 2:

printf("Enter the value in F");

scanf("%f", &a);

printf("The C value is %f",ftoc(a));

break;

case 3:

printf("Enter the value in M");

scanf("%f", &a);

printf("The F value is %f",mtof(a));

break;

case 4:

printf("Enter the value in F");

scanf("%f", &a);

printf("The M value is %f",ftom(a));

break;

case 5:

printf("Enter the value in L");

scanf("%f", &a);

printf("The Cu value is %f",ltocu(a));

break;

case 6:

printf("Enter the value in Cu");

scanf("%f", &a);

printf("The L value is %f",cutol(a));

break;

default:

printf("Invalid choice");

break;

}

return 0;

}

**Conversion.h**

#define ctof(a) ((a \* 9/5)+ 32) /\*Celcius to fahrenheit\*/

#define ftoc(a) ((a - 32) \* 5/9) /\* fahrenheit to Celcius\*/

#define mtof(a) (a \* 3.281) /\*meter to feet\*/

#define ftom(a) (a / 3.281) /\*feet to meter\*/

#define ltocu(a) (a / 28.317) /\* litre to cubic meter\*/

#define cutol(a) (a \* 28.317) /\* cubic meter to liter\*/

**Leap, prime, fib.c**

#include <stdio.h>

#include "fib\_leap.h"

int main()

{

int n,a,b,i,temp;

int n1;

printf("Enter the choice: \n1. Fibonacci series \n2. Prime numbers\n3. Leap Years\n");

scanf("%d",&n);

switch(n)

{

case 1:

printf("\nEnter the total number of terms to get Fibonacci series: ");

scanf("%d",&n1);

fib(n1);

break;

case 2:

printf("\nCheck whether the number is prime or not: ");

scanf("%d",&n1);

prime(n1);

break;

case 3:

printf("\nEnter the year to check whether it is leap or not: ");

scanf("%d",&n1);

leap(n1);

break;

default:

printf("Wrong Choice");

}

return 0;

}

**Leap, prime, fib.h**

#define fib(n) a=0; b=1; printf("%d %d ", a,b); for(i=2;i<n;i++){temp = a; a = b; b = temp+b; printf("%d ", b); }

#define prime(n) a=0; if(n==2){printf("prime"); return 0;} for(i=2;i<n;i++){if(n%i==0) {a++;}} if(a==0){printf("prime");}else{printf("not prime");}

#define leap(n) if (((n%4==0) && (n%100!=0)) || (n%400==0)) {printf("Leap Year");}else{printf("Not Leap");}

**Fsf.c**

#include<stdio.h>

#include "fsf.h"

int main(){

int ch=0, n,i,f,sum;

while(ch!=4){

printf("\n1. Sum 2. Factorial 3. Factor 4. Exit\n");

scanf("%d", &ch);

printf("Enter n: ");

scanf("%d",&n);

switch(ch){

case 1: sum(n);

break;

case 2: factorial(n);

break;

case 3: factor(n);

break;

case 4: break;

}

}

return 0;

}

**Fsf.h**

#define factorial(n) f = 1;for (i = 1; i <= n; i++){ f = f \* i; } printf("Factorial of %d is %d", n, f);

#define sum(n) sum = 0; for (i = 0; i <= n; i++){ sum = sum + i;}printf("Sum of Natural Numbers till %d = %d", n, sum);

#define factor(n) i=1; for (i = 1; i <= n; ++i) { if (n % i == 0) { printf("%d ", i); }}

**Binary\_Convert.h**

#include <math.h>

#define bintodec(a) { \

int dec = 0, i = 0, rem; \

while (a != 0) \

{ \

rem = a % 10; \

a /= 10; \

dec += rem \* pow(2, i); \

++i; \

} \

printf("Decimal : %d", dec); \

}

#define dectobin(a) { \

long bin = 0; \

int rem, i = 1; \

while (a != 0) \

{ \

rem = a % 2; \

a /= 2; \

bin += rem \* i; \

i \*= 10; \

} \

printf("Binary : %d", bin); \

}

#define bintohex(a) { \

long int hex = 0, i = 1, rem; \

while (a != 0) \

{ \

rem = a % 10; \

hex = hex + rem \* i; \

i = i \* 2; \

a = a / 10; \

} \

printf("Equivalent hexadecimal value: %lX", hex); \

}

#define hextobin(hexdec) { \

long int i = 0; \

printf("Equivalent hexadecimal value:"); \

while (hexdec[i]) \

{ \

switch (hexdec[i]) \

{ \

case '0': \

printf("0000"); \

break; \

case '1': \

printf("0001"); \

break; \

case '2': \

printf("0010"); \

break; \

case '3': \

printf("0011"); \

break; \

case '4': \

printf("0100"); \

break; \

case '5': \

printf("0101"); \

break; \

case '6': \

printf("0110"); \

break; \

case '7': \

printf("0111"); \

break; \

case '8': \

printf("1000"); \

break; \

case '9': \

printf("1001"); \

break; \

case 'A': \

case 'a': \

printf("1010"); \

break; \

case 'B': \

case 'b': \

printf("1011"); \

break; \

case 'C': \

case 'c': \

printf("1100"); \

break; \

case 'D': \

case 'd': \

printf("1101"); \

break; \

case 'E': \

case 'e': \

printf("1110"); \

break; \

case 'F': \

case 'f': \

printf("1111"); \

break; \

default: \

printf("\nInvalid hexadecimal digit %c", \

hexdec[i]); \

} \

i++; \

} \

}

**Binary\_Convert.C**

#include <stdio.h>

#include <conio.h>

#include "Binary\_Convert.h"

int main()

{

int choice;

int a;

char c[100];

char hexdec[100];

printf("Menu \n1)Binary to Decimal \n2)Decimal to Binary \n3)Binary to Hexadecimal \n4)Hexadecimal to Binary\n");

scanf("%d", &choice);

switch (choice)

{

case 1:

printf("\nEnter Value in Binary:");

scanf("%d", &a);

bintodec(a);

break;

case 2:

printf("\nEnter Value in Decimal:");

scanf("%d", &a);

dectobin(a);

break;

case 3:

printf("\nEnter Value in Binary:");

scanf("%d", &a);

bintohex(a);

break;

case 4:

printf("\nEnter Value in Hexadecimal:");

scanf("%s", hexdec);

hextobin(hexdec);

break;

default:

printf("\nEnter a valid choice");

break;

}

return 0;

}

**3 Address code:( Quadruple, Triples)**

import java.util.\*;

public class Main{

public static void qQuadruple(String exp[]){

System.out.println("op\targ1\targ2\tresult");

for (int i=0;i < exp.length; i++){

String[]expr = exp[i].split("");

System.out.println(expr[3] + "\t" + expr[2] + "\t" + expr[4] + "\t" + expr[0]);

}

}

public static void rTriple(String exp[]){

System.out.println("op\targ1\targ2");

for (int i=0;i < exp.length; i++){

String[]expr = exp[i].split("");

System.out.println(expr[3] + "\t" + expr[2] + "\t" + expr[4]);

}

}

public static void main(String args[]){

Scanner sc = new Scanner(System.in);

System.out.println("Enter the number of expressions: \t");

int n = sc.nextInt();

System.out.println("Enter each expression in one line: \t");

sc.nextLine();

String expn[] = new String[n];

for (int i=0;i < n;i++)

{

expn[i]= sc.nextLine();

}

System.out.println("\n\n\tQuadruples: ");

System.out.println();

qQuadruple(expn);

System.out.println("\n\n\tTriples: ");

System.out.println();

rTriple(expn);

}

}