**Name:**

**Roll No. :**

**Year : 2014-15**

**;------------------------------------------------------------------------------------------------------------------------------------------**

**Aim : Write an application to parse input text file concurrently and compare the result of concurrent parsing with serial parsing.**

**;------------------------------------------------------------------------------------------------------------------------------------------**

**Lex File (cal.l)**

**%option reentrant**

**%option bison-bridge**

**%option noyywrap**

**%option bison-locations**

**%{**

**#include "parser.h"**

**%}**

**%%**

**[0-9]+ {sscanf(yytext, "%d", &yylval->value); return NUM;}**

**[-+\*/] {return yytext[0];}**

**\n {return(N);}**

**.|[ \t]+ { }**

**%%**

**yacc File (cal.y)**

**%define parse.error verbose**

**%define api.pure true**

**%locations**

**%token-table**

**%glr-parser**

**%lex-param {void \*scanner}**

**%parse-param {void \*scanner}**

**%{**

**/\* your top code here \*/**

**#include<omp.h>**

**int t;**

**%}**

**%union**

**{**

**int value;**

**}**

**%token <value> NUM**

**%token N**

**%left '+' '-'**

**%left '\*' '/'**

**%type <value> expr**

**%%**

**stmt :%empty**

**| stmt expr N {printf("\n Successful parsing by thread ID=%d\t Result of expression= %d\n",omp\_get\_thread\_num(),$2); }**

**;**

**expr**

**: expr '+' expr { $$ = $1+$3 ;}**

**| expr '-' expr { $$ = $1-$3 ;}**

**|expr '\*' expr { $$ = $1\*$3 ;}**

**|expr '/' expr { $$ = $1/$3 ;}**

**|NUM {$$=$1;}**

**;**

**%%**

**int yyerror()**

**{**

**}**

**Main File (main.c)**

**#include "parser.h"**

**#include "lexer.h"**

**//to read filenames for parsing**

**struct fname**

**{**

**char fn[10];**

**};**

**// to store temp. result of scanner**

**struct pwc**

**{**

**char op[20];**

**};**

**int main(int argc, char \*\*argv)**

**{**

**int result,n,i;**

**struct fname file[10];**

**struct pwc mypwc;**

**yyscan\_t scanner;**

**FILE \*f;**

**yylex\_init(&scanner);**

**printf("\n Enter no. of files: ");**

**scanf("%d",&n);**

**printf("\n Enter file names: ");**

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

**{**

**scanf("%s",file[i].fn);**

**}**

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

**#pragma omp parallel num\_threads(n) shared(file) private(f,scanner,result,mypwc)**

**{**

**if(yylex\_init\_extra(&mypwc, &scanner))**

**{**

**perror("init alloc failed");**

**}**

**int tid=omp\_get\_thread\_num();**

**f=fopen(file[tid].fn,"r");**

**yyset\_in(f,scanner);**

**result = (yyparse(scanner));**

**yylex\_destroy(scanner);**

**}**

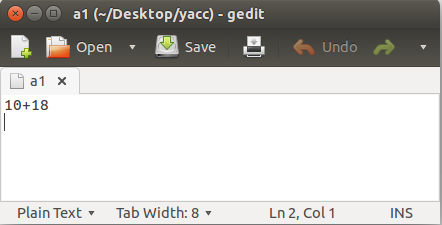
**printf("\n Parsing Completed…\n");**

**sleep(10);**

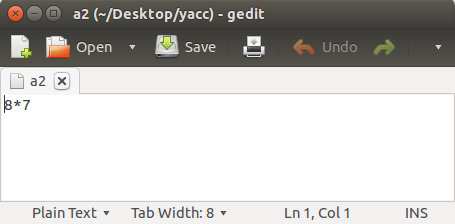
**return 0;**

**}**

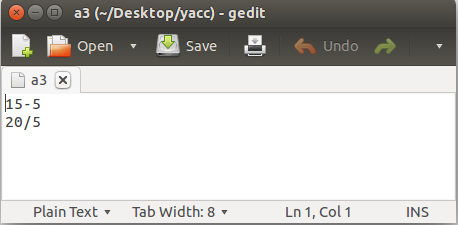
**Three Input Files :**

****

**a1.txt**

****

**a2.txt**

****

**a3.txt**

**-----------------------------------------------------------------OUTPUT---------------------------------------------------------------**

**pccoe@DMSA-19:~/Desktop/test$ flex --header-file=lexer.h --outfile=lexer.c cal.l**

**pccoe@DMSA-19:~/Desktop/test$ bison --output-file=parser.c --defines=parser.h --warning=all --feature=all cal.y**

**pccoe@DMSA-19:~/Desktop/test$ gcc -fopenmp lexer.c parser.c main.c**

**pccoe@DMSA-19:~/Desktop/test$ ./a.out**

**Enter no. of files: 3**

**Enter file names: a1 a2 a3**

**Successful parsing by thread ID=0 Result of expression= 28**

**Successful parsing by thread ID=2 Result of expression= 10**

**Successful parsing by thread ID=2 Result of expression= 4**

**Successful parsing by thread ID=1 Result of expression= 56**

**Parsing Completed…**