Eliminating and detecting left recursion :

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define SIZE 20

int main()

{

char pro[SIZE], alpha[SIZE], beta[SIZE];

int nont\_terminal,i,j, index=3;

printf("Enter the Production as E->E|A: ");

scanf("%s", pro);

nont\_terminal=pro[0];

if(nont\_terminal==pro[index]) //Checking if the Grammar is LEFT RECURSIVE

{

for(i=++index,j=0;pro[i]!='|';i++,j++){

alpha[j]=pro[i];

if(pro[i+1]==0){

printf("This Grammar CAN'T BE REDUCED.\n");

exit(0);

}

}

alpha[j]='\0';

if(pro[++i]!=0) //Checking if there is Character after Vertical Bar (|)

{

for(j=i,i=0;pro[j]!='\0';i++,j++){

beta[i]=pro[j];

}

beta[i]='\0';

printf("left Recursion exits");

printf("\nGrammar Without Left Recursion: \n\n");

printf(" %c->%s%c'\n", nont\_terminal,beta,nont\_terminal);

printf(" %c'->%s%c'|#\n", nont\_terminal,alpha,nont\_terminal);

}

else

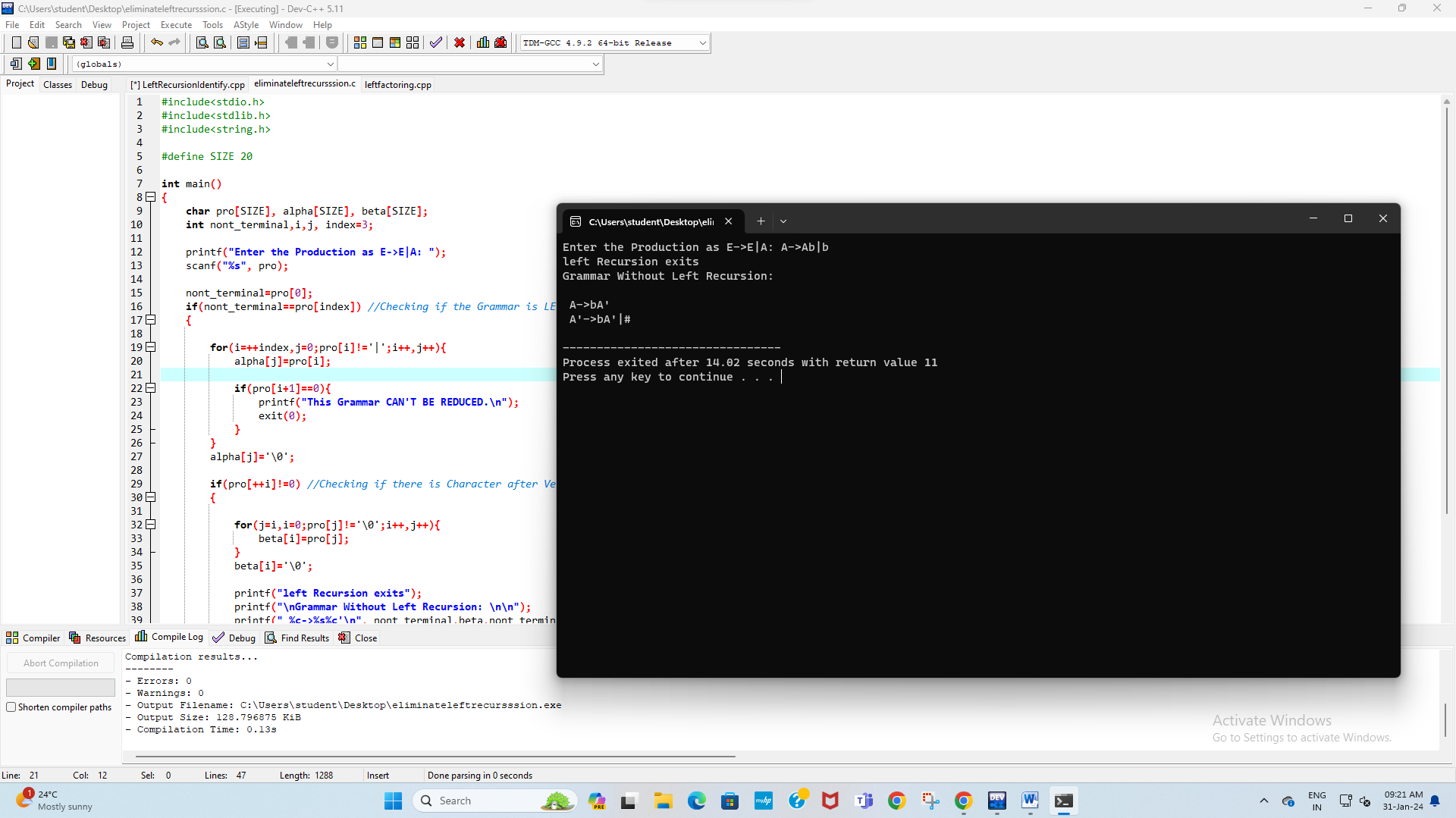
printf("This Grammar CAN'T be REDUCED.\n");

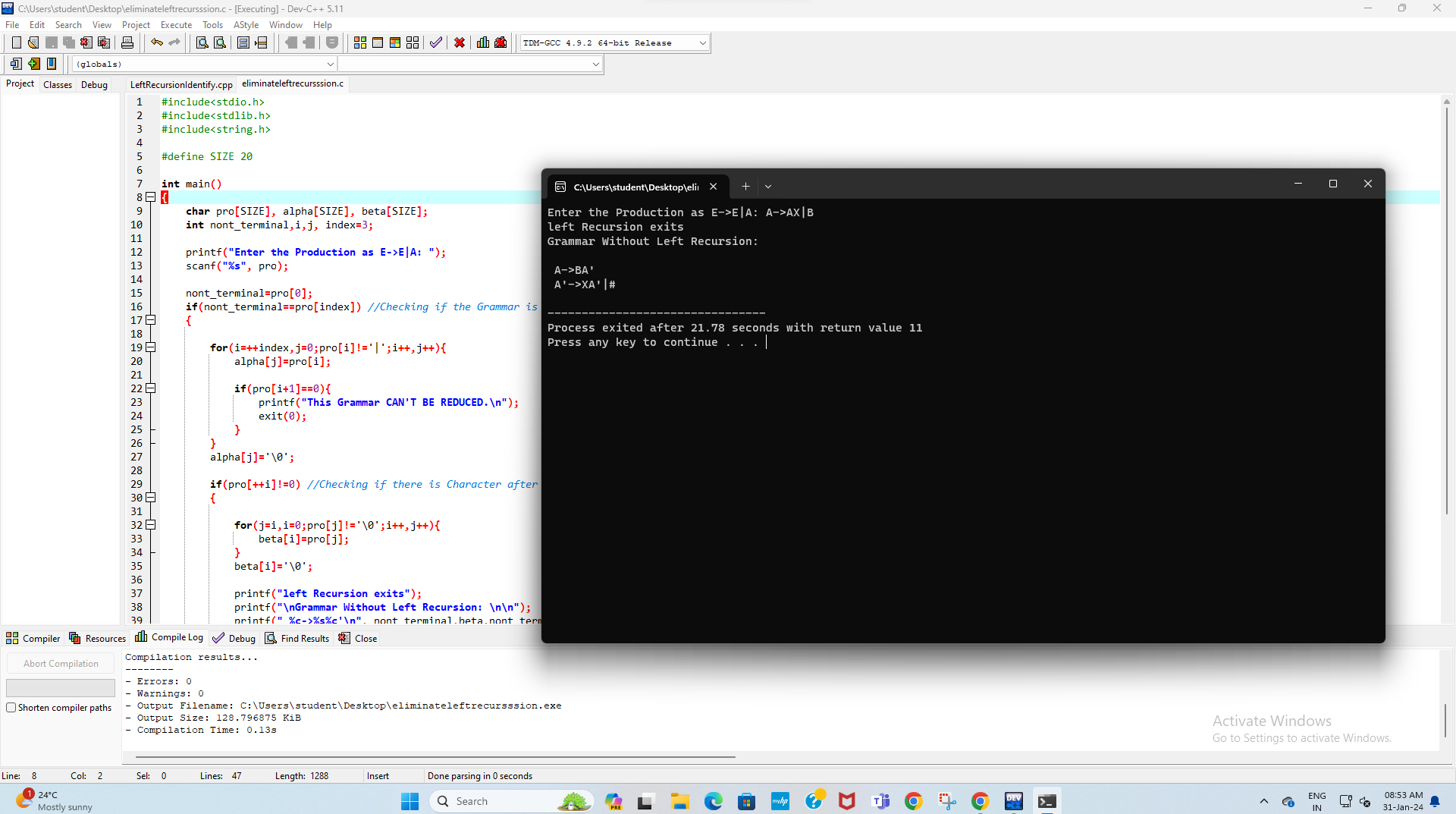
}

else

printf("\n This Grammar is not LEFT RECURSIVE.\n");

}





Left factoring :

#include<stdio.h>

#include<string.h>

int main()

{

char gram[20],part1[20],part2[20],modifiedGram[20],newGram[20],tempGram[20];

int i,j=0,k=0,l=0,pos;

printf("Enter Production : A->");

gets(gram);

for(i=0;gram[i]!='|';i++,j++)

part1[j]=gram[i];

part1[j]='\0';

for(j=++i,i=0;gram[j]!='\0';j++,i++)

part2[i]=gram[j];

part2[i]='\0';

for(i=0;i<strlen(part1)||i<strlen(part2);i++){

if(part1[i]==part2[i]){

modifiedGram[k]=part1[i];

k++;

pos=i+1;

}

}

for(i=pos,j=0;part1[i]!='\0';i++,j++){

newGram[j]=part1[i];

}

newGram[j++]='|';

for(i=pos;part2[i]!='\0';i++,j++){

newGram[j]=part2[i];

}

modifiedGram[k]='X';

modifiedGram[++k]='\0';

newGram[j]='\0';

printf("\nGrammar Without Left Factoring : : \n");

printf(" A->%s",modifiedGram);

printf("\n X->%s\n",newGram);

}

