Day 2

1.symbol Table:

#include<stdio.h>

#include<math.h>

#include<string.h>

#include<ctype.h>

#include<stdlib.h>

int main()

{

int x=0, n, i=0,j=0;

void \*mypointer,\*T4Tutorials\_address[5];

char ch,T4Tutorials\_Search,T4Tutorials\_Array2[15],T4Tutorials\_Array3[15],c;

printf("Input the expression ending with $ sign:");

while((c=getchar())!='$')

{

T4Tutorials\_Array2[i]=c;

i++;

}

n=i-1;

printf("Given Expression:");

i=0;

while(i<=n)

{

printf("%c",T4Tutorials\_Array2[i]);

i++;

}

printf("\n Symbol Table display\n");

printf("Symbol \t addr \t type");

while(j<=n)

{

c=T4Tutorials\_Array2[j];

if(isalpha(toascii(c)))

{

mypointer=malloc(c);

T4Tutorials\_address[x]=mypointer;

T4Tutorials\_Array3[x]=c;

printf("\n%c \t %d \t identifier\n",c,mypointer);

x++;

j++;

}

else

{

ch=c;

if(ch=='+'||ch=='-'||ch=='\*'||ch=='=')

{

mypointer=malloc(ch);

T4Tutorials\_address[x]=mypointer;

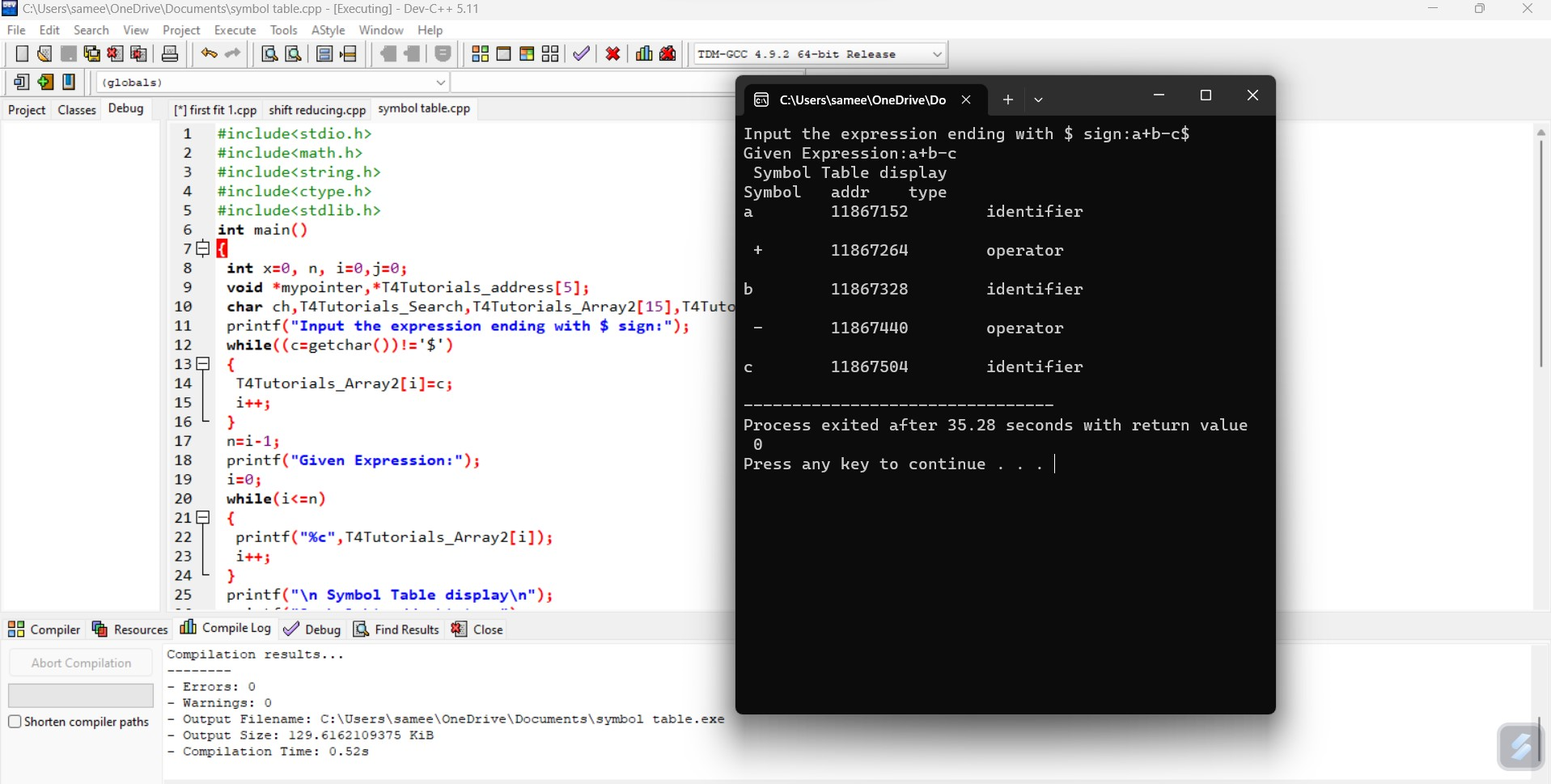
T4Tutorials\_Array3[x]=ch;

printf("\n %c \t %d \t operator\n",ch,mypointer);

x++;

j++;

}}}}



2.shift reducing parsing :

#include<stdio.h>

#include<conio.h>

#include<string.h>

int k=0,z=0,i=0,j=0,c=0;

char a[16],ac[20],stk[15],act[10];

void check();

int main( )

{

puts("GRAMMAR is E->E+E \n E->E\*E \n E->(E) \n E->id");

puts("enter input string ");

gets(a);

c=strlen(a);

strcpy(act,"SHIFT->");

puts("stack \t input \t action");

for(k=0,i=0; j<c; k++,i++,j++)

{

if(a[j]=='i' && a[j+1]=='d')

{

stk[i]=a[j];

stk[i+1]=a[j+1];

stk[i+2]='\0';

a[j]=' ';

a[j+1]=' ';

printf("\n$%s\t%s$\t%sid",stk,a,act);

check();

}

else

{

stk[i]=a[j];

stk[i+1]='\0';

a[j]=' ';

printf("\n$%s\t%s$\t%ssymbols",stk,a,act);

check();

}

}

getch();

}

void check()

{

strcpy(ac,"REDUCE TO E");

for(z=0; z<c; z++)

if(stk[z]=='i' && stk[z+1]=='d')

{

stk[z]='E';

stk[z+1]='\0';

printf("\n$%s\t%s$\t%s",stk,a,ac);

j++;

}

for(z=0; z<c; z++)

if(stk[z]=='E' && stk[z+1]=='+' && stk[z+2]=='E')

{

stk[z]='E';

stk[z+1]='\0';

stk[z+2]='\0';

printf("\n$%s\t%s$\t%s",stk,a,ac);

i=i-2;

}

for(z=0; z<c; z++)

if(stk[z]=='E' && stk[z+1]=='\*' && stk[z+2]=='E')

{

stk[z]='E';

stk[z+1]='\0';

stk[z+1]='\0';

printf("\n$%s\t%s$\t%s",stk,a,ac);

i=i-2;

}

for(z=0; z<c; z++)

if(stk[z]=='(' && stk[z+1]=='E' && stk[z+2]==')')

{

stk[z]='E';

stk[z+1]='\0';

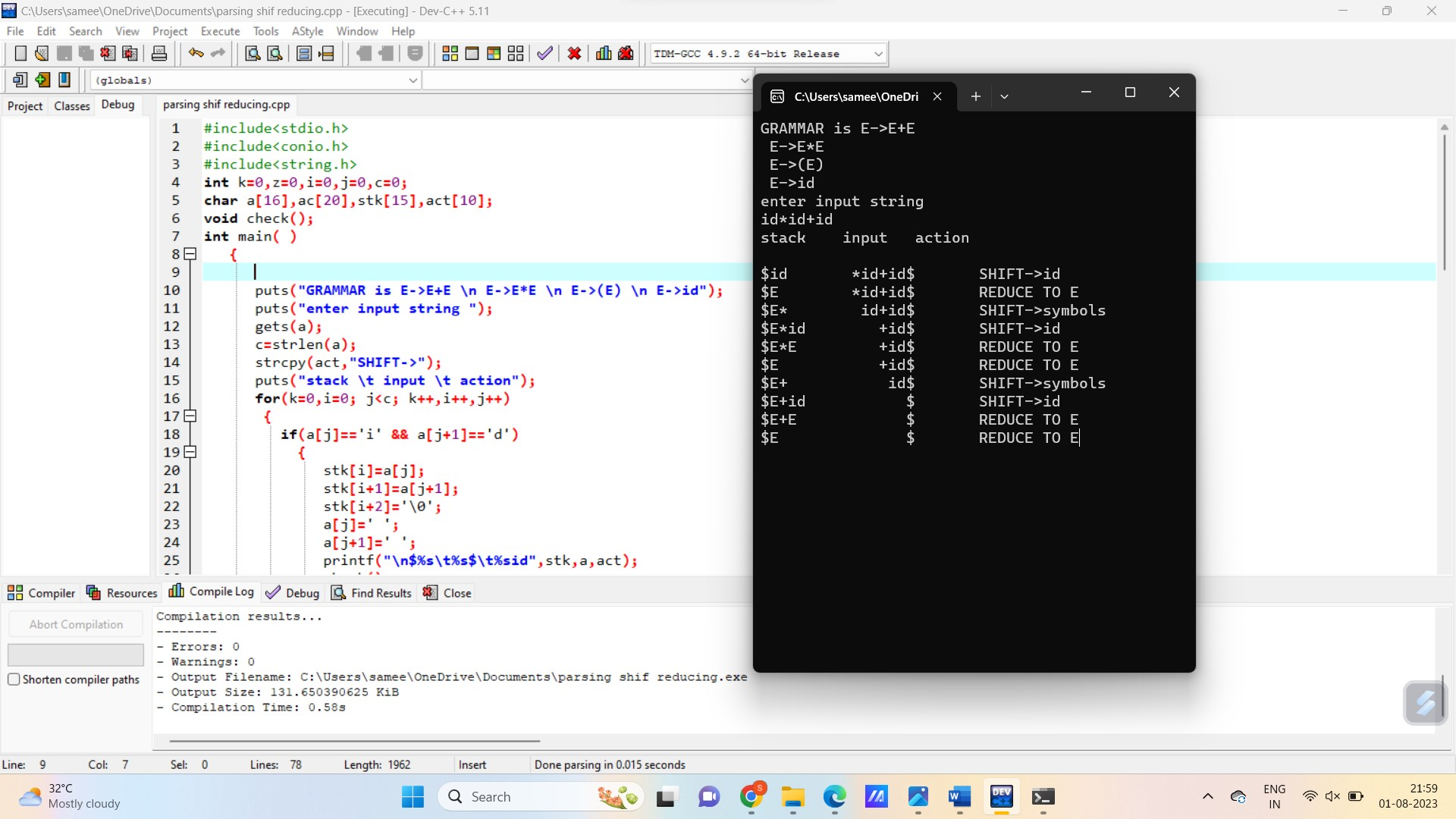
stk[z+1]='\0';

printf("\n$%s\t%s$\t%s",stk,a,ac);

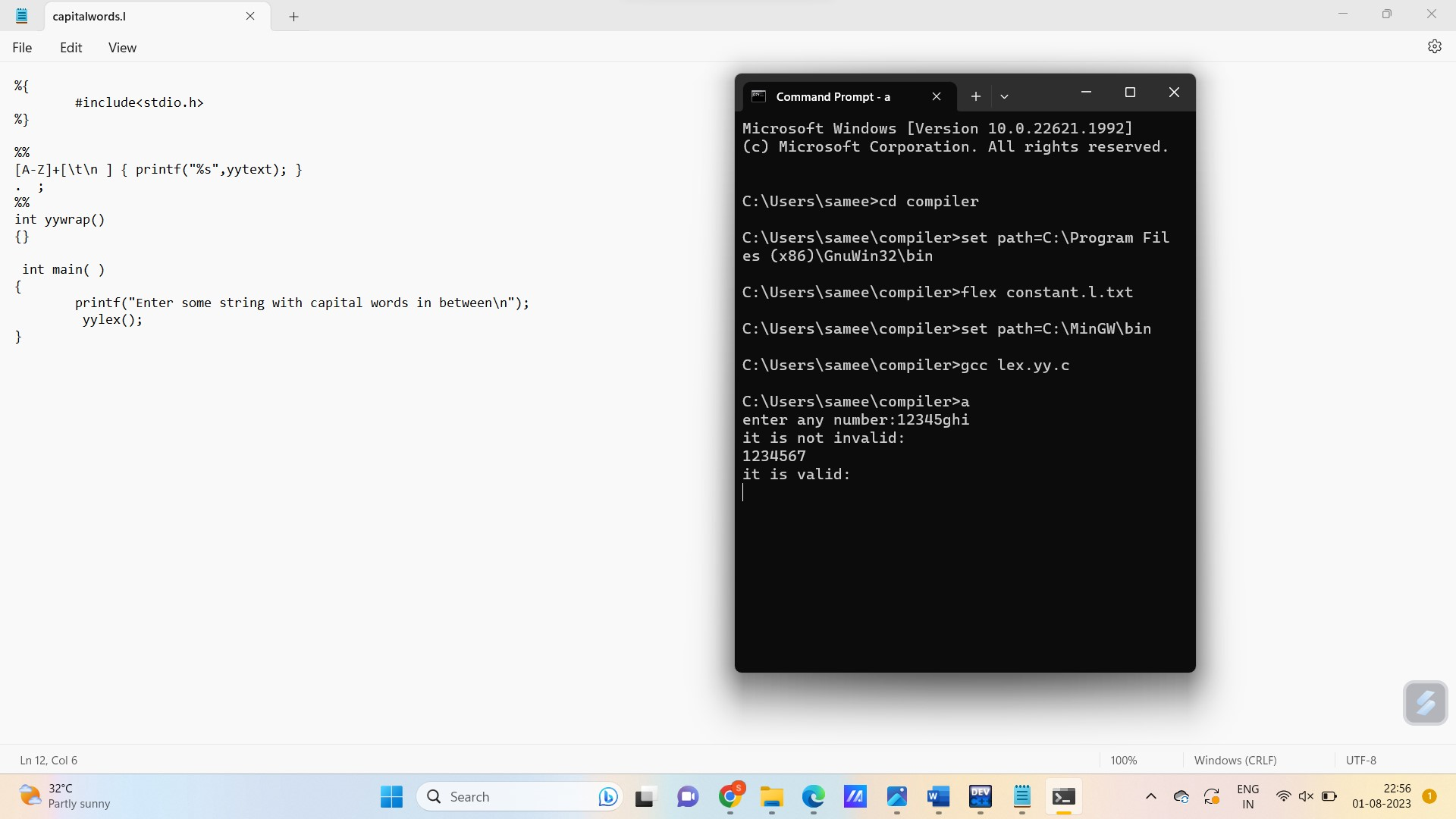
i=i-2;

}

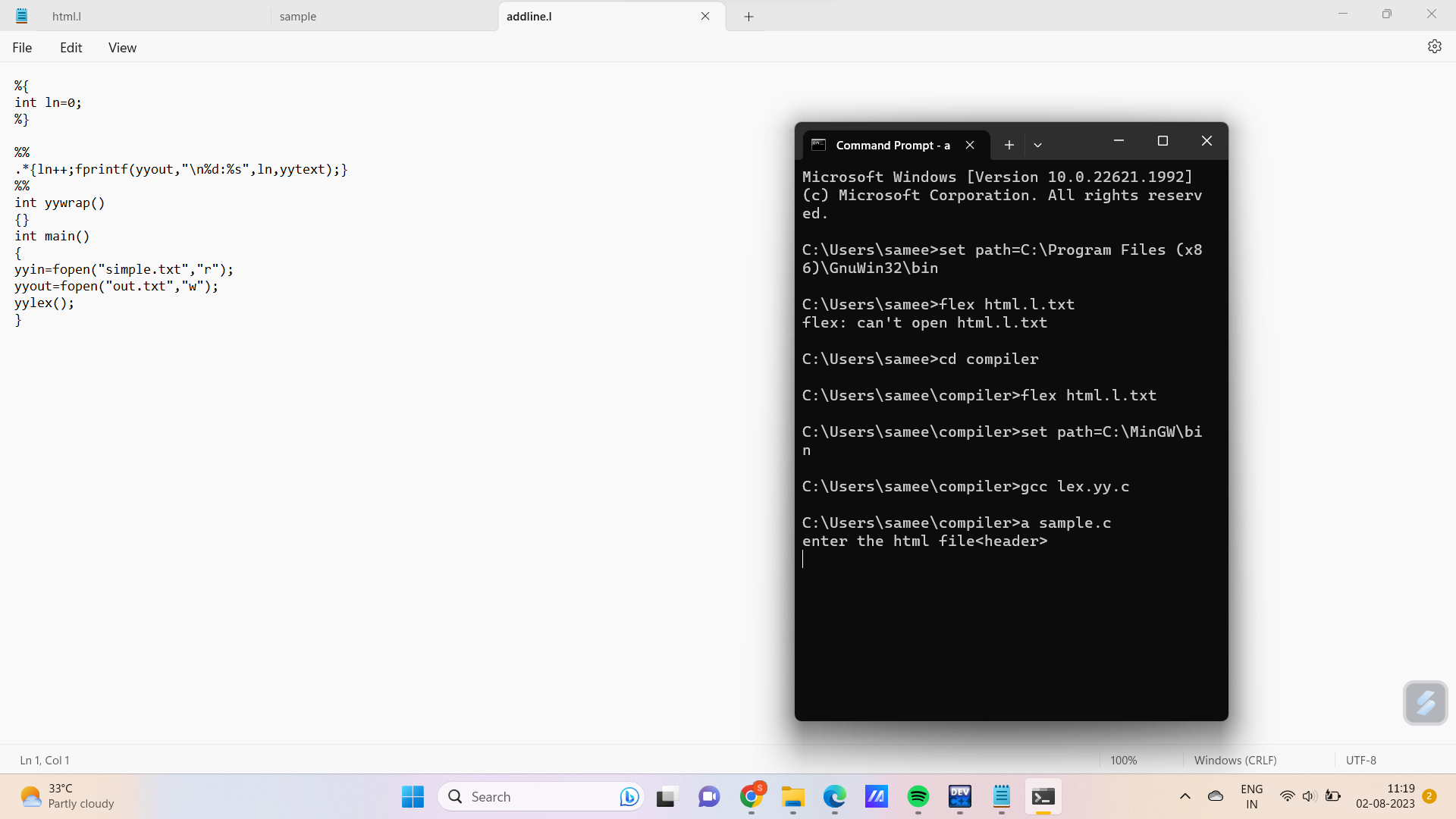
}



3.lex program for constants



4. Html lex



5.token recognition:

#include <stdbool.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

bool isValidDelimiter(char ch) {

if (ch == ' ' || ch == '+' || ch == '-' || ch == '\*' ||

ch == '/' || ch == ',' || ch == ';' || ch == '>' ||

ch == '<' || ch == '=' || ch == '(' || ch == ')' ||

ch == '[' || ch == ']' || ch == '{' || ch == '}')

return (true);

return (false);

}

bool isValidOperator(char ch){

if (ch == '+' || ch == '-' || ch == '\*' ||

ch == '/' || ch == '>' || ch == '<' ||

ch == '=')

return (true);

return (false);

}

bool isvalidIdentifier(char\* str){

if (str[0] == '0' || str[0] == '1' || str[0] == '2' ||

str[0] == '3' || str[0] == '4' || str[0] == '5' ||

str[0] == '6' || str[0] == '7' || str[0] == '8' ||

str[0] == '9' || isValidDelimiter(str[0]) == true)

return (false);

return (true);

}

bool isValidKeyword(char\* str) {

if (!strcmp(str, "if") || !strcmp(str, "else") || !strcmp(str, "while") || !strcmp(str, "do") || !strcmp(str, "break") || !strcmp(str, "continue") || !strcmp(str, "int")

|| !strcmp(str, "double") || !strcmp(str, "float") || !strcmp(str, "return") || !strcmp(str, "char") || !strcmp(str, "case") || !strcmp(str, "char")

|| !strcmp(str, "sizeof") || !strcmp(str, "long") || !strcmp(str, "short") || !strcmp(str, "typedef") || !strcmp(str, "switch") || !strcmp(str, "unsigned")

|| !strcmp(str, "void") || !strcmp(str, "static") || !strcmp(str, "struct") || !strcmp(str, "goto"))

return (true);

return (false);

}

bool isValidInteger(char\* str) {

int i, len = strlen(str);

if (len == 0)

return (false);

for (i = 0; i < len; i++) {

if (str[i] != '0' && str[i] != '1' && str[i] != '2'&& str[i] != '3' && str[i] != '4' && str[i] != '5'

&& str[i] != '6' && str[i] != '7' && str[i] != '8' && str[i] != '9' || (str[i] == '-' && i > 0))

return (false);

}

return (true);

}

bool isRealNumber(char\* str) {

int i, len = strlen(str);

bool hasDecimal = false;

if (len == 0)

return (false);

for (i = 0; i < len; i++) {

if (str[i] != '0' && str[i] != '1' && str[i] != '2' && str[i] != '3' && str[i] != '4' && str[i] != '5' && str[i] != '6' && str[i] != '7' && str[i] != '8'

&& str[i] != '9' && str[i] != '.' || (str[i] == '-' && i > 0))

return (false);

if (str[i] == '.')

hasDecimal = true;

}

return (hasDecimal);

}

char\* subString(char\* str, int left, int right) {

int i;

char\* subStr = (char\*)malloc( sizeof(char) \* (right - left + 2));

for (i = left; i <= right; i++)

subStr[i - left] = str[i];

subStr[right - left + 1] = '\0';

return (subStr);

}

void detectTokens(char\* str) {

int left = 0, right = 0;

int length = strlen(str);

while (right <= length && left <= right) {

if (isValidDelimiter(str[right]) == false)

right++;

if (isValidDelimiter(str[right]) == true && left == right) {

if (isValidOperator(str[right]) == true)

printf("\nValid operator : '%c'", str[right]);

right++;

left = right;

} else if (isValidDelimiter(str[right]) == true && left != right || (right == length && left != right)) {

char\* subStr = subString(str, left, right - 1);

if (isValidKeyword(subStr) == true)

printf("\nValid keyword : '%s'", subStr);

else if (isValidInteger(subStr) == true)

printf("\nValid Integer : '%s'", subStr);

else if (isRealNumber(subStr) == true)

printf("\nReal Number : '%s'", subStr);

else if (isvalidIdentifier(subStr) == true

&& isValidDelimiter(str[right - 1]) == false)

printf("\nValid Identifier : '%s'", subStr);

else if (isvalidIdentifier(subStr) == false

&& isValidDelimiter(str[right - 1]) == false)

printf("\nInvalid Identifier : '%s'", subStr);

left = right;

}

}

return;

}

int main(){

char str[100] = "float x = a + 1b; ";

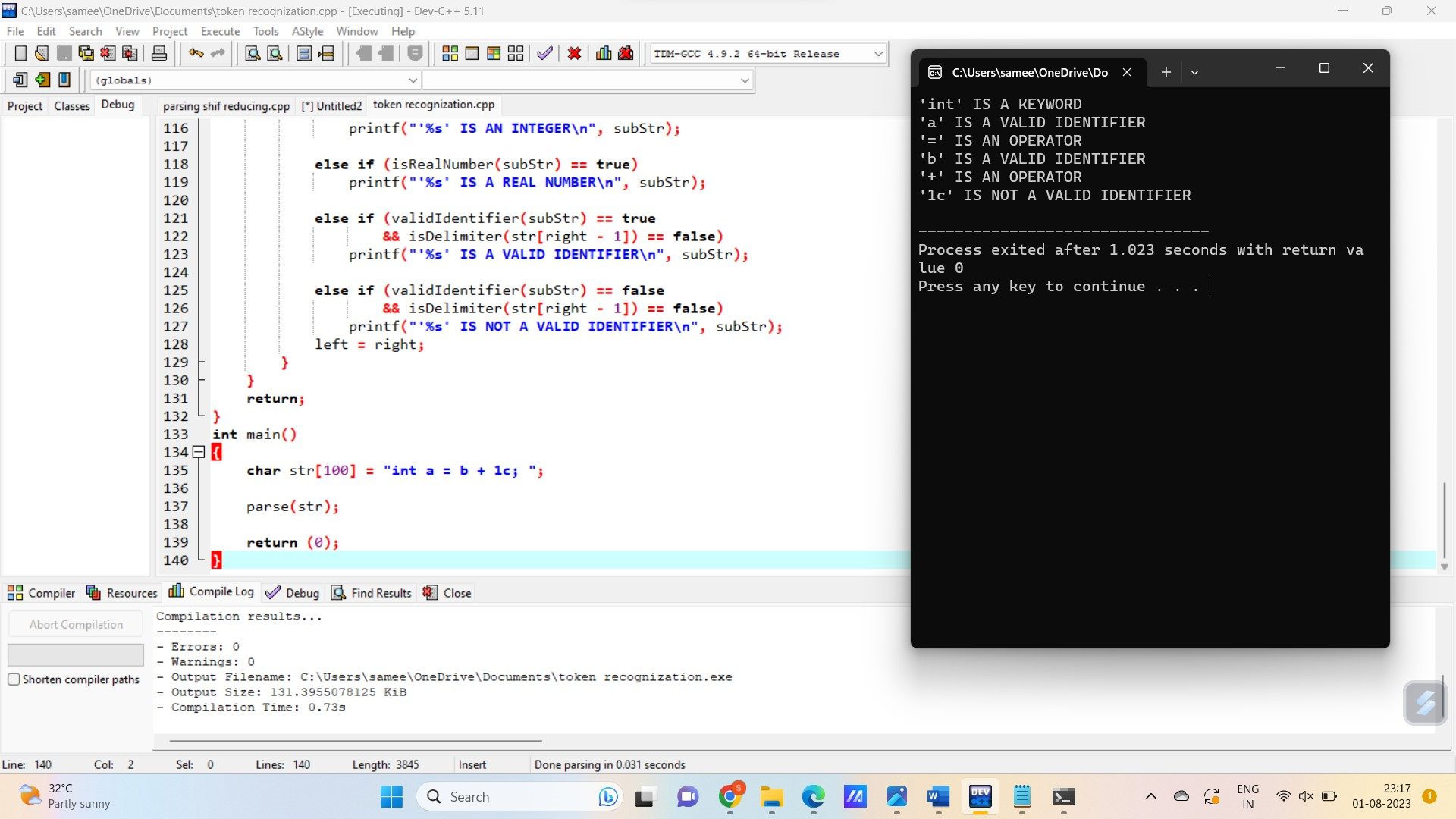
printf("The Program is : '%s' ", str);

printf("All Tokens are : ");

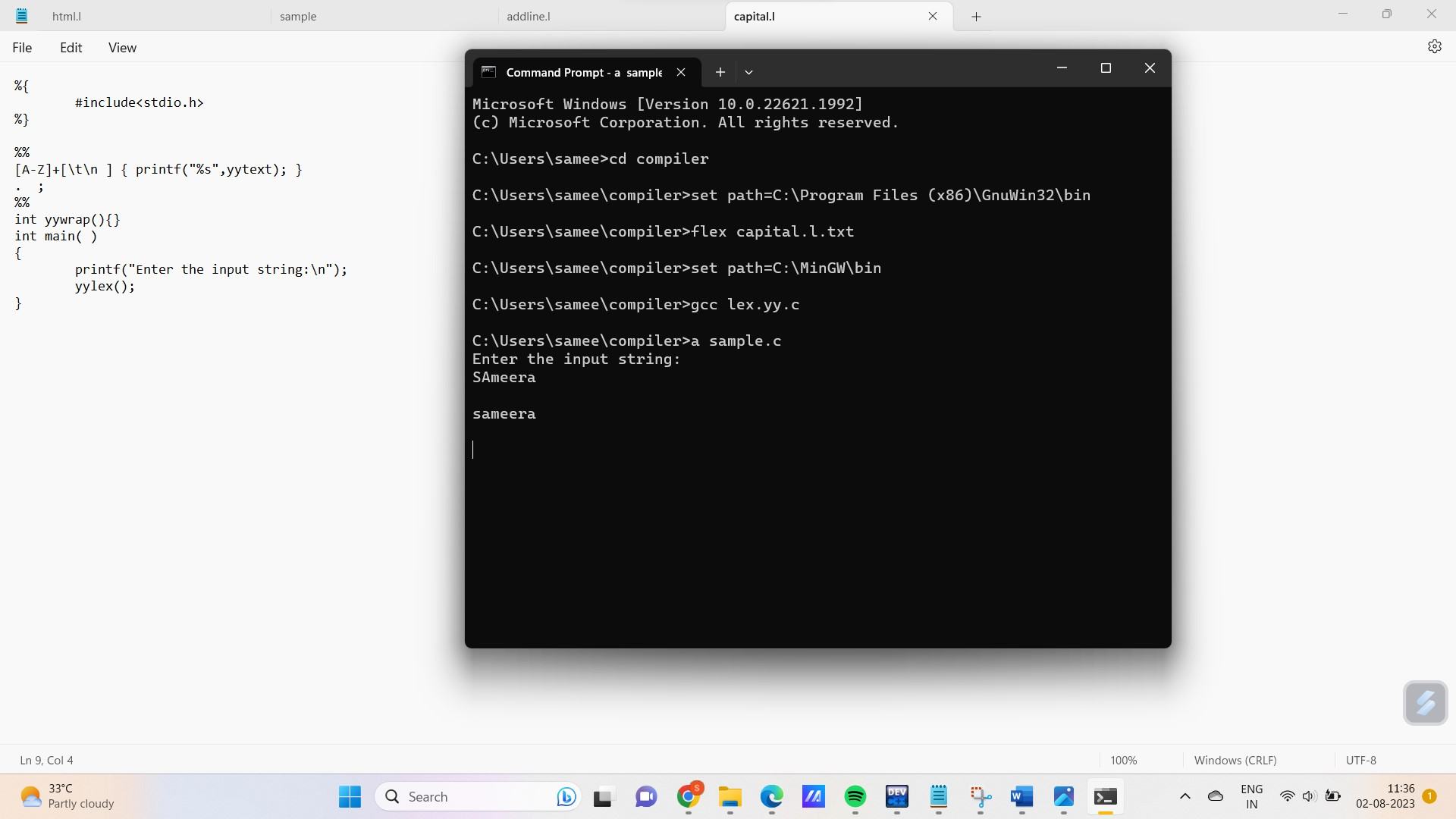
detectTokens(str);

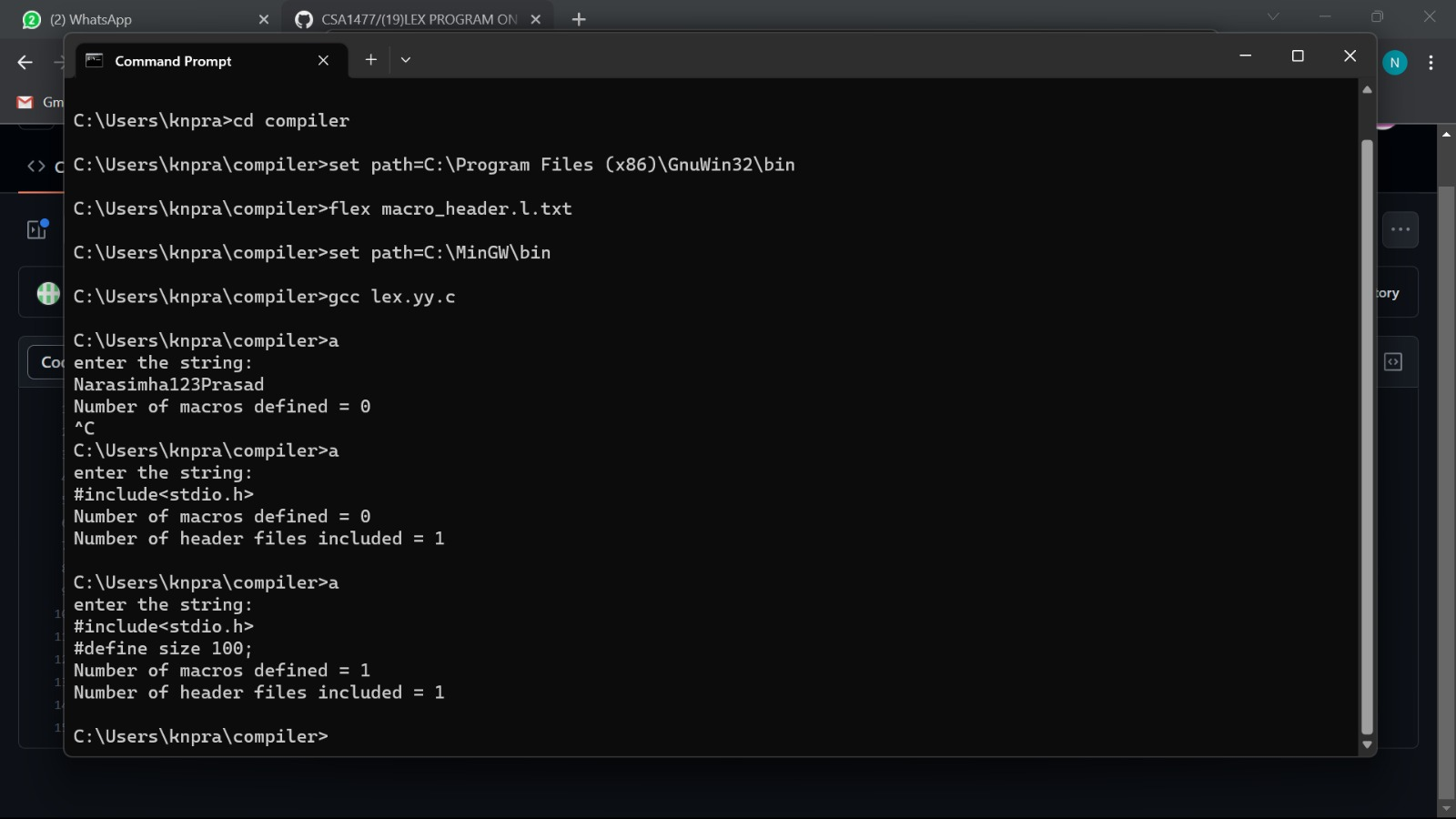
return (0);

}



6.capital:



7. macro 

8.