06/11/23

LAB - 1

Program 1:

To identify keywords, identifiers and seperators in a C program

```
%option noyywrap
%{
    #include<stdio.h>
%}
%%
[int|float|char] {printf("Keyword");}
[a-z A-z]* {printf("Identifier");}
,|; {printf("Seperator");}
%%
void main()
{
  yylex();
}
```

Program 2:

lex program to identify whether the entered input is a number, operator or invalid character. It should ignore white space, tab space.

```
%option noyywrap
%{
    #include<stdio.h>
%}
%%
[0-9]* {printf("Number: %s\n", yytext);}
[+ -] {printf("Operator: %s\n", yytext);}
[ \t\n] {/* ignore whitespace and newline */}
[a-z A-Z]* {printf("Invalid character: %s\n", yytext);}
%%
```

```
int main()
{
  printf("Enter ");
  yylex();
  return 0;
}
```

OUTPUT:

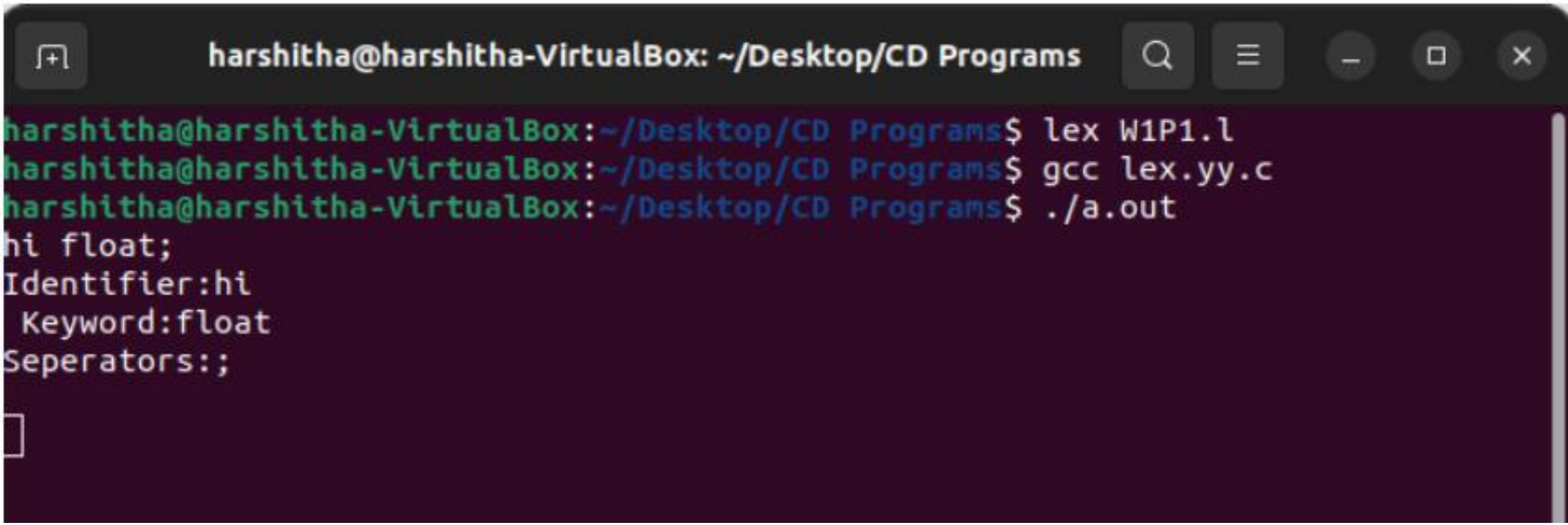
```
lex program2.l
gcc lex.yy.c
./a.out

Enter
*
Operator : *
12
Number : 12
46kj+
Number : 46
Invalid character : kj
Operator : +
```

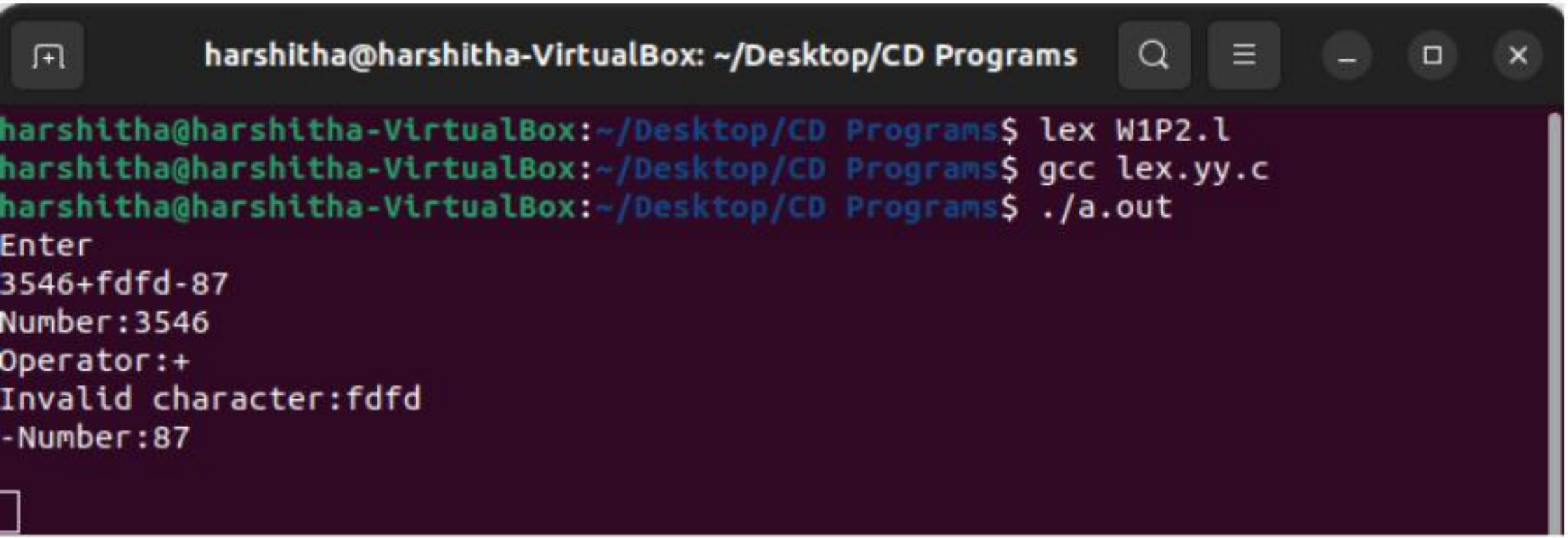
Program 3 : Echo program

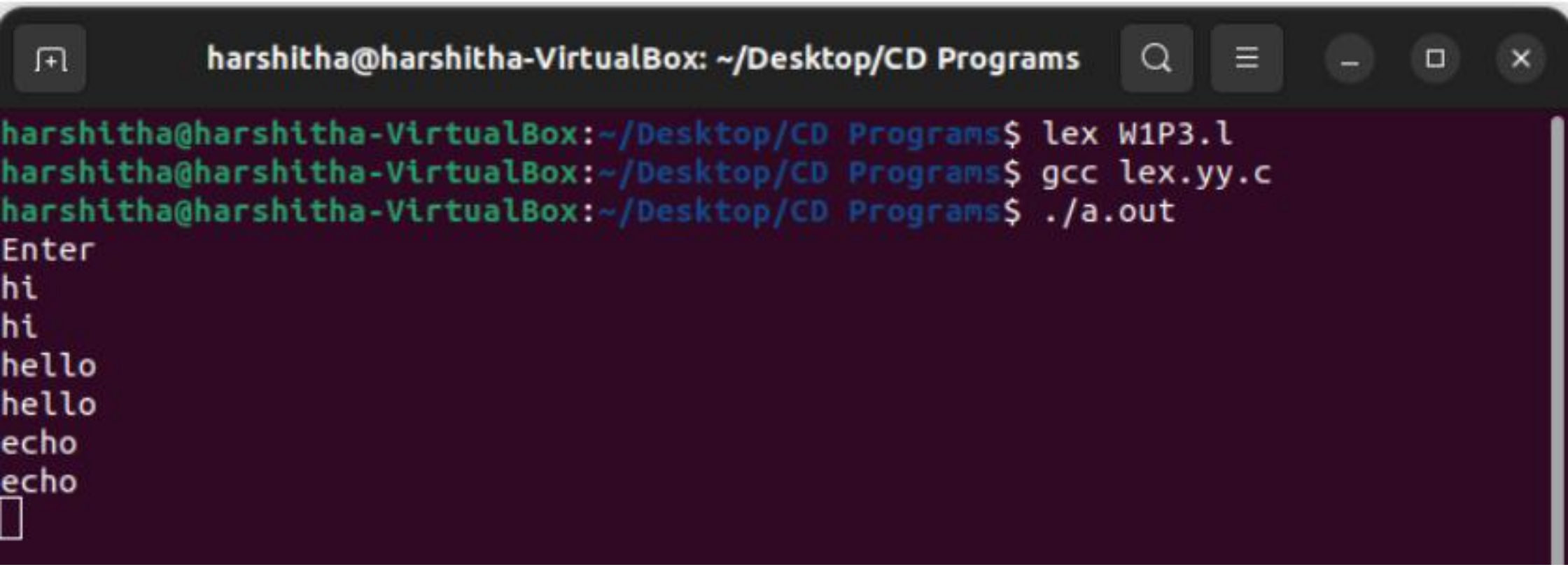
```
%%
.ECHO;
%%
int yywrap(void)
{
}
int main(void)
{
  yylex();
  return 0;
}
```

OUTPUT:

```
hello
hello
```


harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W1P1.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
hi float;
Identifier:hi
Keyword:float
Seperators:;


harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W1P2.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter
3546+fdfd-87
Number:3546
Operator:+
Invalid character:fdfd
-Number:87


harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W1P3.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter
hi
hi
hello
hello
echo
echo

20/11/23

LAB-2

Program 4: Write a LEX program to identify datatype - int, char, float and variable

```
%option noyywrap
%{
   #include<stdio.h>
%}
%%
[-+]?[0-9]+ {printf ("int");}
[a-z A-Z]+ {printf ("char");}
[-+]?[0-9]*[.][0-9]* {printf ("float");}
[a-z A-Z]* {printf ("variable");}
%%
void main()
{
   yylex();
}
```

Program 5: Write a LEX program to identify each character as consonants / vowels.

```
[aeiouAEIOU] {printf ("vowel");}
[a-z A-Z) {printf ("consonant");}
```

Program 6: Write a LEX program to identify alphabets as characters and numbers as digits

```
[+-]?[0-9]* {printf ("%s is a digit", yytext);}
[a-z A-Z]* {printf ("%s is stream of characters", yytext);}
```

Date __/__/__

Page _____

Program 7: Write a LEX program to count the number of words in an input sentence.

```
%option noyywrap
%{
   #include <stdio.h>
%}
 int len = 0;
%%
([a-zA-Z0-9])* {len++;}
"\n" {printf("Number of words in the given
                sentence is %d", len);}
%%
void main()
 {
   yylex();
}
```

Program 8: Write a LEX program to count the number of vowels and consonants in a given string.

```
  int vowel = 0;
  int consonant = 0;
%%
[aeiouAEIOU] {vowel++;}
[a-zA-Z] {consonant++;}

"\n" {printf("Number of vowels is %d and
        Number of consonants is %d", vowel,
        consonant);}
%%
```

30/11/22

Program 9: Identify alphanumeric string

```
[a-zA-Z]* {printf("Alphabets");}
[0-9]* {printf("Numbers");}
[a-zA-Z0-9]* {printf("Alphanumeric");}
```

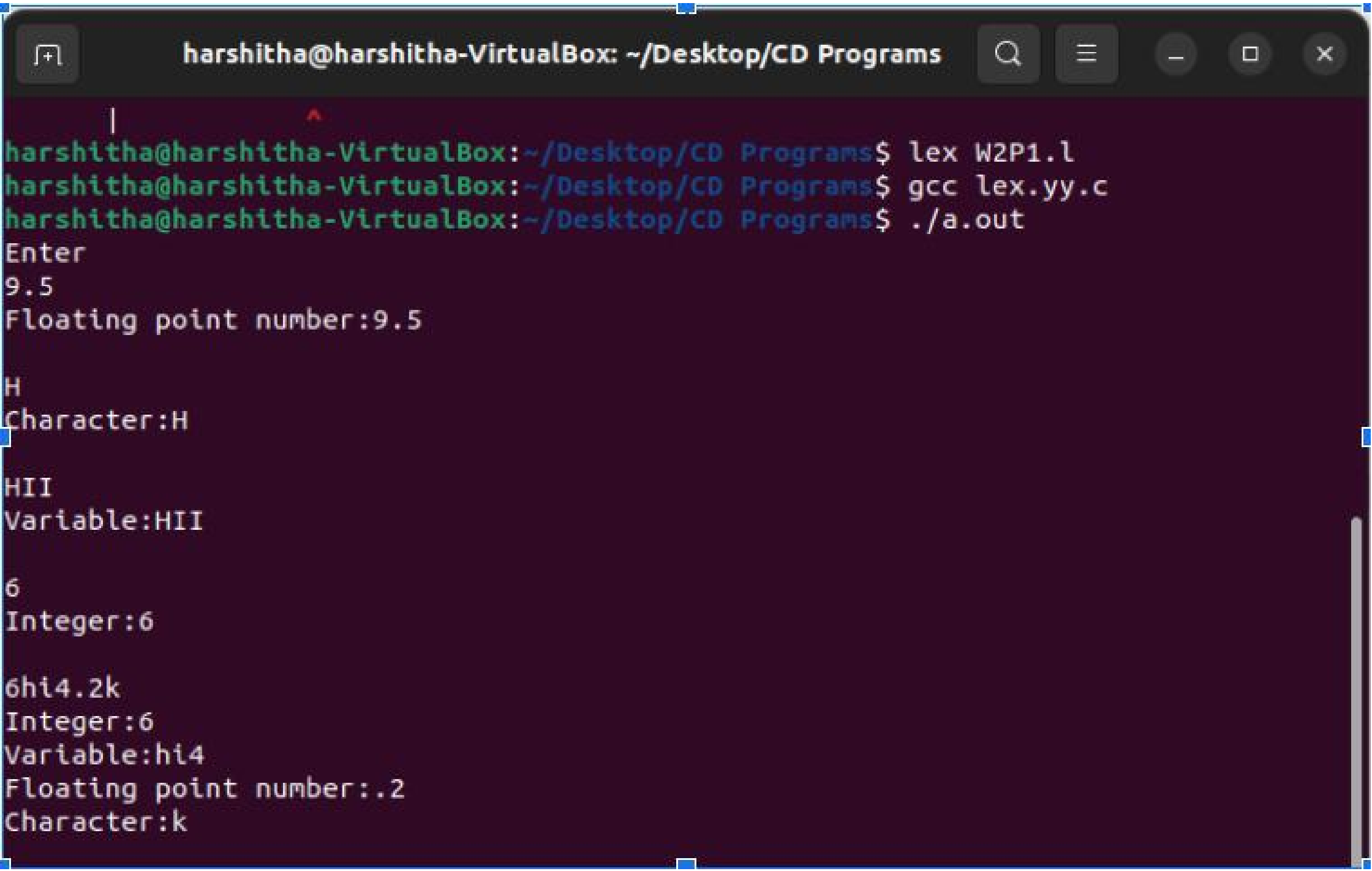
Program 10:
Read input from file and print on terminal

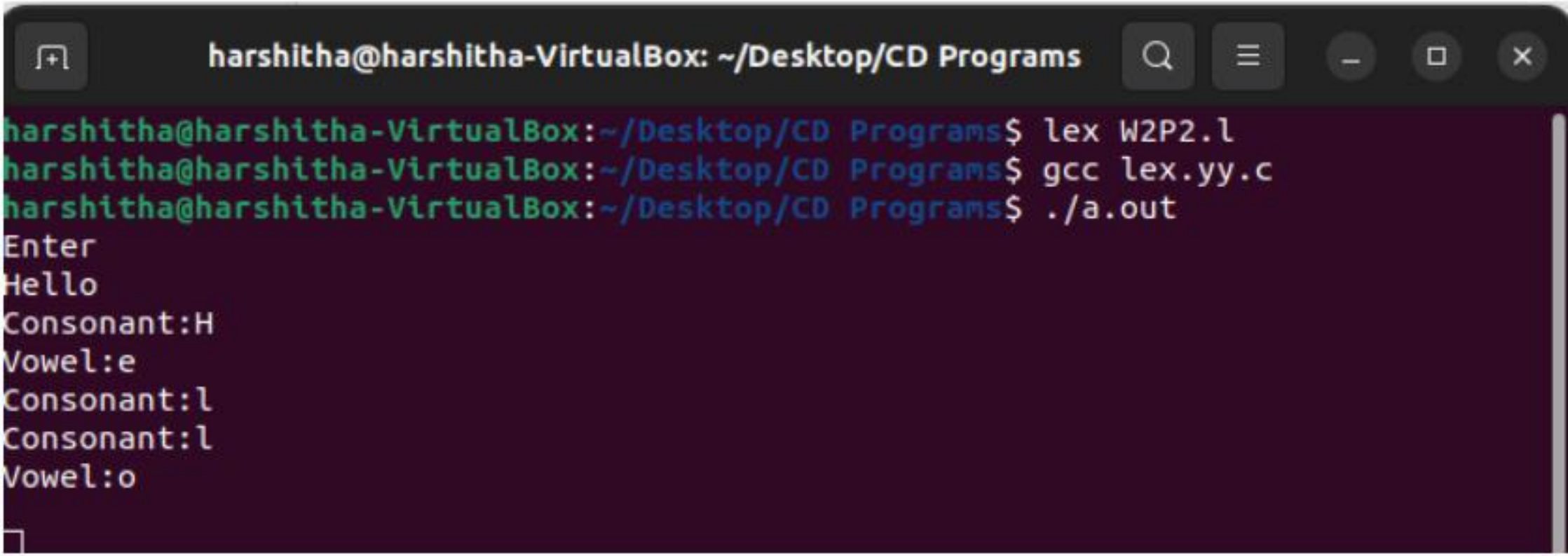
```
void main()
{
  char fname[10];
  printf("Enter file name");
  scanf("%s", fname);
  yyin = fopen(fname, "r");
  yylex();
  fclose(yyin);
}
```

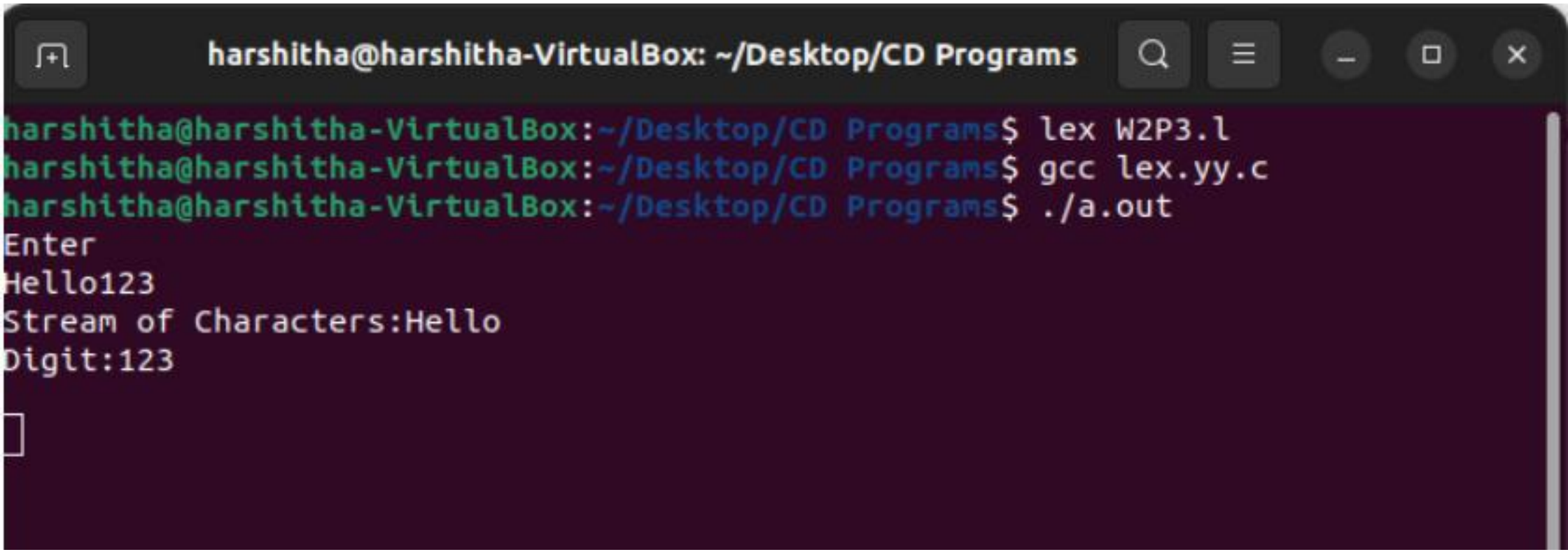
Program 11:
Read input from file but output stored in another file. Ask for output file name.

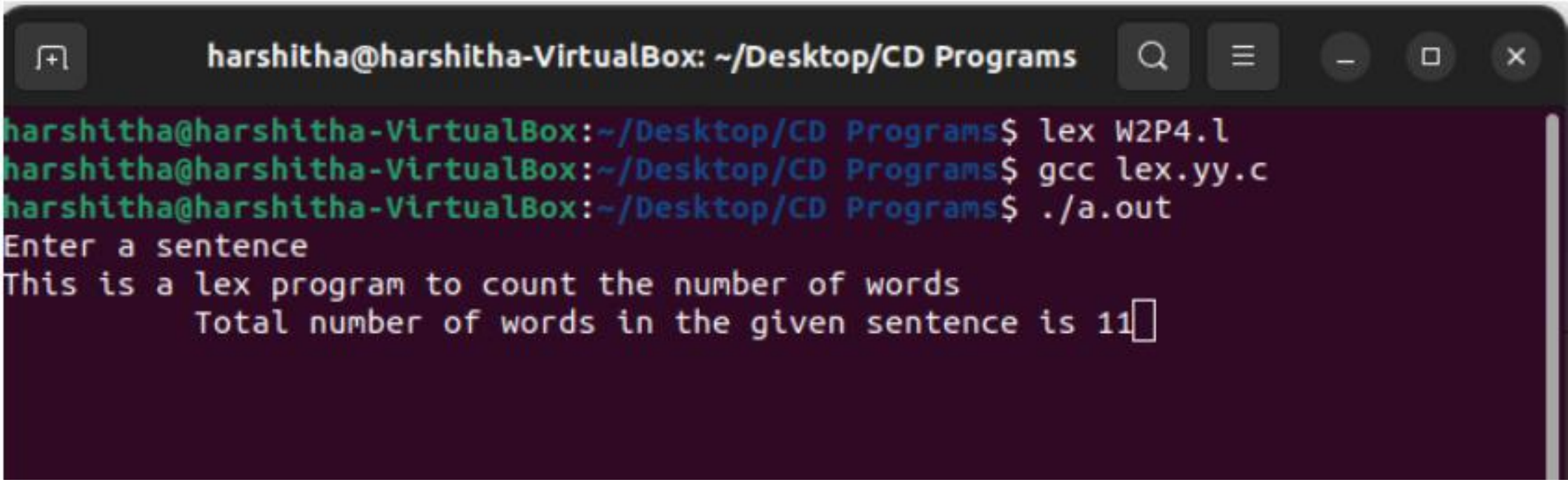
```
yyin = fopen(fname, "r");
yyout = fopen(ofname, "w");
yylex();
fclose(yyin)
fclose(yyout);
return 0;

int main(){
  char fname[20];
  extern FILE *yyin;
```

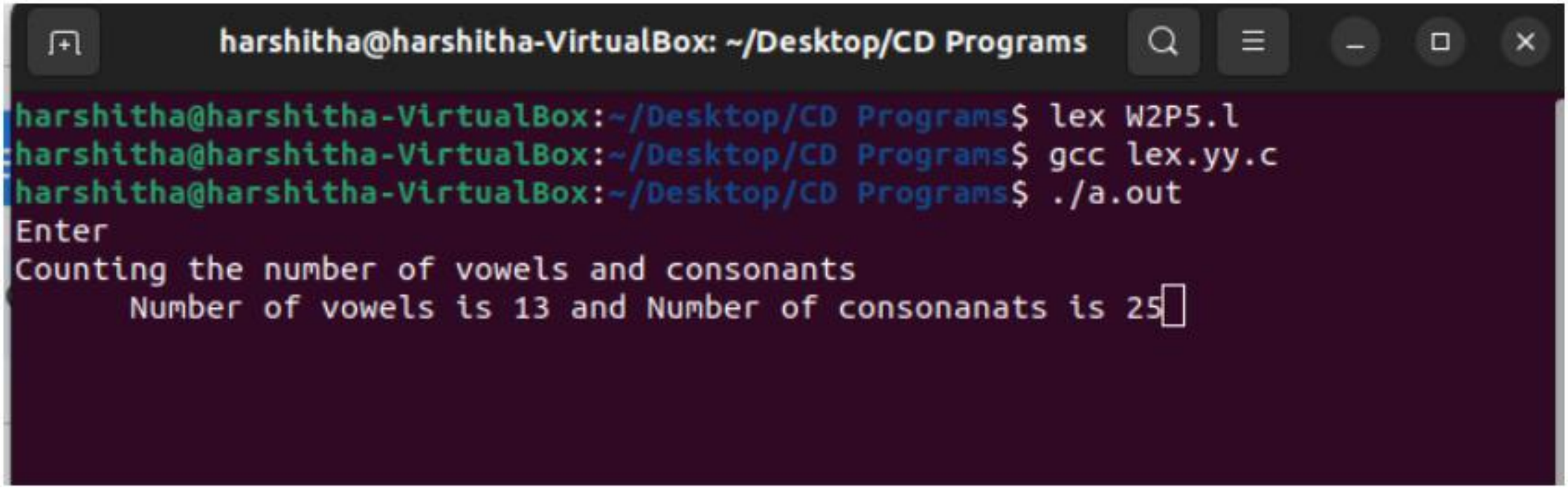
```
printf ("Enter the input file : \n");
scanf ("%s", fname);
yyin = fopen ("fname", "r");
yylex();
return 0;
}
```

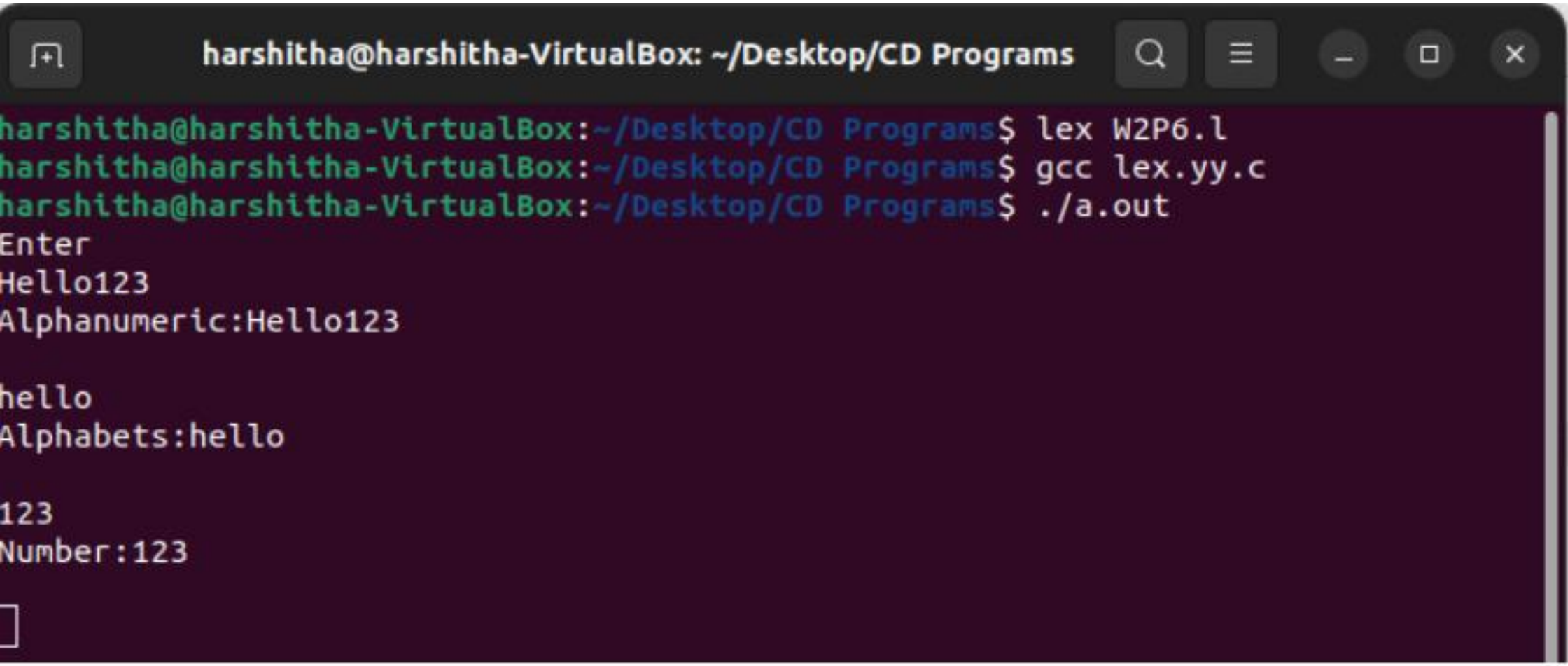

harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W2P1.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter
9.5
Floating point number:9.5
H
Character:H
HII
Variable:HII
6
Integer:6
6hi4.2k
Integer:6
Variable:hi4
Floating point number:.2
Character:k

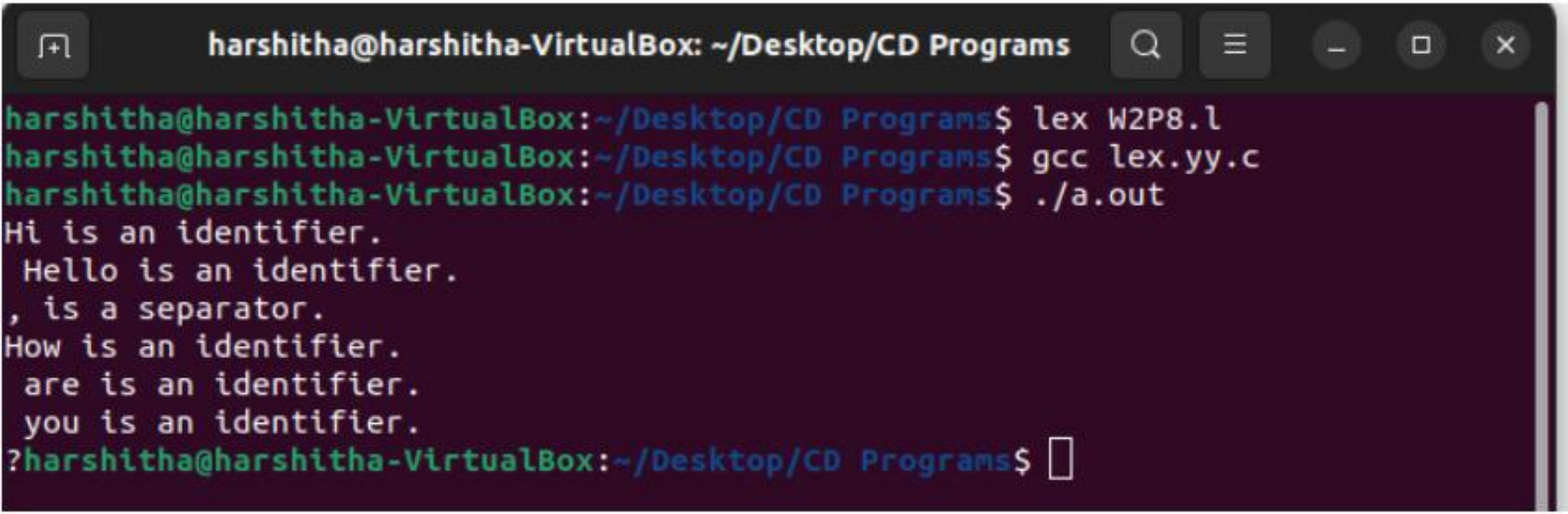

harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W2P2.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter
Hello
Consonant:H
Vowel:e
Consonant:l
Consonant:l
Vowel:o

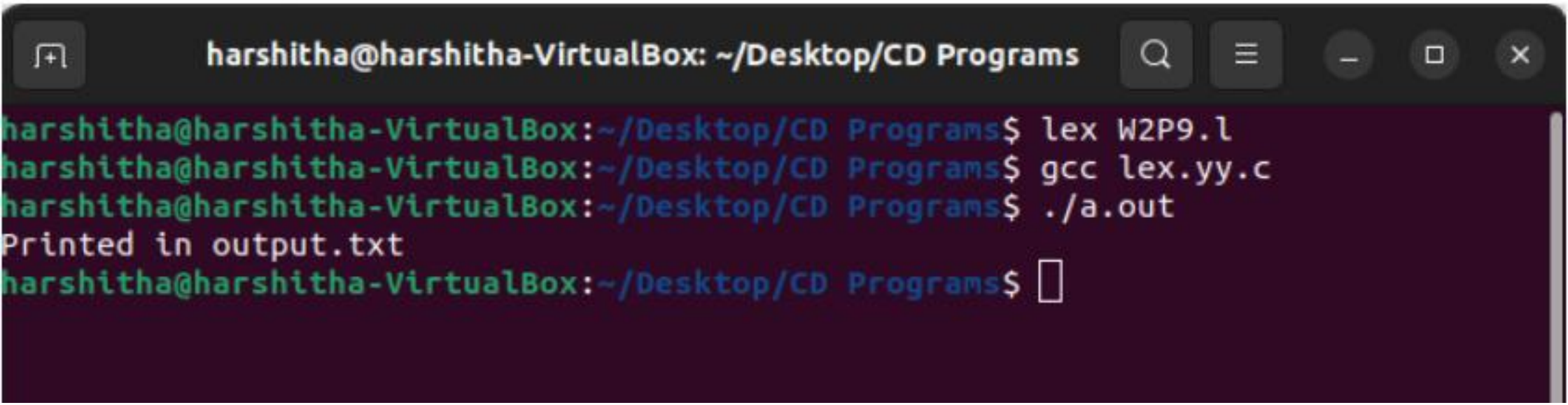

harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W2P3.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter
Hello123
Stream of Characters:Hello
Digit:123


harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W2P4.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter a sentence
This is a lex program to count the number of words
Total number of words in the given sentence is 11


harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W2P5.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter
Counting the number of vowels and consonants
Number of vowels is 13 and Number of consonanats is 25


harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W2P6.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter
Hello123
Alphanumeric:Hello123
hello
Alphabets:hello
123
Number:123


harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W2P8.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Hi is an identifier.
Hello is an identifier.
, is a separator.
How is an identifier.
are is an identifier.
you is an identifier.
?harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$


harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W2P9.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Printed in output.txt
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$

output.txt

```
1 Hi is an identifier.
2  Hello is an identifier.
3 , is a separator.
4 How is an identifier.
5  are is an identifier.
6  you is an identifier.
7 ?
```

24/11/23

Date / /
Page

LAB - 3

1. **Program 1:**

Read and input sentence and check if it is compound or simple. If a sentence has the words - and, or, but, because, if, then, nevertheless, then it is compound, else it is simple.

```
%{
    #include<stdio.h>
    int flag = 0;
%}
%%
and |or| but |because| if | then |nevertheless
                                {flag = 1;}
.;
\n {if (flag == 1)
        printf ("yes, compound");
    else
        printf ("simple");
    }
%%
int yywrap()
{
}
int main()
{
  printf ("Enter a string");
  yylex();
  return 0;
}
```

Output:

```
Bring book and pen
It is a compound sentence
Hello
It is a simple sentence.
```

Program 2:

Write a program in LEX to recognise floating point numbers.

```
%option noyywrap
%{
   #include<stdio.h>
%}
%%
[-+]?[0-9]+[.][0-9]+ {printf("It is a floating
                        point number\n"); }
[-+]?[0-9]+ {printf("It is not a floating point
                     number\n"); }
%%
void main()
 {
   printf("Enter a number");
   yylex();
 }
```

Output:

```
Enter a number: -23.6
It is a floating point number.
45
It is not a floating point number.
```

Program 3

Write a program to read an input sentence and to check if the sentence begins with English articles (A, a, AN, An, THE, The).

```
%option noyywrap
%{
   #include<stdio.h>
   int a = 0;
%}
%%
[A|a|AN|An|THE|The][a-zA-Z]*. {a=1;}
\n {return 0;}
%%
void main()
{
  printf("Enter a sentence \n");
  a=0;
  yylex();
  if(a==1)
  {
    printf("It starts with article");
  }
  else
  {
    printf("It does not start with article");
  }
}
```

Output:

An apple

It starts with article.

Program 4:
Write a program to check if the input sentences ends with any of the punctuation marks (?, ., !)

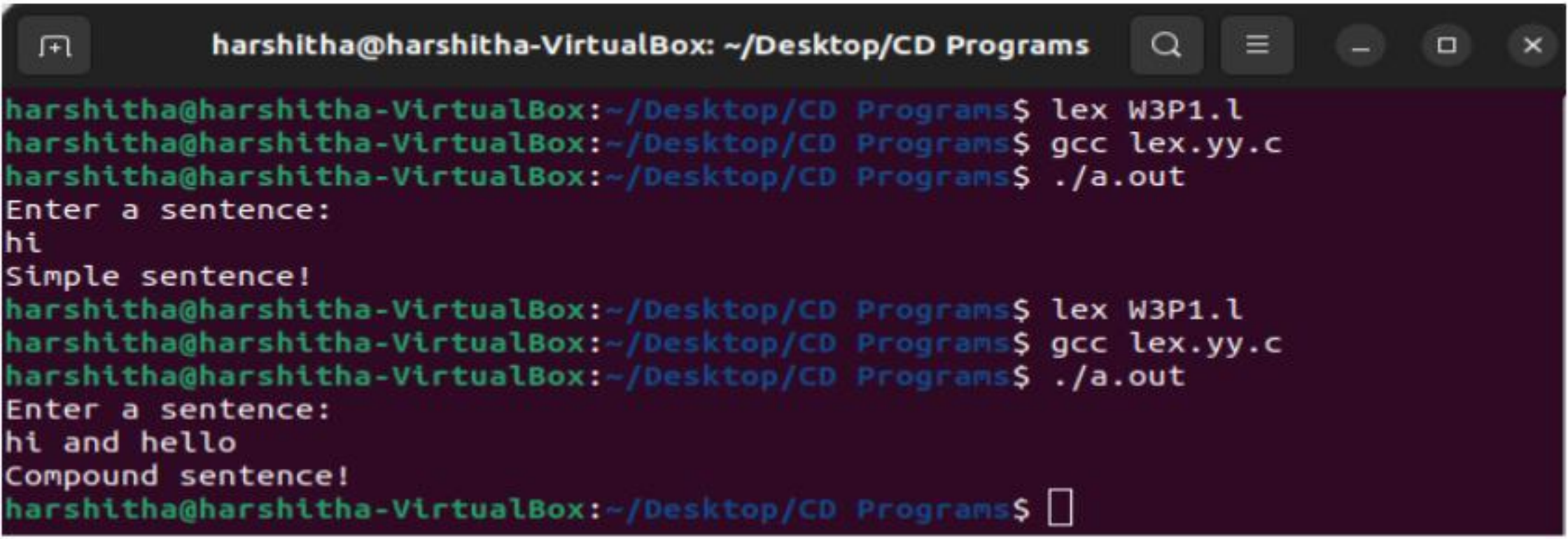
```
%{
[a-zA-Z]+[?|.|!] {a=1;}
\n {return 0;}
%}
```

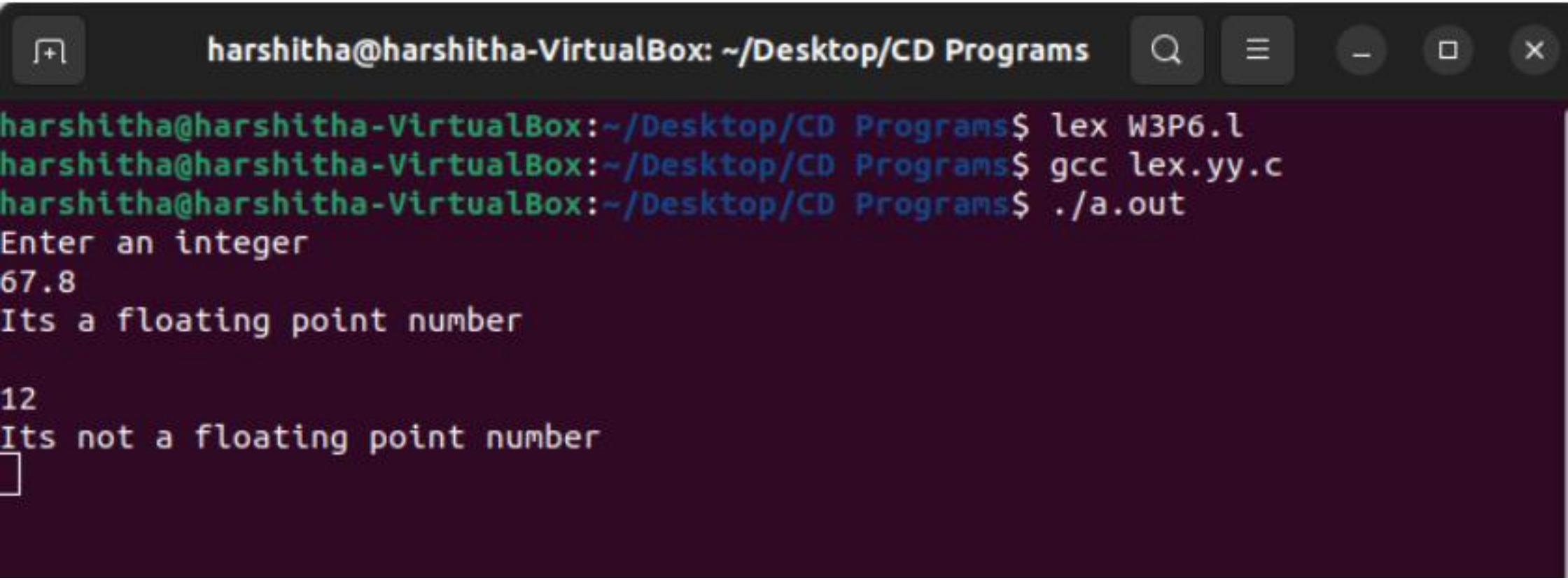
Output:
Hi Hello
It does not end with punctuation
Hello Hi?
It ends with punctuation.

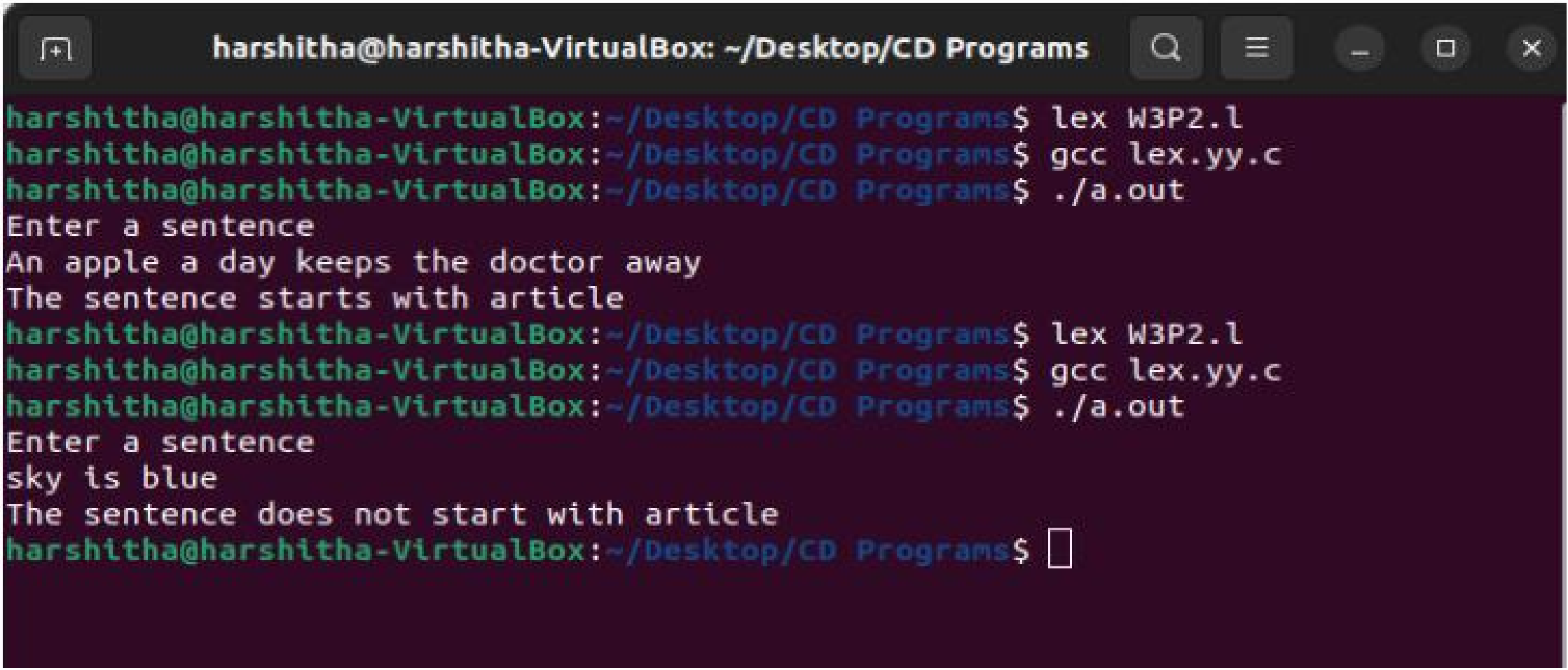
Program 5:
Write a program to read and check if the user entered number is signed or unsigned.

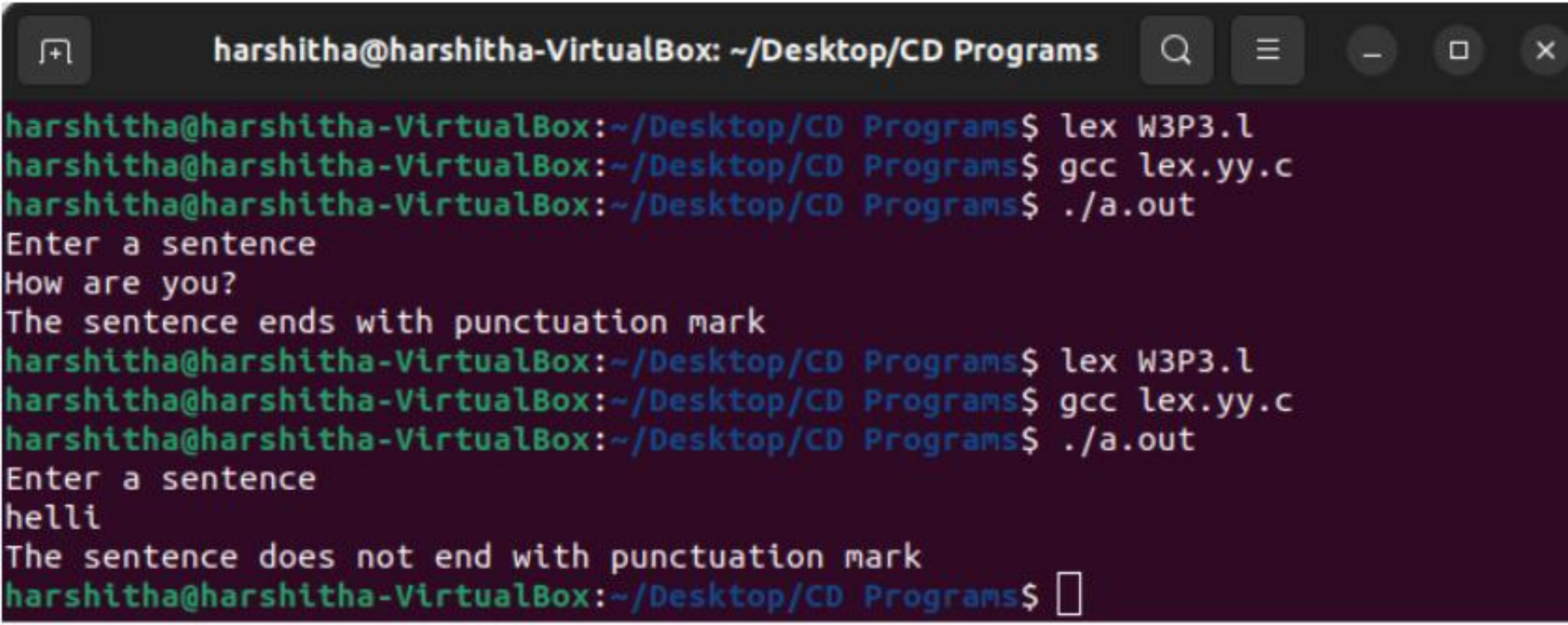
```
%{
[+|-][0-9]+ {printf("It is signed integer");}
[0-9]+ {printf("It is unsigned integer");}
%}
```

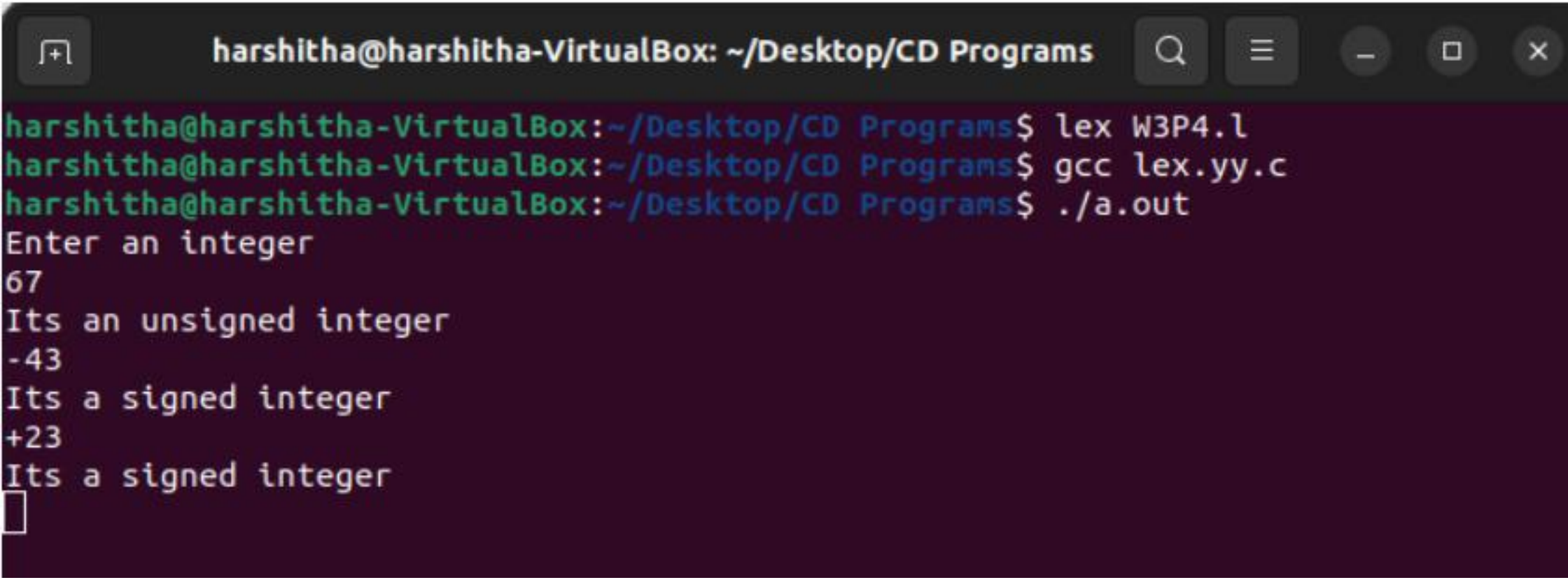
Output:
+23
It is signed integer
23
It is unsigned integer.


harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W3P1.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter a sentence:
hi
Simple sentence!
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W3P1.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter a sentence:
hi and hello
Compound sentence!
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$


harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W3P6.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter an integer
67.8
Its a floating point number
12
Its not a floating point number


harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W3P2.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter a sentence
An apple a day keeps the doctor away
The sentence starts with article
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W3P2.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter a sentence
sky is blue
The sentence does not start with article
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$


harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W3P3.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter a sentence
How are you?
The sentence ends with punctuation mark
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W3P3.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter a sentence
helli
The sentence does not end with punctuation mark
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$


harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W3P4.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter an integer
67
Its an unsigned integer
-43
Its a signed integer
+23
Its a signed integer

11/22/23

Date / /
Page

LAB-4

Program 1:

Write a LEX program that copies a file, replacing each non-empty sequence of white space by a single blank.

```
%{
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
char string[200];
%}

%%
[\n] {fprintf(yyout, "%s\n", string);
      string[0] = '\0'; }
[ ]* {fprintf(yyout, "%s", string);
      string[0] = '\0'; fprint(yyout, "%s", " ");
. strcat(string, yytext)
<<EOF>> {fprintf(yyout, "%s", string); return 0;}
%%

int main()
{
  extern FILE *yyin, *yyout;
  char filename[100];
  printf("This program is going copy a file,
          replacing each nonempty sequence of
          white space by a single blank! \n
          Enter the name of the file to copy \n");
  scanf("%s", filename);
  yyin = fopen(filename, "r");
```

```
    if (yyin == NULL){
        printf ("cannot open file %s \n", filename);
        exit(0);
    }
    printf(" Enter the name of the file to open
                for writing \t");
    scanf ("%s", filename);
    yyout = fopen (filename, "w");
    if (yyout == NULL)
    {
        printf ("cannot open file %s \n", filename);
        exit(1);
    }
    yylex();
}

int yywrap (void)
{
    return 1;
}
```

2. Write a LEX program to recognise the following tokens over the alphabets {0, 1, .. 9}

a) The set of all string ending in 00.

```
%{
    #include<stdio.h>
    #include<stdlib.h>
%}
%%
[0-9]*[00] { printf ("Ending with 00 \n"); }
.*  { printf ("Does not end with 00 \n"); }
%%
```

```
int yywrap()
{
}
int main()
{
  yylex();
  return 0;
}
```

OUTPUT:

2300

Ends with 00.

(b) The set of all strings with three consecutive 222's.

```
%%
[0-9]*[222][0-9]* {printf("Has consecutive 222");}
.* {printf("Does not have consecutive 222");}
%%
```

OUTPUT:

1232221

Has consecutive 222

Date __/__/__

Page _____

(d) The set of all strings beginning with a 1 which, interpreted as the binary representation of an integer, is congruent to zero modulo 5.

```
%{
1[01]* { for (i = yyleng-1; i>=0; i--)
         {
           value = value + (yytext[i]-48)*
                                    pow(2,j)
           j++;
         }
         if (value %. 5 == 0)
           {
             flag = 1;
           }
  [\n] return 0;
%}

int main()
 {
   yylex();
   if (flag == 1)
     {
       printf("success \n");
     }
   else
     {
       printf("fail \n");
     }
   return 0;
 }
```

(e) The set of all strings such that the 10th symbol from the end is 1.

digits [0-9]

%%

{digits}*1{digits}{digits}{digits}{digits}{digits}{digits}{digits}{digits}{digits}{digits}

{printf("%s 10th symbol from right end is 1", yytext);}

.* {printf("%s 10th symbol from right end is not 1", yytext);}

%%

OUTPUT:

2134567892 2

10th Symbol from right is 1.

(f) The set of all four digits numbers whose sum is 9

%%

{digits}{digits}{digits}{digits} {for(i = yyleng-1; i>=0; i--)

{

value += (yytext[i] - 48);

}

if (value == 9)

{

flag = 1;

}

[\n] return 0;

%%

int main()

{

yylex();

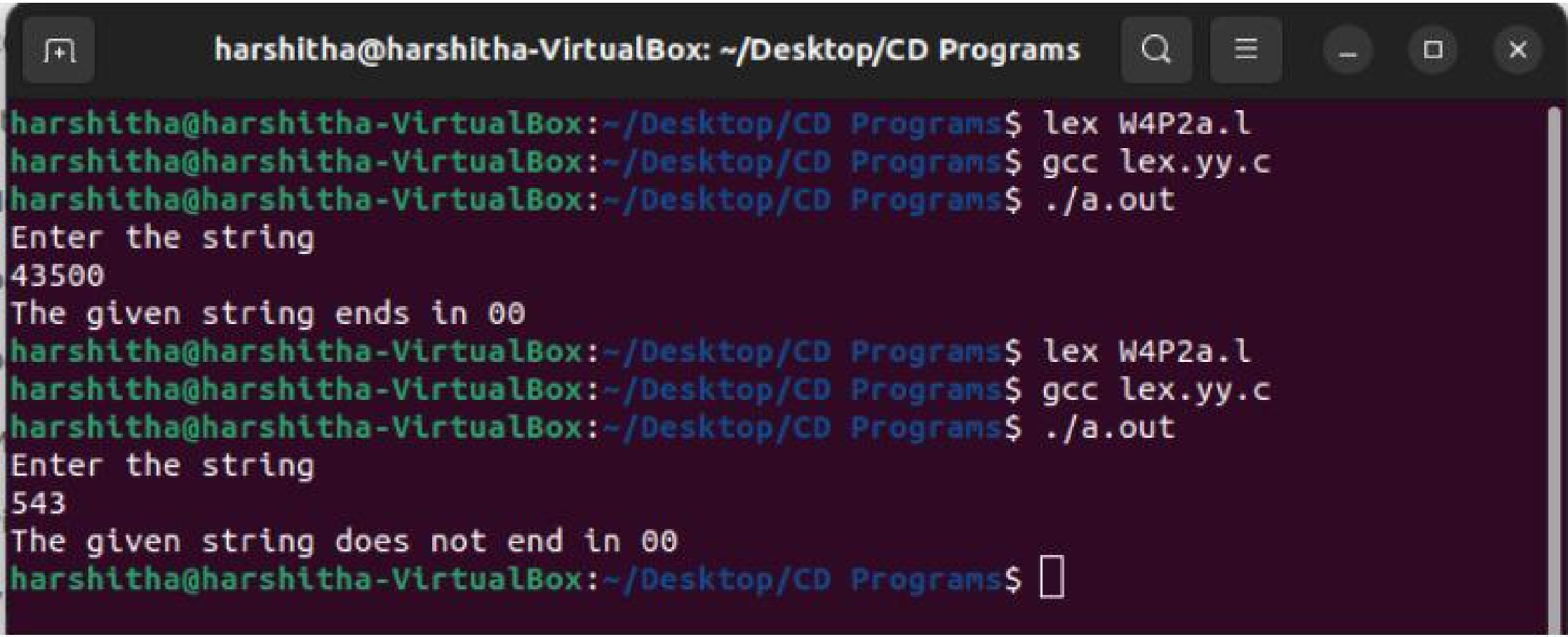
```
if (flag == 1)
{
    printf ("success \n");
}
else
{
    printf ("fail \n");
}
return 0;
}
```

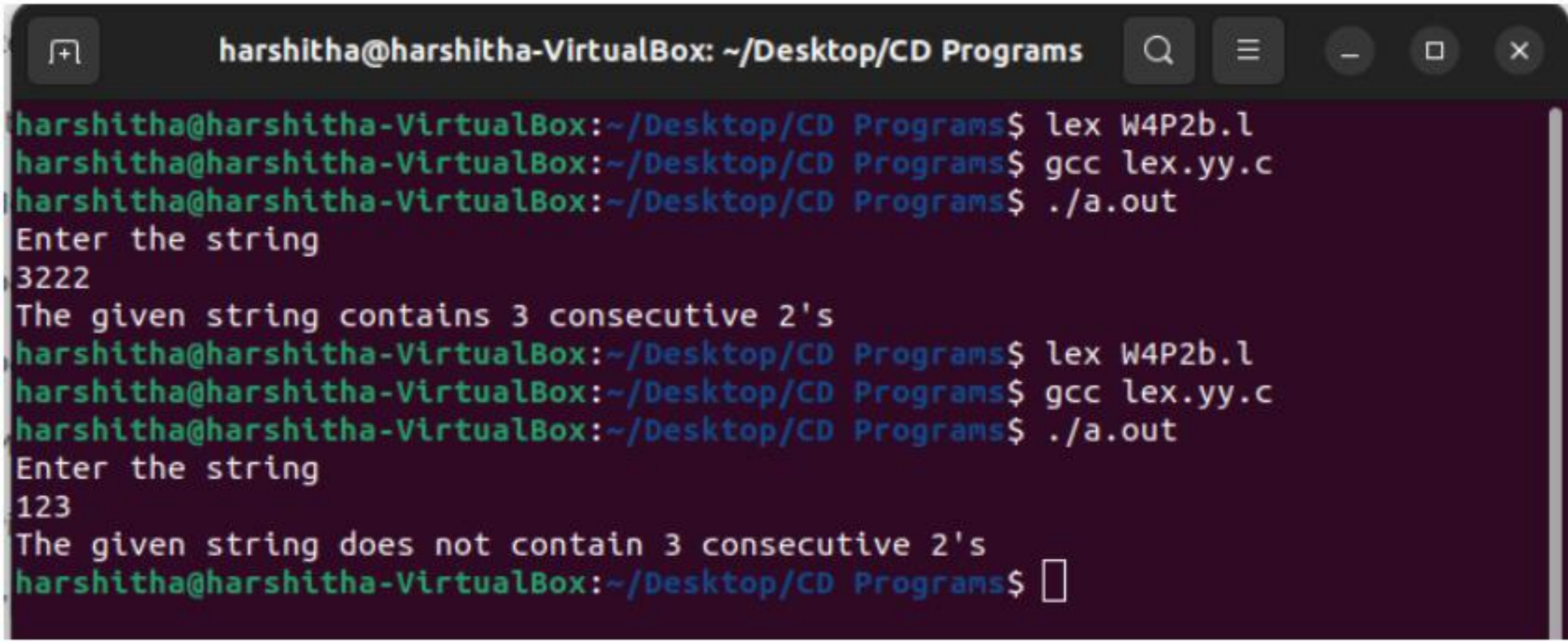
*input.txt

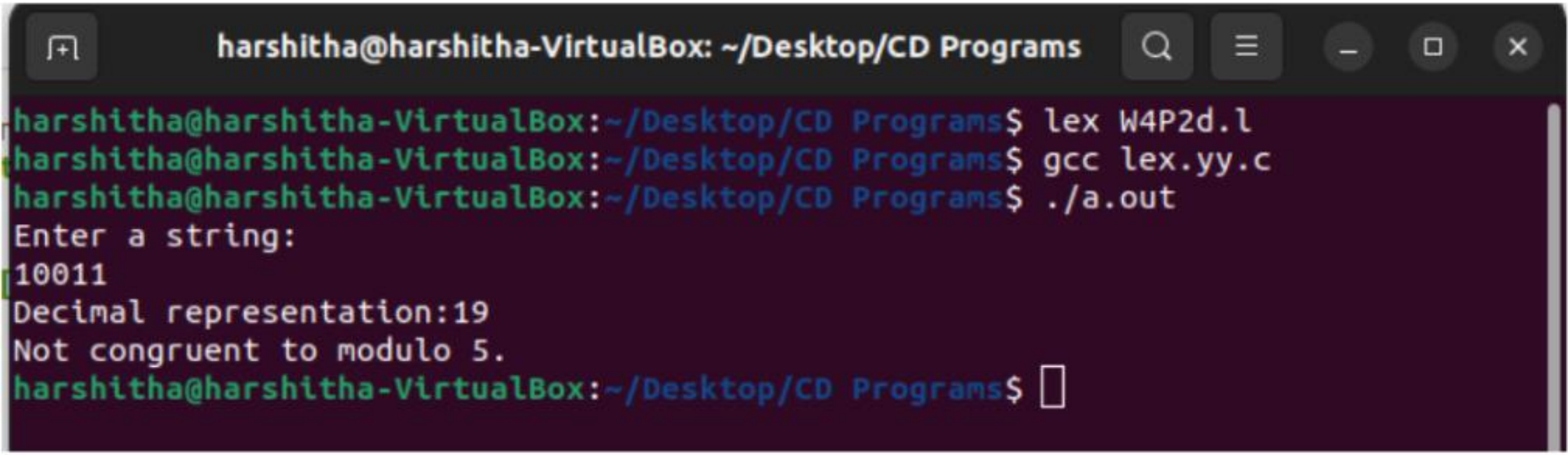
```
1 Hi,      Hello,   How are you?
```

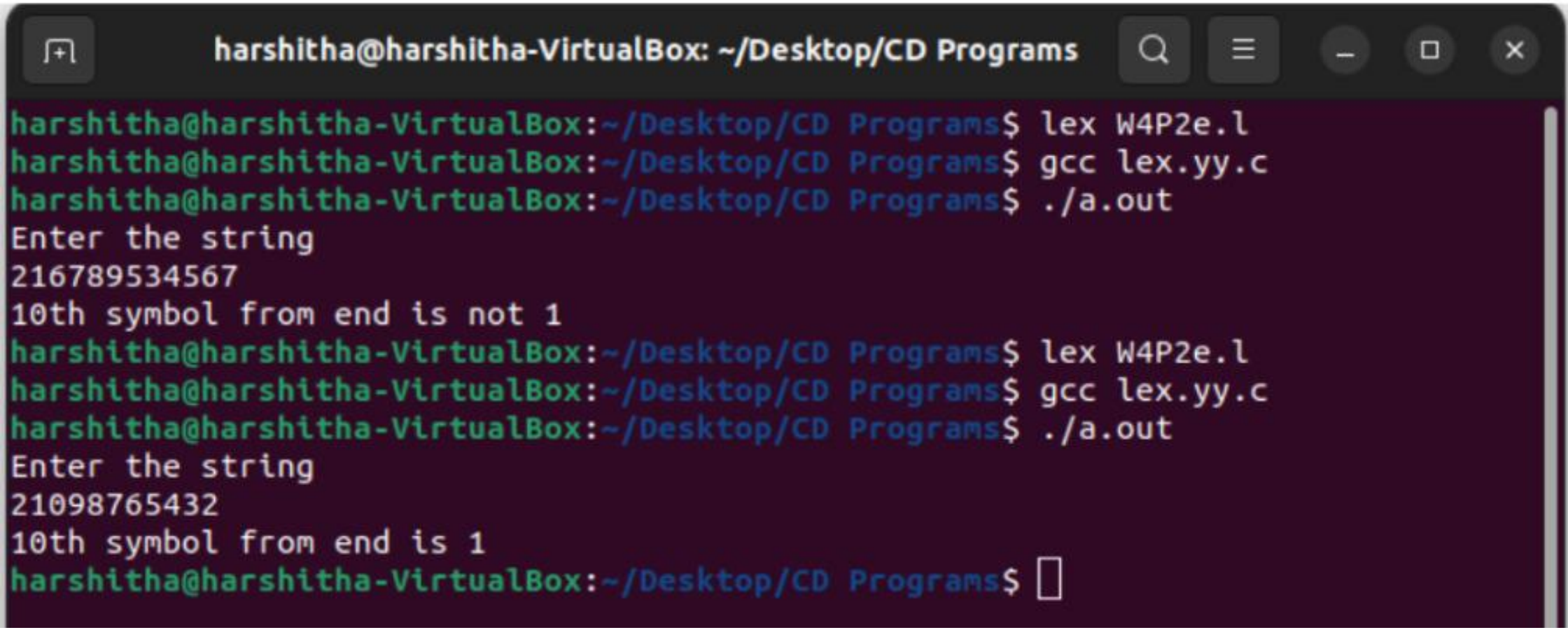
output.txt

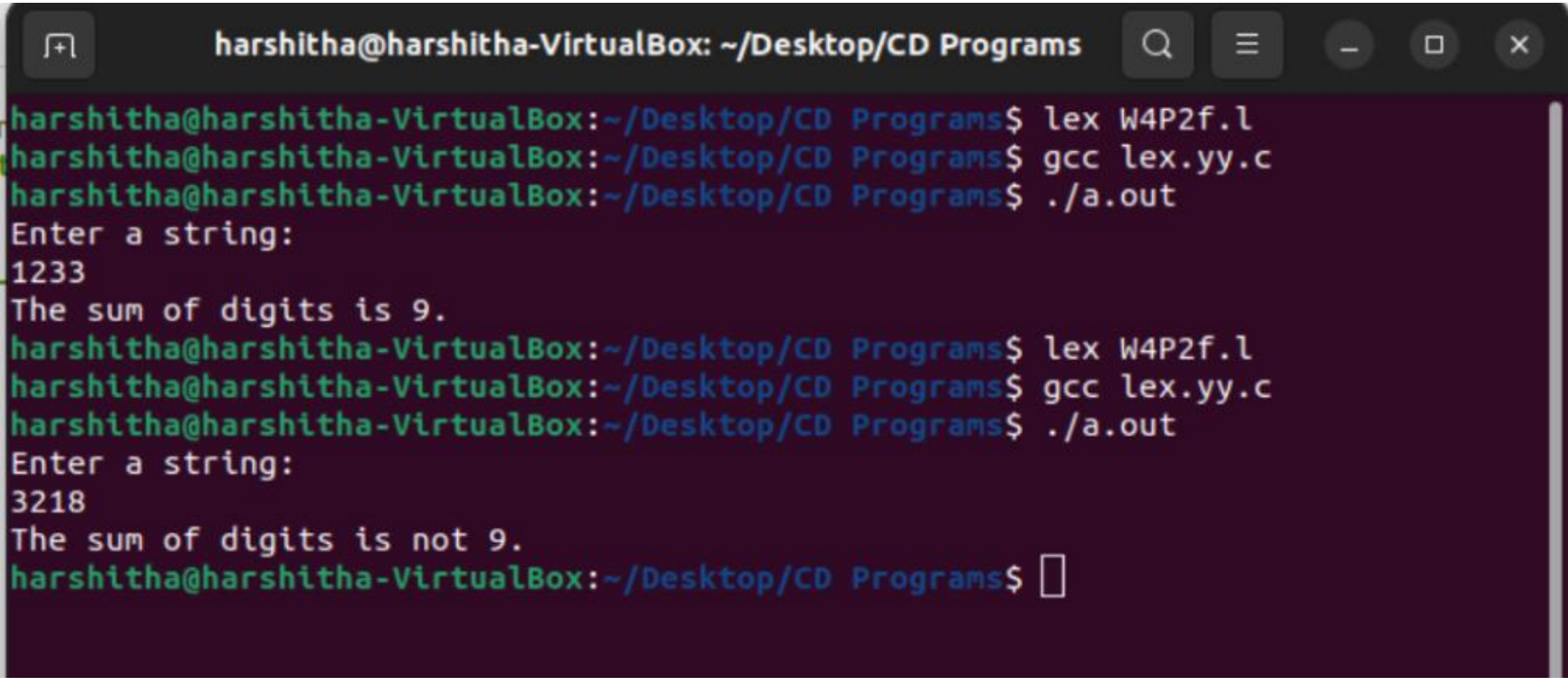
```
1 Hi, Hello, How are you?
```


harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W4P2a.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter the string
43500
The given string ends in 00
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W4P2a.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter the string
543
The given string does not end in 00
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$


harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W4P2b.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter the string
3222
The given string contains 3 consecutive 2's
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W4P2b.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter the string
123
The given string does not contain 3 consecutive 2's
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$


harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W4P2d.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter a string:
10011
Decimal representation:19
Not congruent to modulo 5.
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$


harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W4P2e.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter the string
216789534567
10th symbol from end is not 1
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W4P2e.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter the string
21098765432
10th symbol from end is 1
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$


harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W4P2f.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter a string:
1233
The sum of digits is 9.
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex W4P2f.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter a string:
3218
The sum of digits is not 9.
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$

12/12/23

Date / /
Page

LAB-5

Program:
Write a C program to design Lexical Analyser to recognise any five keywords, identifiers, numbers, operators and punctuations

```
bool isPunctuator (char ch)
{
    if (ch == ' ' || ch == '+' || ch == '-' || ch == '*' ||
        ch == '/' || ch == ',' || ch == ';' || ch == '>' ||
        ch == '<' || ch == '=' || ch == '(' || ch == ')' ||
        ch == '[' || ch == ']' || ch == '{' || ch == '}')
    {
        return true;
    }
    return false;
}
bool isOperator (char ch)
{
    if (ch == '+' || ch == '-' || ch == '*' || ch == '/'
        || ch == '>' || ch == '<' || ch == '=')
    {
        return true;
    }
    return false;
}

bool isIdentifier (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' || isPunctuator (str[0]) == true
```

```
        {
            return false;
        }
    return true;
}

bool isKeyword (char * str)
{
    if (!strcmp(str, "if") || !strcmp(str, "else")
        || !strcmp(str, "while") || !strcmp(str, "do")
        || !strcmp(str, "break"))
        return true;
    return false;
}

bool isNumber (char ch)
{
    if (ch == '0' || ch == '1' || ch == '2' || ch == '3' ||
        ch == '4' || ch == '5' || ch == '6' || ch == '7'
        || ch == '8' || ch == '9')
        return true;
    return false;
}

void parse (char * str)
{
    int left = 0, right = 0;
    int len = strlen(str);

    while (right <= len)
    {
        if (!isPunctuator(str[right]))
            right++;
```

```
if (isPunctuator(str[right]) || right == len)
{
    char * subStr = subString(str, left,
                              right-1);
    if (right!=len && isOperator(str[right]))
        printf("'%c' IS AN OPERATOR \n",
               str[right]);
    else if (isKeyword(subStr))
        printf("'%s' IS A KEYWORD \n", subStr);
    else if (isNumber(subStr))
        printf("'%s' IS A NUMBER \n", subStr);
    else if (isIdentifier(subStr))
        printf("'%s' IS A VALID IDENTIFIER
               \n", subStr);
    else
        printf("'%s' IS NOT A VALID IDENTIFIER"
               , subStr);
    left = ++right;
  }
}
free(subString);
}

int main() {
    char str[100] = "int a = b + 1c;";
    parse(str);
    return 0;
}
```

Date __/__/__

Page _____

Output:

'int' is a KEYWORD
'a' is a VALID IDENTIFIER
'=' is an OPERATOR
',' is a VALID OPERATOR
'b' is a NUMBER
'+' is an OPERATOR
'' IS A VALID IDENTIFIER
'c' IS NOT A VALID IDENTIFIER

Sun 18/12/23

```
Keyword: if
Operator: (
Identifier: x
Operator: >
Number: 0
Operator: )
Operator: {
Keyword: return
Identifier: x
Punctuation:;
Operator: }
Keyword: else
Operator: {
Keyword: return
Operator: -x
Punctuation:;
Operator: }
```

08/01/24

Date __/__/__
Page _____

LAB-6

Write a program to perform Recursive Descent Parsing on the following grammar.

S → cAd , A → ab | a.

```
#include <stdio.h>
#include <stdlib.h>

char input[100];
int ind = 0;
void match (char expected)
 {
   if (input[ind] == expected)
      ind++;
 }
void A();
void S()
 {
   match('c');
   A();
   match('d');
 }

void A()
 {
   if (input[ind] == 'a')
     {
       match('a');
       if (input[ind] == 'b')
         match('b');
     }
   else
```

```
    {
        printf("Parsing failed \n", ind);
        exit(1);
    }
}
int main()
{
    printf("enter the input string \n");
    scanf("%s", input);
    s();
    if (input[ind] == '$')
        printf("Parsing successful\n");
    else
        printf("Parsing failed. Extra characters
                found \n");
    return 0;
}
```

OUTPUT:

```
Enter the input string.
cabd$
Parsing successful.
```

```
main.c: In function 'A':
main.c:33:16: warning: too many arguments for format [-Wformat-extra-args]
   33 |         printf("Parsing failed.\n", ind);
      |                ^~~~~~~~~~~~~~~~~~~
Enter the input string:
cabd$
Hello
Parsing successful.


...Program finished with exit code 0
Press ENTER to exit console.
```

08/01/24

Date / /
Page

LAB-1

Design a suitable grammar for evaluation of arithmetic expression having + and - operators. + has least priority and its left associative. - has higher priority and is right associate.

w7.yl

```
%{
#include "y.tab.h"
%}
%%
[0-9]+ {yylval = atoi(yytext); return NUM;}
[\t] ;
\n return 0;
. return yytext[0];
%%
int yywrap()
{
}
```

w7.y

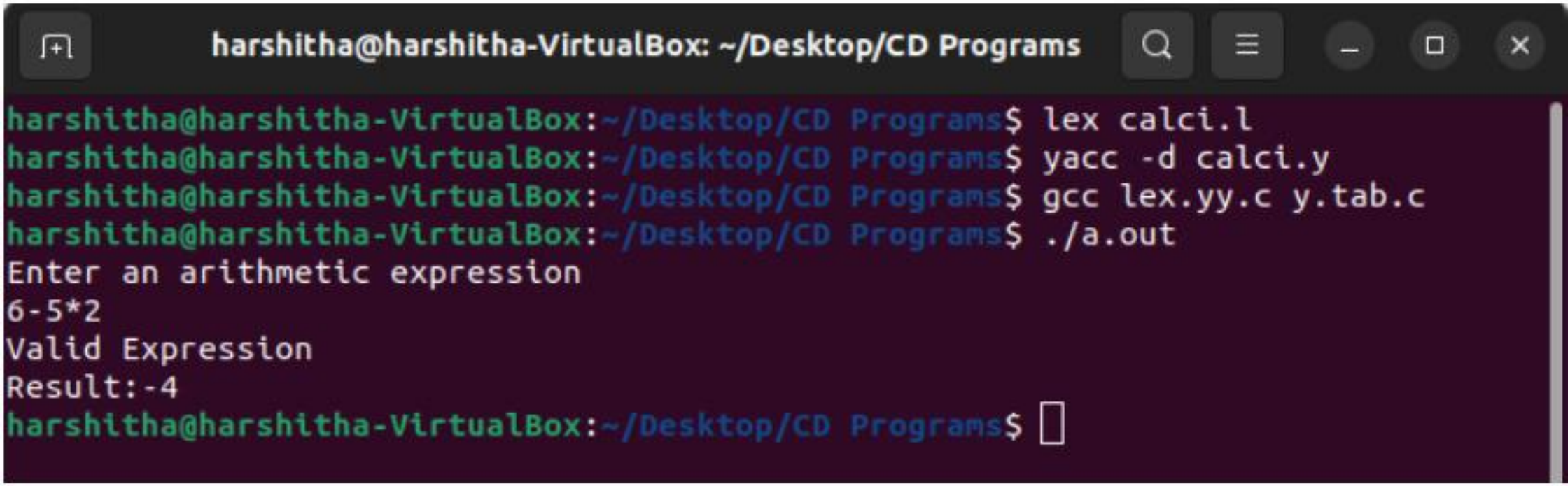
```
%{
 #include <stdio.h>
%}

%token NUM
%left '+'
%right '-'
```

```
%%
expr: e {printf("Valid expression \n");
         printf("Result: %d \n", $$);
         return 0;}
e: e '+' e      { $$ = $1 + $3; }
 | e '-' e      { $$ = $1 - $3; }
 | NUM          { $$ = $1; }
 ;
%%

int main()
{
  printf("Enter an arithmetic expression");
  yyparse();
  return 0;
}

int yyerror()
{
  printf("\n Invalid expression \n");
  return 0;
}
```

Sun 8/11/2

OUTPUT:

```
Enter an arithmetic expression
5 + 6 - 3 - 6
Valid expression
Result: 14
```


harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex calci.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ yacc -d calci.y
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c y.tab.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter an arithmetic expression
6-5*2
Valid Expression
Result:-4
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$

29/01/24

Date __/__/__
Page

LAB - 8

Syntax Tree:

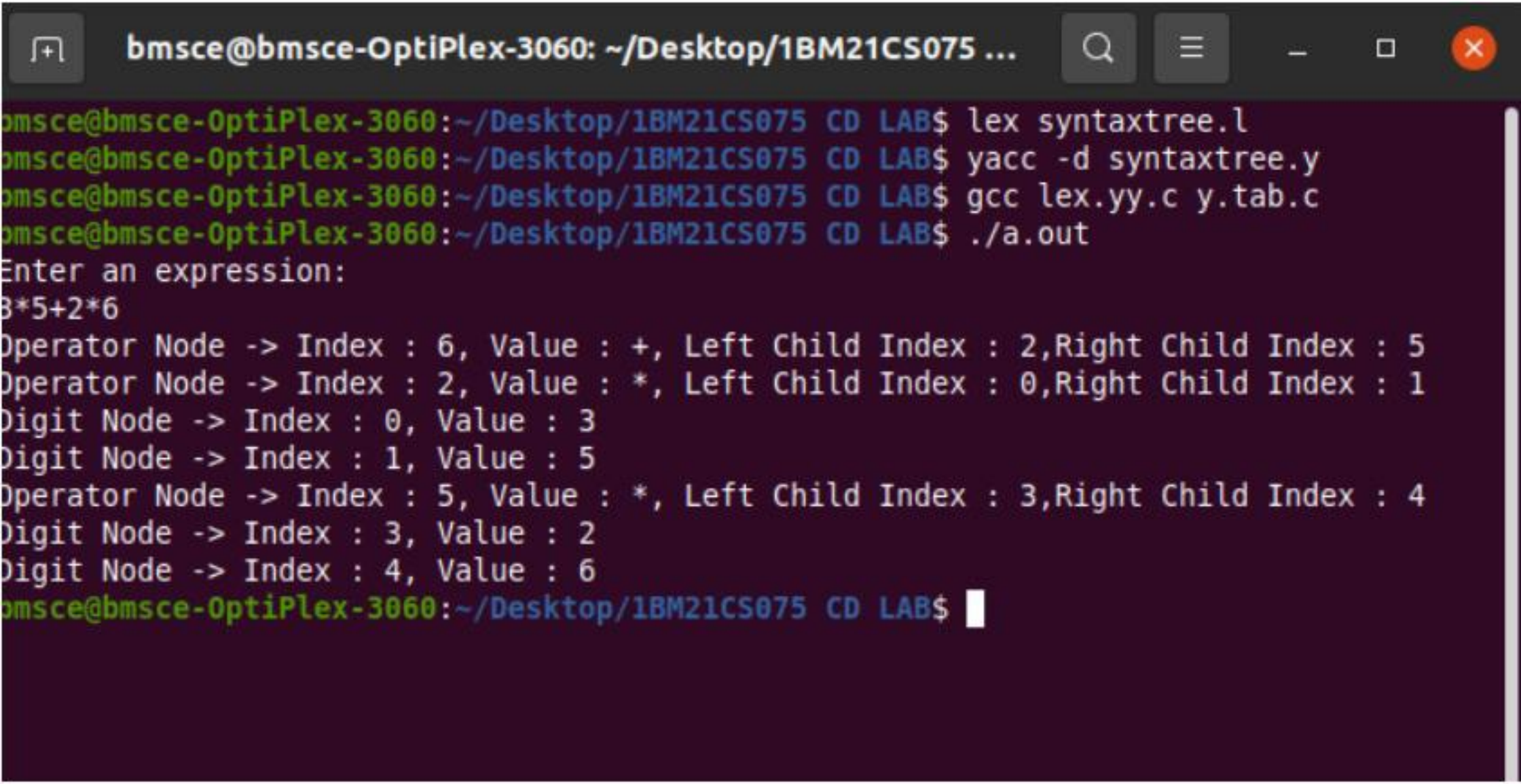
p3.l

```
%{
#include "y.tab.h"
extern int yylval;
%}
%.%.
[0-9]+ {yylval = atoi(yytext); return digit;}
[\t];
[\n] return 0;
. return yytext[0];
%.%.
int yywrap()
{
}
```

p1.y

```
%{
#include <math.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct tree_node
{
  char val[10];
  int lc;
  int rc;
};
int ind;
struct tree_node syn_tree[100];
```

```
void my_print_tree (int cur_ind);
int mknode (int lc, int rc, char val[10]);
%}
%token digit
%%
S: E { my_print_tree ($1); }
;
E: E '+' T  { $$ = mknode ($1, $3, "+"); ; }
 | T        { $$ = $1; }
 ;
T: T '*' F  { $$ = mknode ($1, $3, "*"); ; }
 ;
F: '(' E ')'  { $$ = $2; }
 | digit { char buf[10]; sprintf (buf, "%d",
           yylval); $$ = mknode (-1, -1, buf); }
%%
int main()
{
  ind = 0;
  printf ("Enter an expression \n");
  yyparse();
  return 0;
}
int yyerror()
{
  printf ("NTTW Error \n");
}
int mknode (int lc, int rc, char val[10])
{
  strcpy (syn_tree[ind].val, val);
  syn_tree[ind].lc = lc;
  syn_tree[ind].rc = rc
  ind++;
  return ind-1;
}
```


bmsce@bmsce-OptiPlex-3060: ~/Desktop/1BM21CS075 ...
bmsce@bmsce-OptiPlex-3060:~/Desktop/1BM21CS075 CD LAB$ lex syntaxtree.l
bmsce@bmsce-OptiPlex-3060:~/Desktop/1BM21CS075 CD LAB$ yacc -d syntaxtree.y
bmsce@bmsce-OptiPlex-3060:~/Desktop/1BM21CS075 CD LAB$ gcc lex.yy.c y.tab.c
bmsce@bmsce-OptiPlex-3060:~/Desktop/1BM21CS075 CD LAB$ ./a.out
Enter an expression:
3*5+2*6
Operator Node -> Index : 6, Value : +, Left Child Index : 2,Right Child Index : 5
Operator Node -> Index : 2, Value : *, Left Child Index : 0,Right Child Index : 1
Digit Node -> Index : 0, Value : 3
Digit Node -> Index : 1, Value : 5
Operator Node -> Index : 5, Value : *, Left Child Index : 3,Right Child Index : 4
Digit Node -> Index : 3, Value : 2
Digit Node -> Index : 4, Value : 6
bmsce@bmsce-OptiPlex-3060:~/Desktop/1BM21CS075 CD LAB$

29/01/24

Date
Page

LAB-8

Infix to Postfix Expression

infixtopostfix.l

```
%{
#include <stdio.h>
#include <stdlib.h>
#include "y.tab.h"
extern int yylval;
%}
%%
[0-9]+ { yylval = atoi(yytext); return num; }
[\t];
\n { return 0; }
. { return yytext[0]; }
%%
int yywrap()
{
}
```

infixtopostfix.y

```
%{
#include <stdio.h>
#include <stdlib.h>
int yyerror(const char *s);
int yylex(void);
%}
%token num
%left '+' '-'
%left '*' '/'
%left ')'
%left '('
```

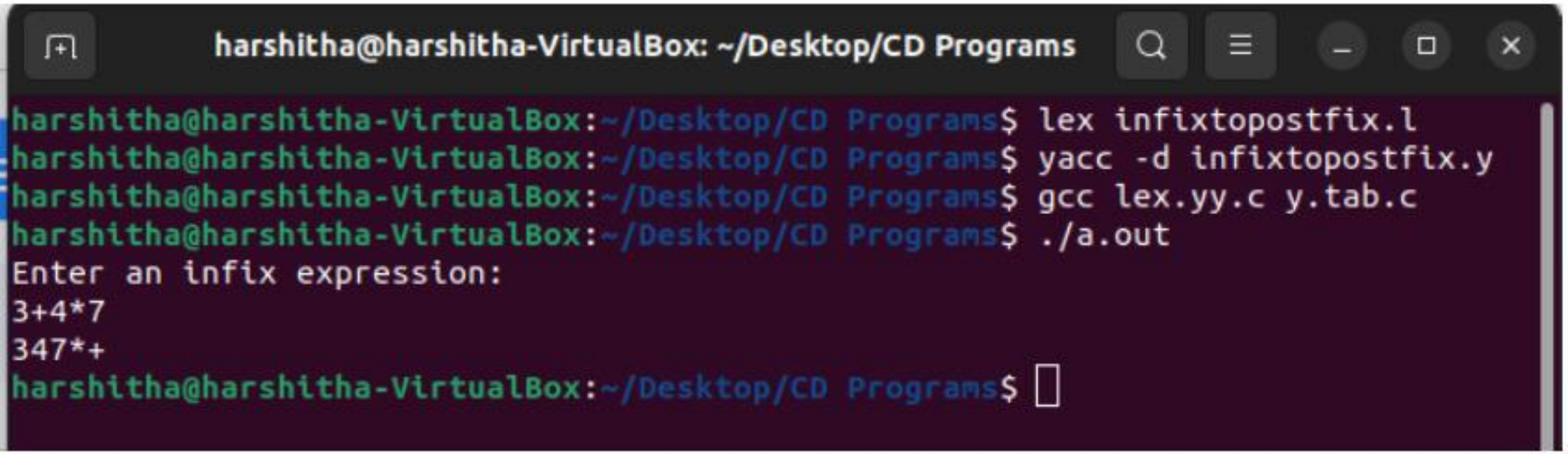
```
%right '^'
%%
s: e {printf("\n");}
 ;
e: e '+' t {printf("+");}
 | e '-' t {printf("-");}
 | t
 ;
t: t '*' h {printf("*");}
 | t '/' h {printf("/");}
 | h
 ;
h: f '^' h {printf("^");}
 | f
 ;
f: '(' e ')'
 | num {printf("%d", $1);}
 ;
%%

void main()
{
 printf("Enter an infix expression:\n");
 yyparse();
}
int yyerror(const char *s)
{
 printf("Invalid infix expression\n");
 return 0;
}
```

OUTPUT:

Enter an infix expression

3+4*7

347*+


harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex infixtopostfix.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ yacc -d infixtopostfix.y
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c y.tab.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter an infix expression:
3+4*7
347*+
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$

29/01/24

Write a YACC program to recognise the grammar ($a^n b$, $n >= 5$)

stringmatch.l

```
%{
#include <stdio.h>
#include <stdlib.h>
#include "y.tab.h"
extern int yylval;
%}
%%
[aA] { yylval = yytext[0]; return A; }
[bB] { yylval = yytext[0]; return B; }
\n { return NL; }
. { return yytext[0]; }
%%
int yywrap()
{
  return 1;
}
```

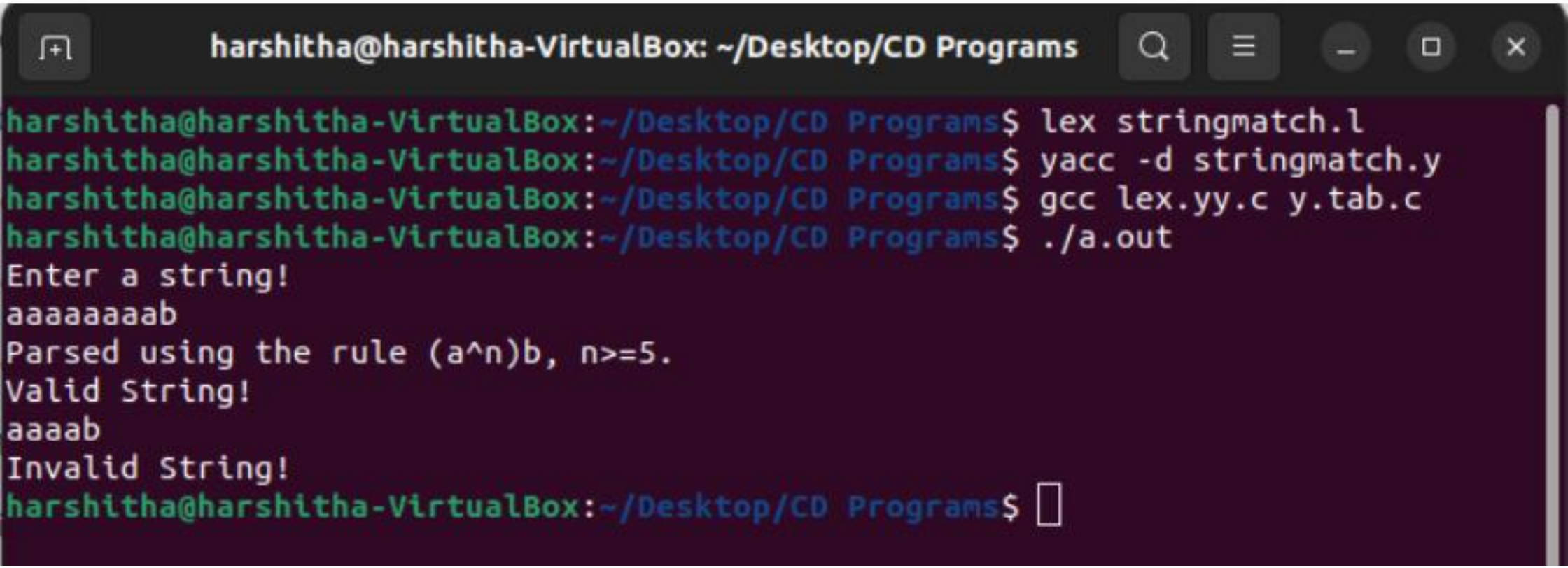
stringmatch.y

```
%{
#include <stdio.h>
#include <stdlib.h>
int yyerror(char *s);
int yylex(void);
%}
%token A
%token B
%token NL
%%
```

```
%%
Stmt : A A A A A S B NL {printf ("Parsed using
       the rule (a^n)b, n>=5 \n valid string");}
     ;
S : S A
  | ;
%%
void main()
{
  printf ("Enter a string! \n");
  yyparse();
}
int yyerror (char *s)
{
  printf ("Invalid String! \n");
  return 0;
}
```

Output:

```
Enter a String:
aaaaaab
Parsed using the rule (a^n)b, n>=5
valid string!
```


harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex stringmatch.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ yacc -d stringmatch.y
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c y.tab.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter a string!
aaaaaaaaab
Parsed using the rule (a^n)b, n>=5.
Valid String!
aaaab
Invalid String!
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$

29/01/24

Date
Page

LAB-9

Use YACC to generate 3-Address Code for a given expression

p1.l

```
a [0-9]+
a [a-zA-Z]+
%{
#include <stdio.h>
#include <stdlib.h>
#include "y.tab.h"
extern int yylval;
extern char iden[20];
%}
%%
{d} {yylval = atoi(yytext); return digit;}
{a} {strcpy(iden, yytext); yylval = 1; return id;}
[\t] {;}
\n return 0;
. return yytext[0];
%%
```

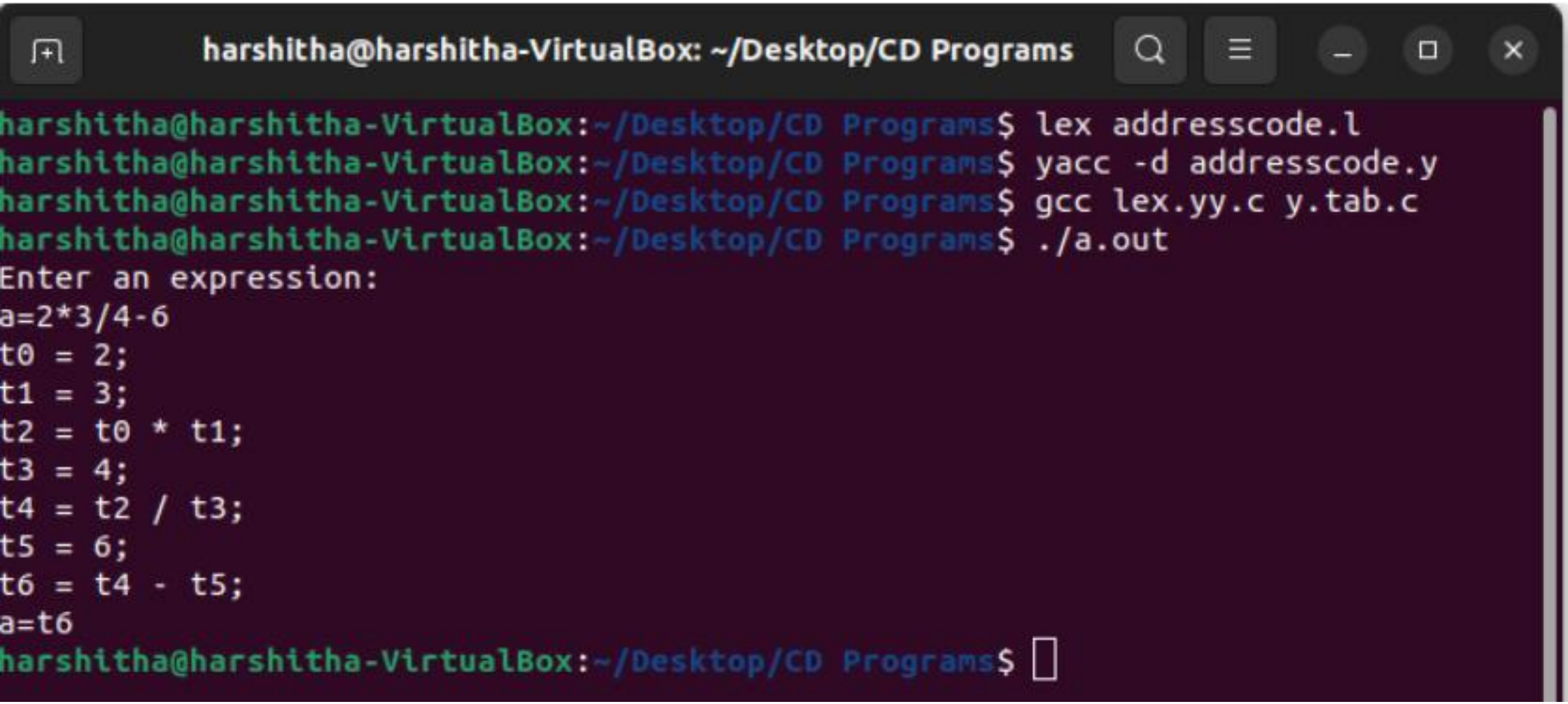
~~p1.y~~

```
%{
#include <stdio.h>
#include <math.h>
#include <ctype.h>
int var_cnt = 0;
char iden[20];
%}
%token id
%token digit
```

```
%{
S : id '=' E {printf("%s = t%d\n", iden, var_cnt);}
E : E '+' T {$$ = var_cnt; var_cnt++; printf
             ("t%d = t%d + t%d;\n", $$,
  | E '-' T {$$ = var_cnt; var_cnt++; printf
             ("t%d = t%d - t%d;\n", $$, $1, $3);}
  }
  | T {$$ = $1;}
  ;
T : T '*' F {$$ = var_cnt; var_cnt++;
             printf("t%d = t%d * t%d;",
             $$, $1, $3);}
  | T '/' F {$$ = var_cnt; var_cnt++; printf
             ("t%d = t%d / t%d;\n", $$, $1, $3);}
  | F {$$ = $1;}
F : P '^' F {$$ = var_cnt; var_cnt++; printf
             ("t%d = t%d ^ t%d;\n", $$, $1, $3);}
  | P {$$ = $1;}
P : '(' E ')' {$$ = $2;}
  | digit {$$ = var_cnt; var_cnt++; printf
             ("t%d = %d;\n", $$, $1);}
  ;
%}
```

Output:

```
Enter an expression
a = 2 + 3 * 6
t0 = 2
t1 = 3
t2 = 6
t3 = t1 * t2;
t4 = t0 + t3;
a = t4
```


harshitha@harshitha-VirtualBox: ~/Desktop/CD Programs
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ lex addresscode.l
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ yacc -d addresscode.y
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ gcc lex.yy.c y.tab.c
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$ ./a.out
Enter an expression:
a=2*3/4-6
t0 = 2;
t1 = 3;
t2 = t0 * t1;
t3 = 4;
t4 = t2 / t3;
t5 = 6;
t6 = t4 - t5;
a=t6
harshitha@harshitha-VirtualBox:~/Desktop/CD Programs$