LAB 1
DISCUSS ABOUT THE FLEX , ITS SYNTAX, BLOCK DIAGRAM AND ITS WORKING

INTRODUCTION TO FLEX :

Flex is an open source program designed to automatically and quickly generate scanners, also known as tokenizers, which recognize lexical patterns in text. Flex is an acronym that stands for "fast lexical analyzer generator. " It is a free alternative to Lex, the standard lexical analyzer generator in Unix-based systems. Flex was originally written in the C programming language by Vern Paxson in 1987.

SYNTAX OF FLEX :

In the input file, there are 3 sections:

1. **Definition Section:** The definition section contains the declaration of variables, regular definitions, manifest constants. Anything written in this brackets is copied directly to the file **lex.yy.c**
   Syntax:

   %{

   // Definitions

   %}

2. **Rules Section:** The rules section contains a series of rules in the form: *pattern action* and pattern must be unintended and action begin on the same line in {} brackets.
   Syntax:

   %%

   pattern action

   %%

3. **User Code Section:** This section contains C statements and additional functions. We can also compile these functions separately and load with the lexical analyzer.

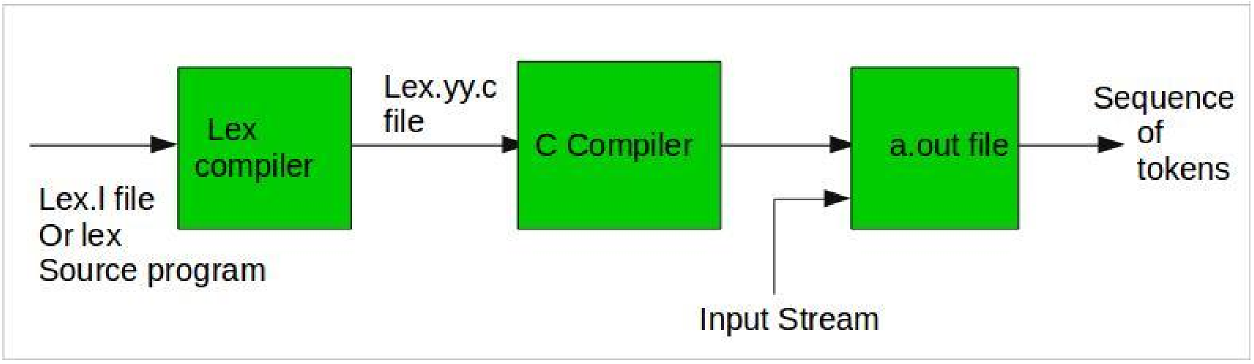
   Syntax:

   %{

   // Definitions

```
%}

%%
Rules
%%
User code section
```

BLOCK DIAGRAM AND ITS WORKING:



Given image describes how the Flex is used:

Step 1: An input file describes the lexical analyzer to be generated named lex.l is written in lex language. The lex compiler transforms lex.l to C program, in a file that is always named lex.yy.c.

Step 2: The C compiler compile lex.yy.c file into an executable file called a.out.

Step 3: The output file a.out take a stream of input characters and produce a stream of tokens.

HOW TO RUN THE PROGRAM:

To run the program, it should be first saved with the extension **.l or .lex**. Run the below commands on terminal in order to run the program file.

Step 1: lex filename.l or lex filename.lex depending on the extension file is saved with

Step 2: gcc lex.yy.c

Step 3: ./a.out

Step 4: Provide the input to program in case it is required

LAB 2

WRITE A FLEX PROGRAM TO READ A C-FILE AS AN INPUT AND PRODUCE AN OUTPUT NEW C-FILE REPLACING ALL FLOAT KEYBOARD FROM INPUT FILE TO DOUBLE KEYWORDS IN OUTPUT FILE

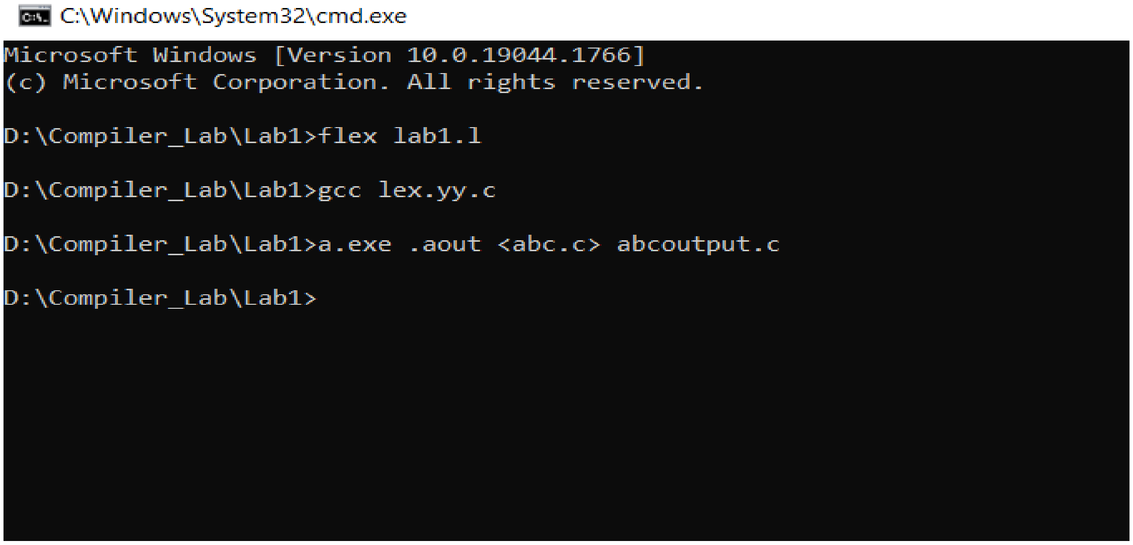
//SOURCE CODE (lab1.l)

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

%%
"float" {fprintf(yyout,"double"); }
\.|[\n] {fprintf(yyout,yytext);}
%%

int yywrap()
{
return 1;
}

int main(int argc, char *argv[])
{
yyin=fopen(argv[1],"r");
yyout=fopen(argv[2],"r+");
yylex();
}
```

//INPUT FILE(abc.c)

```
#include<stdio.h>
#include<conio.h>
int main()
{
float a;
float b;
float c;
float d;
printf("THis is float");
getch();
return 0;
}
```

//OUTPUT



```
Microsoft Windows [Version 10.0.19044.1766]
(c) Microsoft Corporation. All rights reserved.

D:\Compiler_Lab\Lab1>flex lab1.l

D:\Compiler_Lab\Lab1>gcc lex.yy.c

D:\Compiler_Lab\Lab1>a.exe .aout <abc.c> abcoutput.c

D:\Compiler_Lab\Lab1>
```

//OUTPUT FILE (abcoutput.c)

```
#include<stdio.h>
#include<conio.h>
int main()
{

double a;
double b;
double c;
double d;
printf("THis is double");
getch();
return 0;
}
```

LAB 3

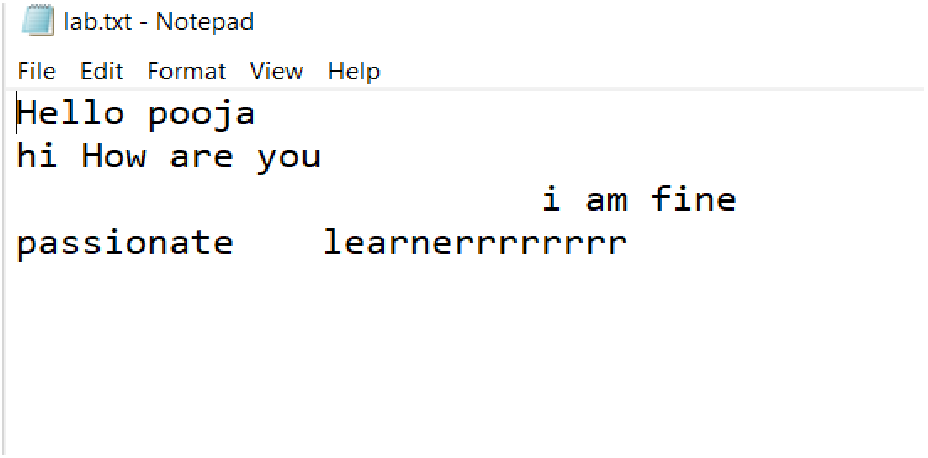
WRITE A C-PROGRAM TO COUNT NUMBER OF CHARACTERS, WHITE SPACES, TABS AND LINES IN A GIVEN FILE

//SOURCE CODE

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<ctype.h>
int main()
{
FILE *fp1;
char ch;
int space=0,lines=1,tabs=0,chars=0;
fp1=fopen("lab.txt","r");
while(!feof(fp1))
{
ch=fgetc(fp1);
if(isgraph(ch)||ch==' '||ch=='\n'||ch=='\t')
{
chars++;
}
if(ch==' ')
{
space++;
}
if(ch=='\n')
{
lines++;
}
if(ch=='\t')
{
Tabs++;
}
}
printf("spaces\t-->%d\n",space);
printf("lines\t-->%d\n",lines);printf("tabs\t-->%d\n",tabs);printf("chars\t-->%d\n",chars);
}
```

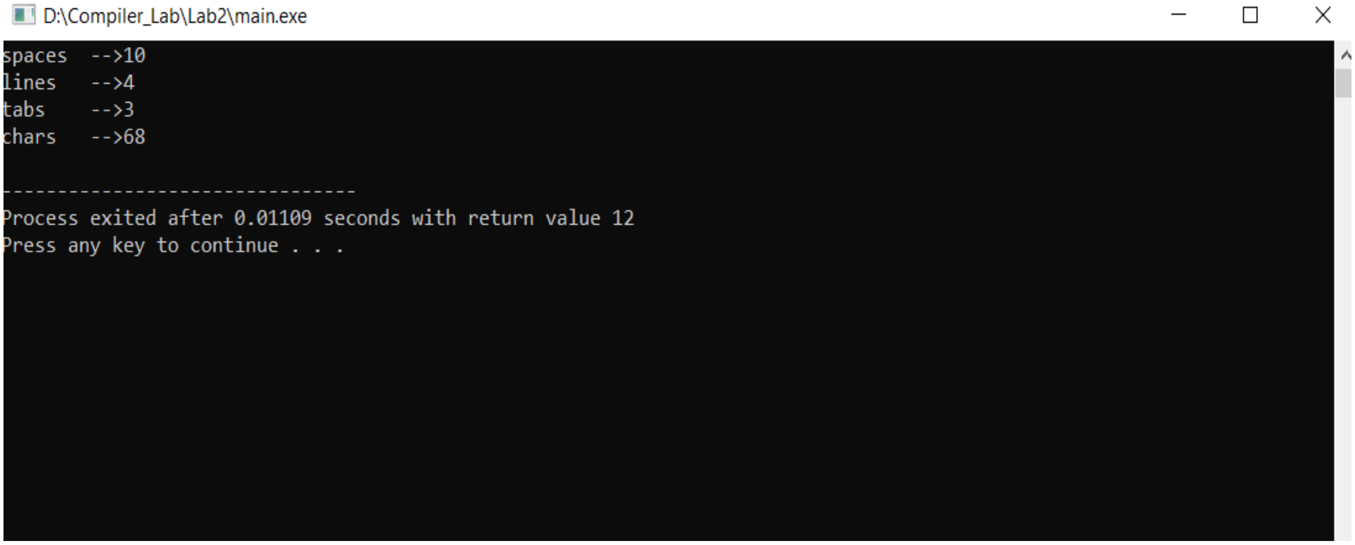
//INPUT FILE

(lab.txt)



```
Hello pooja
hi How are you
                        i am fine
passionate    learnerrrrrrrr
```

//OUTPUT



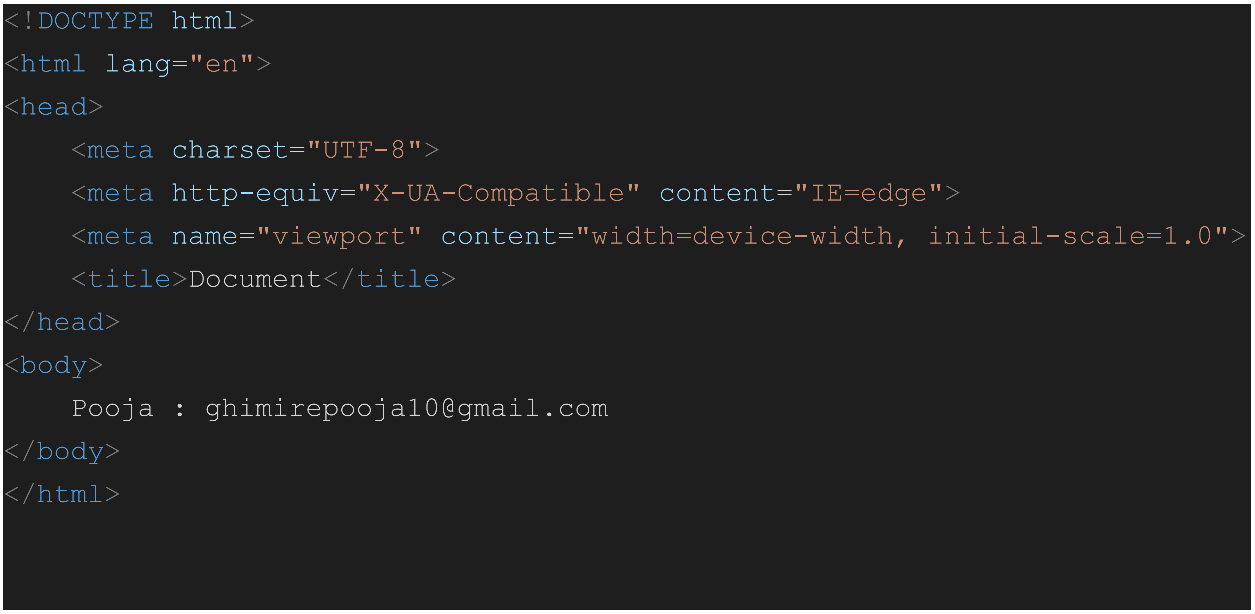
```
spaces  -->10
lines   -->4
tabs    -->3
chars   -->68

--------------------------------
Process exited after 0.01109 seconds with return value 12
Press any key to continue . . .
```

LAB 4
WRITE A FLEX PROGRAM THAT READS HTML FILE AS AN INPUT AND OUTPUT NEW HTML FILE BY APPENDING MAIL-TO: TAG TO ALL THE MAIL ADDRESS IN INPUT HTML

//SOURCE CODE

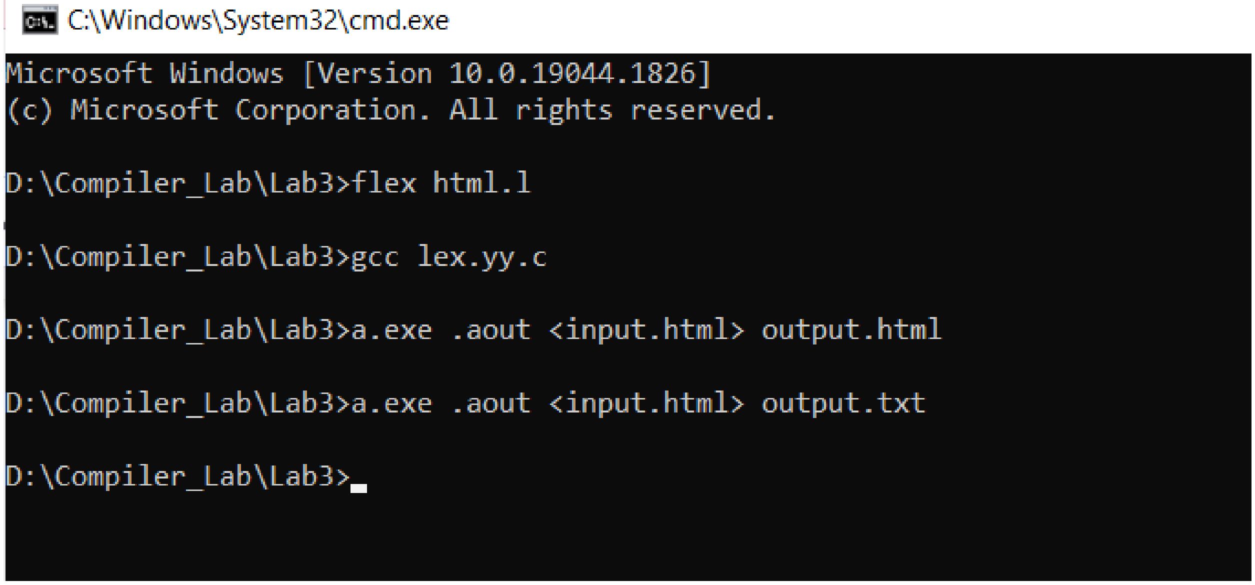
```
%{
#include<stdio.h>
char *first = "<a href=\"mailto:";
char *second = "\">mail</a>";
char name[80];
char buf[80];
%}
digit [0-9]
str [a-z]
symb [._]
delim [\t]*
%%
{delim}{str}+({symb}|{digit}+|{str}+)*\@{str}+\.{str}+{delim} {
snprintf(buf, sizeof buf,"%s%s%s", first, yytext, second);
fprintf(yyout, buf);
}
%%
int yywrap()
{
return 1;
}
int main(int argc, char *argv[])
{
if(argc!=2)
{
printf("Usage: <./a.out> <sourcefile> > <destination file>\n");
exit(0);
}
yyin = fopen(argv[1], "r");
yyout = fopen(argv[2], "w");
yylex();
}
```

//INPUT FILE



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    Pooja : ghimirepooja10@gmail.com
</body>
</html>
```

//OUTPUT



```
Microsoft Windows [Version 10.0.19044.1826]
(c) Microsoft Corporation. All rights reserved.

D:\Compiler_Lab\Lab3>flex html.l

D:\Compiler_Lab\Lab3>gcc lex.yy.c

D:\Compiler_Lab\Lab3>a.exe .aout <input.html> output.html

D:\Compiler_Lab\Lab3>a.exe .aout <input.html> output.txt

D:\Compiler_Lab\Lab3>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    Pooja : <a href="mailto:ghimirepooja10@gmail.com">mail</a>
</body>
</html>
```

LAB 5

WRITE A FLEX PROGRAM THAT READ A FILE CONTAINING THE INTEGER AND FLOAT NUMBERS SEPARATED BY WHITE SPACES AND IDENTIFY THE MAXIMUM, MINIMUM AMONG THEM AND ALSO DISPLAY THE TOTAL SUM OF THE NUMBERS IN A SEPARATE FILE

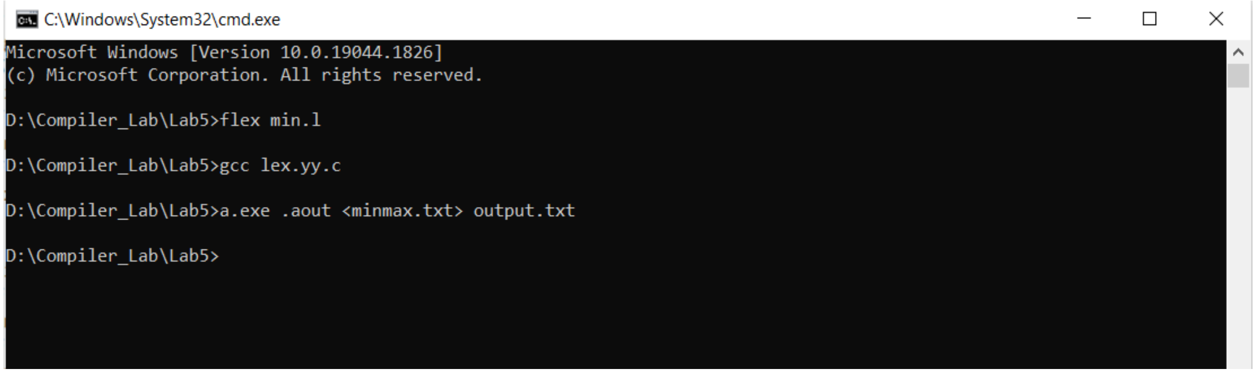
//SOURCE CODE FOR IDENTIFYING MAXIMUM, MINIMUM AMONG NUMBERS

```
%{
#include<stdio.h>
float min = 999, max = 0;
%}
digits [0-9]
%%
{digits}+[\.]{digits}* {
if (atof(yytext) < min)
min = atof(yytext);
if (atof(yytext) > max)
max = atof(yytext);
}
{digits}+{
if (atoi(yytext) < min)
min = (float) atoi(yytext);
if (atoi(yytext) > max)
max = (float) atoi(yytext);
}
%%
int yywrap()
{
return 1;
}
int main(int argc, char *argv[])
{
if(argc!=2)
{
printf("Usage: <./a.out> <sourcefile> > <destination file>");
exit(0);
}
yyin = fopen(argv[1], "r");
yyout = fopen(argv[2], "w");
yylex();
printf("\nMinimum: %f\nMaximum: %f\n", min, max);
}
```

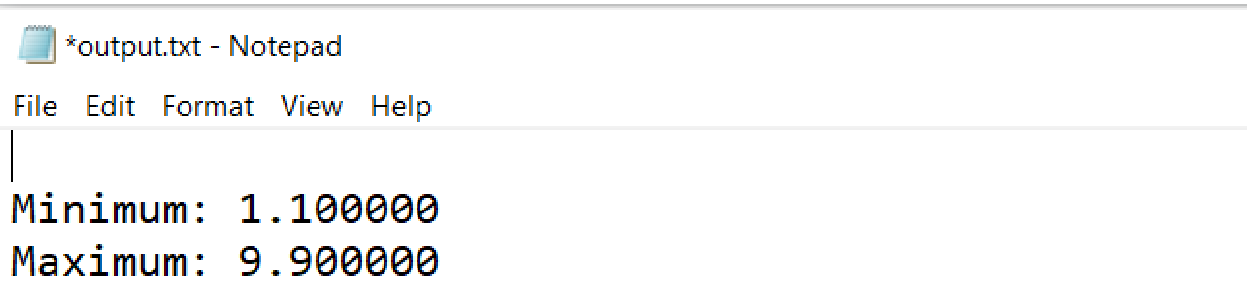
//INPUT FILE

```
1.1
3
5.5
2.8
7
9.9
```

//OUTPUT



```
Microsoft Windows [Version 10.0.19044.1826]
(c) Microsoft Corporation. All rights reserved.

D:\Compiler_Lab\Lab5>flex min.l

D:\Compiler_Lab\Lab5>gcc lex.yy.c

D:\Compiler_Lab\Lab5>a.exe .aout <minmax.txt> output.txt

D:\Compiler_Lab\Lab5>
```



```

Minimum: 1.100000
Maximum: 9.900000
```

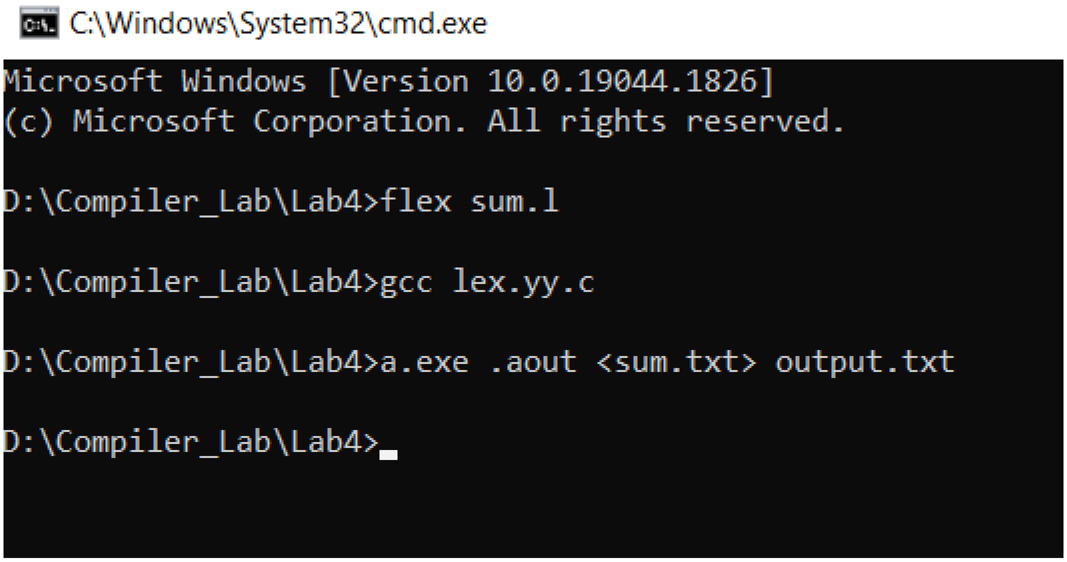
//SOURCE CODE FOR DISPLAYING TOTAL SUM OF ALL NUMBERS

```
%{
#include<stdio.h>
float sum = 0.0;
%}
digits [0-9]
%%
{digits}+[\.]{digits}+ { sum += atof(yytext); }
{digits}+ {sum +=(float) atoi(yytext); }
%%
int yywrap()
{
return 1;
}
int main(int argc, char **argv)
{
if(argc!=2)
{
printf("Error! \n");
exit(0);
}
yyin = fopen(*(argv+1), "r");
yylex();
printf("Sum: %f \n", sum);
}
```

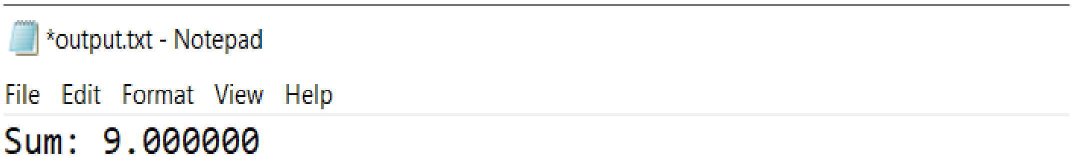
//INPUT FILE

```
1
2.5
2.5
3
```

//OUTPUT



```
Microsoft Windows [Version 10.0.19044.1826]
(c) Microsoft Corporation. All rights reserved.

D:\Compiler_Lab\Lab4>flex sum.l

D:\Compiler_Lab\Lab4>gcc lex.yy.c

D:\Compiler_Lab\Lab4>a.exe .aout <sum.txt> output.txt

D:\Compiler_Lab\Lab4>
```



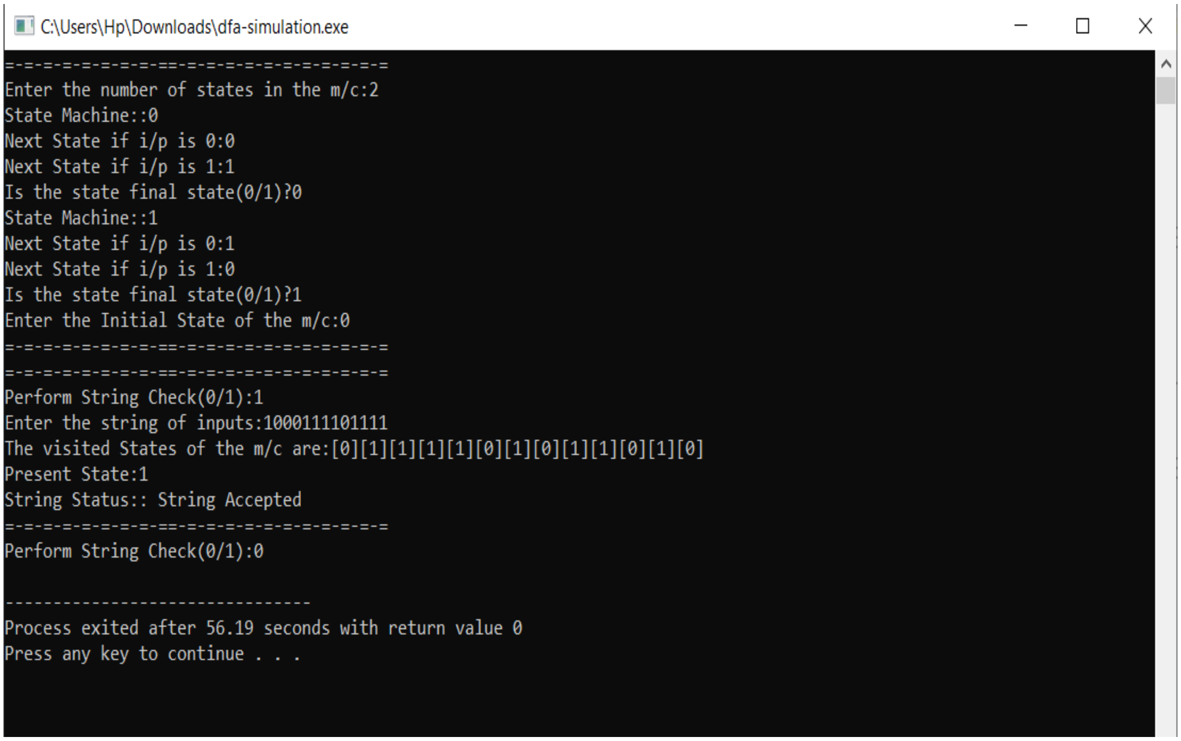
```
Sum: 9.000000
```

LAB 6
WRITE A C PROGRAM FOR THE IMPLEMENTATION OF DETERMINISTIC FINITE AUTOMATA (DFA)

//Source code

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

struct node{
  int id_num;
  int st_val;
  struct node *link0;
  struct node *link1;
};
struct node *start, *q, *ptr;
int vst_arr[100], a[10];
int main(){
  int count, i, posi, j;
  char n[10];

  printf("=-=-=-=-=-=-=-=-==-=-=-=-=-=-=-=-=-=-=-=-=-=\n");
  printf("Enter the number of states in the m/c:");
  scanf("%d",&count);

  q=(struct node *)malloc(sizeof(struct node)*count);

  for(i=0;i<count;i++){
    (q+i)->id_num=i;

    printf("State Machine::%d\n",i);
    printf("Next State if i/p is 0:");
    scanf("%d",&posi);
    (q+i)->link0=(q+posi);

    printf("Next State if i/p is 1:");
    scanf("%d",&posi);
    (q+i)->link1=(q+posi);
    printf("Is the state final state(0/1)?");
    scanf("%d",&(q+i)->st_val);
  }

  printf("Enter the Initial State of the m/c:");
  scanf("%d",&posi);
  start=q+posi;

  printf("=-=-=-=-=-=-=-=-==-=-=-=-=-=-=-=-=-=-=\n");

  while(1){
    printf("=-=-=-=-=-=-=-=-==-=-=-=-=-=-=-=-=-=-=\n");
    printf("Perform String Check(0/1):");
    scanf("%d",&j);
    if(j){
      ptr=start;
      printf("Enter the string of inputs:");
      scanf("%s",n);
      posi=0;

      while(n[posi]!='\0'){
    a[posi]=(n[posi]-'0');
    //printf("%c\n",n[posi]);
    //printf("%d",a[posi]);
    posi++;
      }

      i=0;
      printf("The visited States of the m/c are:");
      do{
    vst_arr[i]=ptr->id_num;
    if(a[i]==0){
      ptr=ptr->link0;
    }
    else if(a[i]==1){
      ptr=ptr->link1;
    }
    else{
      printf("iNCORRECT iNPUT\n");
      return;
    }
    printf("[%d]",vst_arr[i]);
    i++;
      }while(i<posi);

      printf("\n");
      printf("Present State:%d\n",ptr->id_num);
      printf("String Status:: ");
      if(ptr->st_val==1)
    printf("String Accepted\n");
      else
    printf("String Not Accepted\n");
    }
    else
      return 0;
  }
    printf("=-=-=-=-=-=-=-=-==-=-=-=-=-=-=-=-=-=-=\n");
  return 0;
}
```

//OUTPUT



```
=-=-=-=-=-=-=-=-==-=-=-=-=-=-=-=-=-=-=
Enter the number of states in the m/c:2
State Machine::0
Next State if i/p is 0:0
Next State if i/p is 1:1
Is the state final state(0/1)?0
State Machine::1
Next State if i/p is 0:1
Next State if i/p is 1:0
Is the state final state(0/1)?1
Enter the Initial State of the m/c:0
=-=-=-=-=-=-=-=-==-=-=-=-=-=-=-=-=-=-=
=-=-=-=-=-=-=-=-==-=-=-=-=-=-=-=-=-=-=
Perform String Check(0/1):1
Enter the string of inputs:1000111101111
The visited States of the m/c are:[0][1][1][1][1][0][1][0][1][1][0][1][0]
Present State:1
String Status:: String Accepted
=-=-=-=-=-=-=-=-==-=-=-=-=-=-=-=-=-=-=
Perform String Check(0/1):0

--------------------------------
Process exited after 56.19 seconds with return value 0
Press any key to continue . . .
```

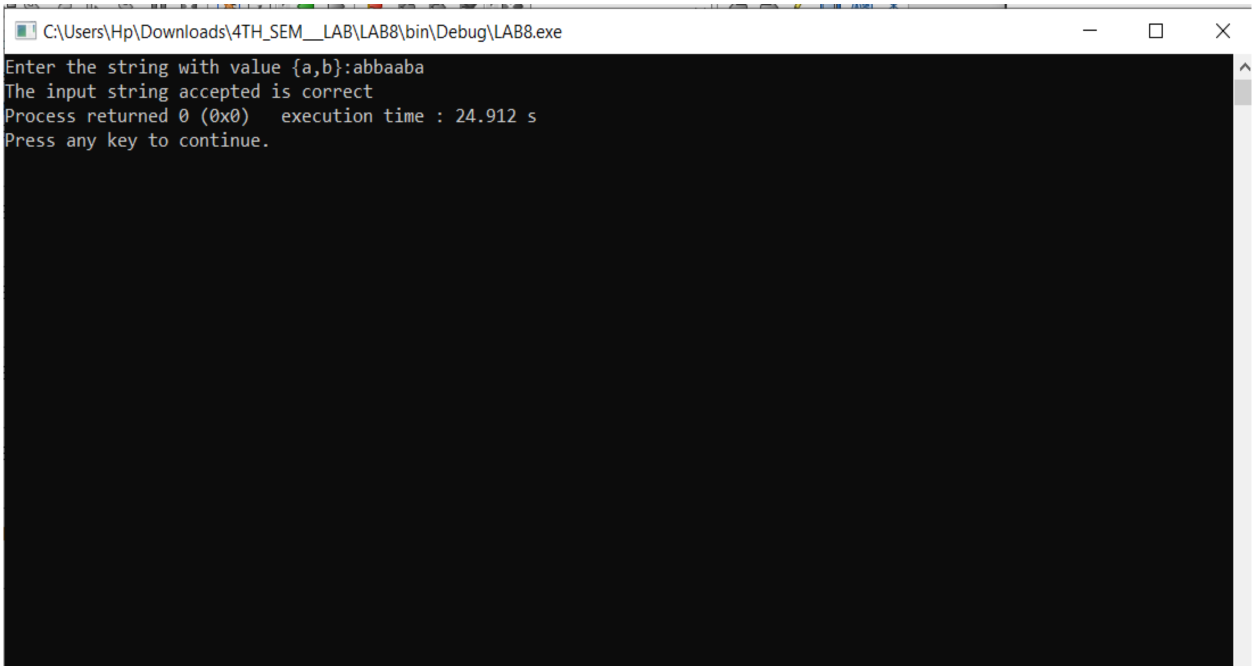
LAB 7
WRITE A PROGRAM FOR THE IMPLEMENTATION OF NON-DETERMINISTIC FINITE AUTOMATA (NFA)

//Source code

```
#include <iostream>

using namespace std;

string Z;
 void A(string, int);
void B(string, int);
void C(string, int);

void A(string s, int i)

{

          if (i == s.length()) {

          Z="incorrect";

          return;

          }

          if (s[i] == 'a')

          {    A(s, i + 1);

          B(s, i + 1);}

          else

                    A(s, i + 1);

}

void B(string s, int i)

{
	if (i == s.length()) {
	Z="incorrect";
	return;
	}
	if (s[i] == 'a')
	C(s, i + 1);
}
void C(string s, int i)
{
	if (i == s.length()) {
	Z="correct";
	return;
	}
	C(s, i + 1);
}
int main()
{
	string s;
	cout<<"Enter the string with value {a,b}:";
	cin>>s;
	A(s,0);
	cout<<"The input string accepted is "<<Z;
	return 0;
```

```
}
```

//OUTPUT



```
Enter the string with value {a,b}:abbaaba
The input string accepted is correct
Process returned 0 (0x0)   execution time : 24.912 s
Press any key to continue.
```

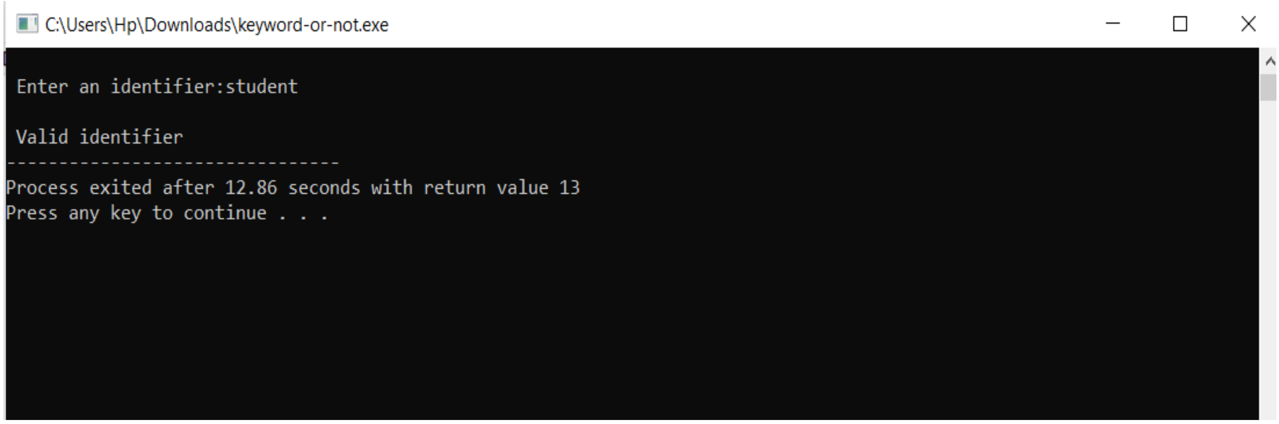
LAB 8

WRITE A C PROGRAM TO IDENTIFY WHETHER A GIVEN STRING IS IDENTIFIER OR NOT

//SOURCE CODE

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{

char a[10];
int flag, i=1;
printf("\n Enter an identifier:");
gets(a);

if(isalpha(a[0]))

flag=1;

else

printf("\n Not a valid identifier");

while(a[i]!='\0')
{

if(!isdigit(a[i])&&!isalpha(a[i]))

{

flag=0;

break;

}

i++;
}

if(flag==1)

printf("\n Valid identifier");

getch();

}
```

//OUTPUT



```
 Enter an identifier:student

 Valid identifier
--------------------------------
Process exited after 12.86 seconds with return value 13
Press any key to continue . . .
```

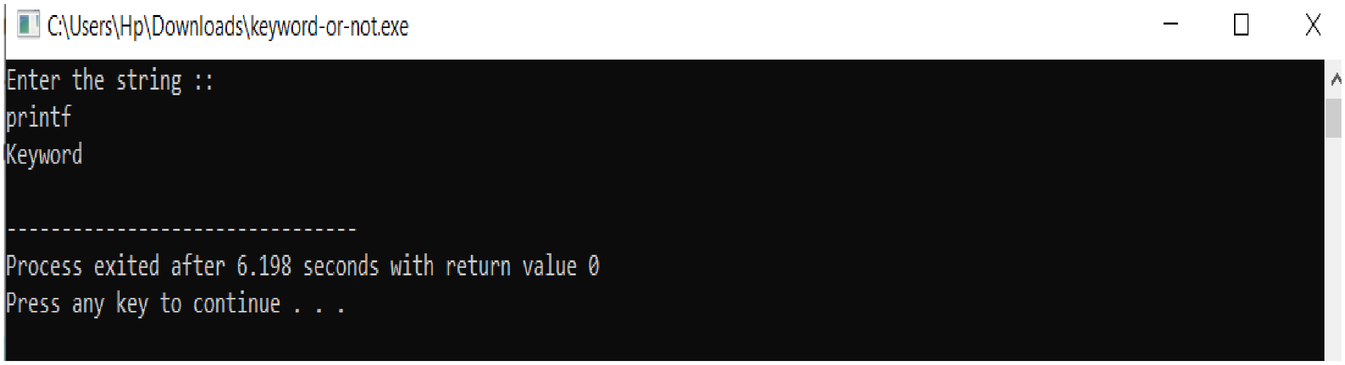
LAB 9

WRITE A C PROGRAM TO IDENTIFY WHETHER A GIVEN STRING IS KEYWORD OR NOT

//SOURCE CODE

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main()
{
char a[5][10]={"printf","scanf","if","else","break"};
char str[10];
int i,flag;

puts("Enter the string :: ");
gets(str);
for(i=0;i<strlen(str);i++)
{
if(strcmp(str,a[i])==0)
{
flag=1;
break;
}
else
flag=0;
}
if(flag==1)
puts("Keyword");
else
puts("String");

return 0;
}
```

//OUTPUT



```
Enter the string ::
printf
Keyword

--------------------------------
Process exited after 6.198 seconds with return value 0
Press any key to continue . . .
```