Name : Diksha Gupta

Roll No. : 27 | A2

PRACTICAL NO. 1

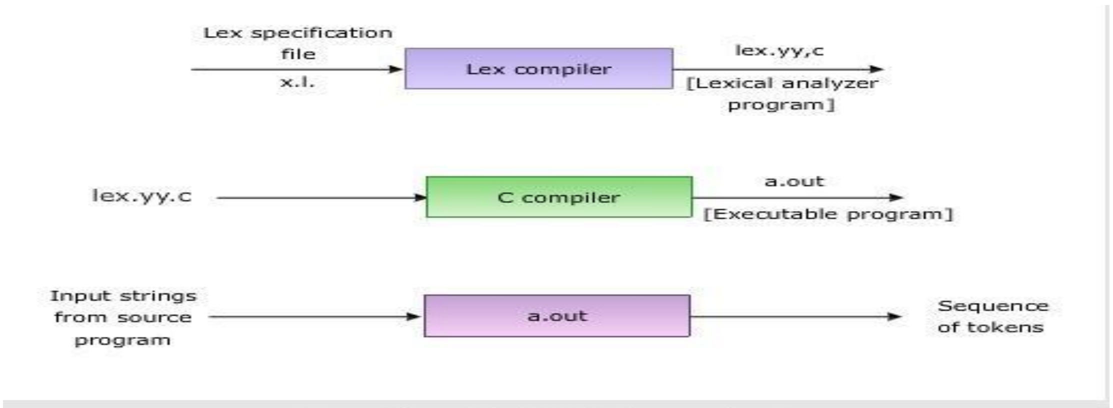
THEORY

LEX: Lex is a program generator designed for lexical processing of character input streams. It accepts a high level, problem-oriented specification for character string matching, and produces a program in a general purpose language which recognizes regular expressions. The regular expressions are specified by the user in the source specifications given to Lex. The Lex written code recognizes these expressions in an input stream and partitions the input stream into strings matching the expressions. At the boundaries between strings program sections provided by the user are executed. The Lex source file associates the regular expressions and the program fragments. As each expression appears in the input to the program written by Lex, the corresponding fragment is executed.

Lex is not a complete language, but rather a generator representing a new language feature which can be added to different programming languages, called ``host languages.'' Just as general purpose languages can produce code to run on different com puter hardware, Lex can write code in different host languages.

Lex turns the user's expressions and actions (called source in this pic) into the host general-purpose language; the generated program is named yylex. The yylex program will recognize expressions in a stream (called input in this pic) and perform the specified actions for each expression as it is detected.

Diagram of LEX



Generation of Lexical Analyzer using Lex

Format for Lex file

The general format of Lex source is:

{definitions}
%%
{rules}
%%

{user subroutines}

where the definitions and the user subroutines are often omitted. The second %% is optional, but the first is required to mark the beginning of the rules. The absolute minimum Lex program is thus %% (no definitions, no rules) which translates into a program which copies the input to the output unchanged.

Regular Expression

A regular expression (or RE) specifies a set of strings that matches it; the functions in this module let you check if a particular string matches a given regular expression (or if a given regular expression matches a particular string, which comes down to the same thing).

Regular expressions can be concatenated to form new regular expressions; if A and B are both regular expressions, then AB is also a regular expression. In general, if a string p matches A and another string q matches B, the string pq will match AB. This holds unless A or B contain low precedence operations; boundary conditions between A and B; or have numbered group references. Thus, complex expressions can easily be constructed from simpler primitive expressions. Regular expressions can contain both special and ordinary characters. Most ordinary characters, like "A", "a", or "0", are the simplest regular expressions; they simply match themselves. You can concatenate ordinary characters, so last matches the string 'last'. (In the rest of this section, we'll write RE's in this special style, usually without quotes, and strings to be matched 'in single quotes'.)Some characters, like "|" or "(", are special. Special characters either stand for classes of ordinary characters or affect how the regular expressions around them are interpreted.

Lex Library Routines

Lex library routines are those functions which have a detailed knowledge of the lex functionalities and which can be called to implement various tasks in a lex program.

The following table gives a list of some of the lex routines.

Lex Routine	Description
Main()	Invokes the lexical analyzer by calling the yylex subroutine.
yywrap()	Returns the value 1 when the end of input occurs.
yymore()	Appends the next matched string to the current value of the yytext array rather than replacing the contents of the yytext array.
yyless(int n)	Retains n initial characters in the yytext array and returns the remaining characters to the input stream.
yyreject	Allows the lexical analyzer to match multiple rules for the same input string. (The yyreject subroutine is called when the special action REJECT is used.)
yylex()	The default main () contains the call of yylex()

ANSWER THE QUESTIONS:

1. **Use of yywrap**

ANS : The function yywrap*()* is called when the scanner encounters the end of the file(input). If yywrap*()* **returns 0** then the scanner continues scanning but when yywrap*()* **returns 1** this means that the end of the file has encountered.

2. Use of yylex function

ANS : yylex() returns a value indicating the type of token that has been obtained. If the token has an actual value, this value (or some representation of the value, for example, a pointer to a string containing the value) is returned in an external variable named yylval.

3. What does lex.yy.c. do?

ANS : The lex command stores the yylex function in a file named lex. yy. c. You can use the yylex function alone to recognize simple one-word input, or you can use it with other C language programs to perform more difficult input analysis functions.

PRACTICAL NO. E1

Aim: Design a lexical analyzer to identify the tokens such as keywords, identifiers, operators,constants (Int &amp; float), special symbols and strings for C language using LEX.

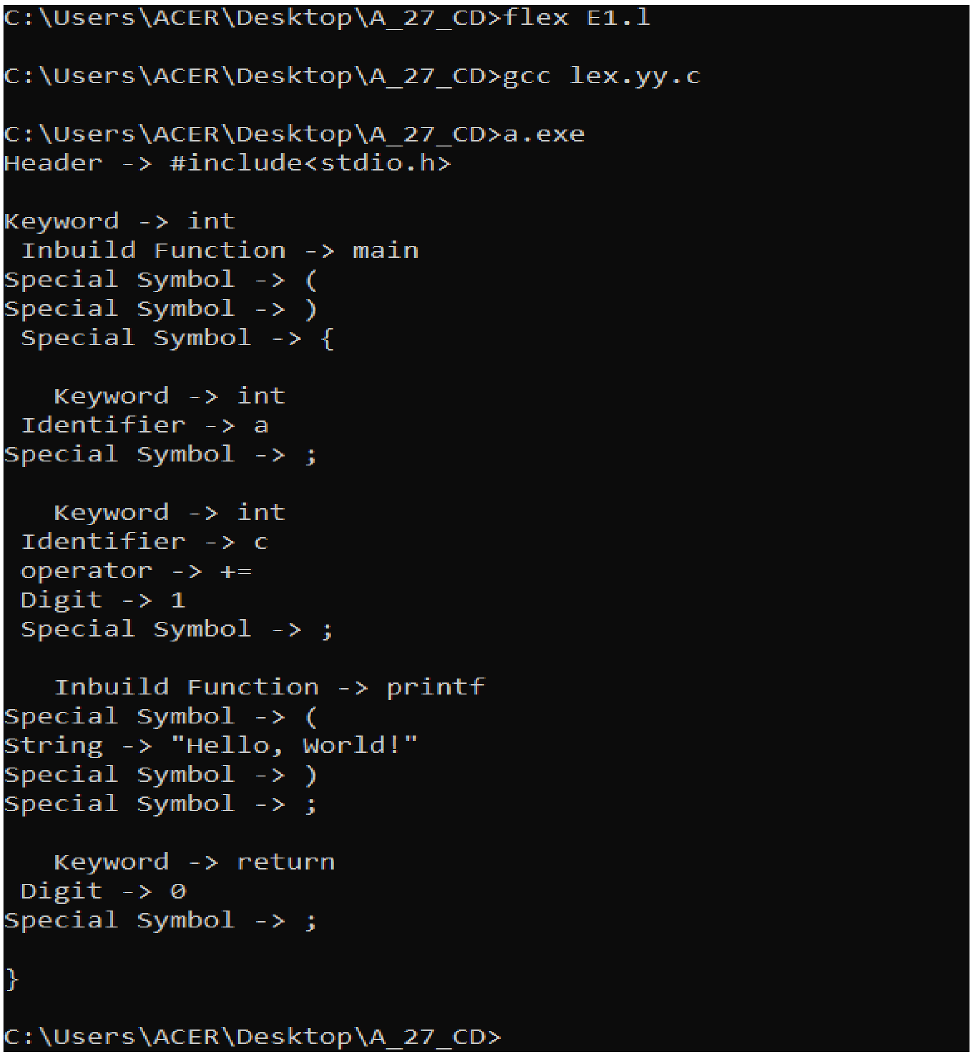
PROGRAM:

```
%{
#include<stdio.h>
%}
%%
int|float|double|char|return {printf("Keyword -> %s\n",yytext);}
main|printf {printf("Inbuild Function -> %s\n",yytext);}
\+|\-|\*|\%|\!|\&|\|\=|\+= {printf("operator -> %s\n",yytext);}
"#include"[" "]*"<"[a-z]+".h>" {printf("Header -> %s\n",yytext);}
[a-zA-Z][a-zA-Z0-9]* {printf("Identifier -> %s\n",yytext);}
\;|\"|\,|_|\(|\)|\{|\{ {printf("Special Symbol -> %s\n",yytext);}
[0-9]+ {printf("Digit -> %s\n",yytext);}
\"(\\.|[^"\\])*\" {printf("String -> %s\n",yytext);}
%%
main(void)
{
 yyin = fopen("myfile.txt","r");
 yylex();
}
int yywrap()
{
return(1);
}
```

INPUT:

```
  #include<stdio.h>
  int main() {
  int a;
  int c += 1 ;
  printf("Hello, World!");
  return 0;
}
```

OUTPUT:



```
C:\Users\ACER\Desktop\A_27_CD>flex E1.l

C:\Users\ACER\Desktop\A_27_CD>gcc lex.yy.c

C:\Users\ACER\Desktop\A_27_CD>a.exe
Header -> #include<stdio.h>

Keyword -> int
 Inbuild Function -> main
Special Symbol -> (
Special Symbol -> )
 Special Symbol -> {

   Keyword -> int
 Identifier -> a
Special Symbol -> ;

   Keyword -> int
 Identifier -> c
 operator -> +=
 Digit -> 1
 Special Symbol -> ;

   Inbuild Function -> printf
Special Symbol -> (
String -> "Hello, World!"
Special Symbol -> )
Special Symbol -> ;

   Keyword -> return
 Digit -> 0
Special Symbol -> ;

}

C:\Users\ACER\Desktop\A_27_CD>
```

PRACTICAL NO. E2

Aim: Write a Lex program to find the parameters given below. Consider as input a question paper of an examination and find:Date of examination, semester, number of questions, numbers of words, lines, small letters,capital letters, digits, and special characters.

PROGRAM:

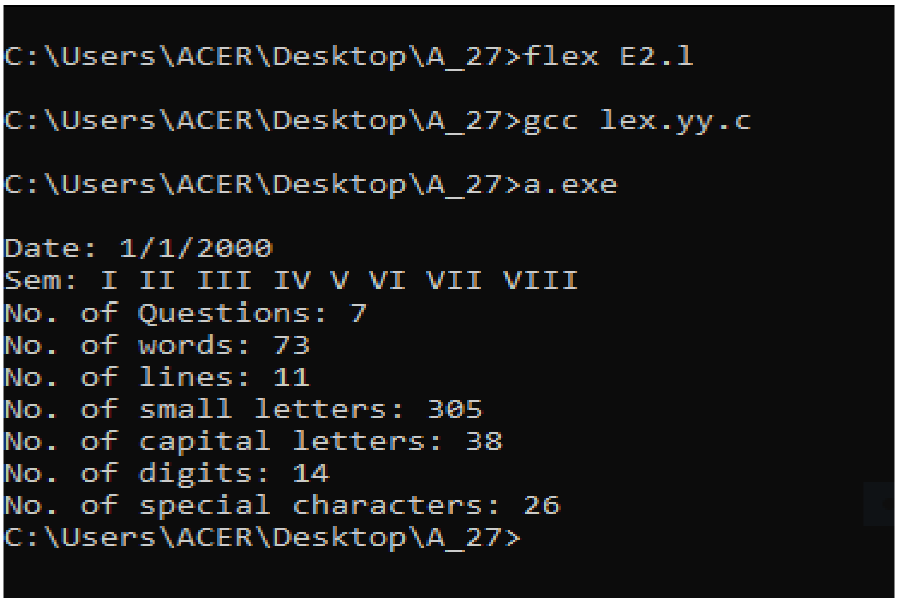
```
 %{
#include<stdio.h>
#include<string.h>
char date[10],sem[50];
int qcount=0,words=0,lines=0,small=0,capital=0,digits=0,spc=0;
%}
Date [0-9]*\/[0-9]*\/[0-9]*
```

```
Special [:|,|?|/]
%%
{Date} {strcpy(date,yytext); digits+=(yyleng-2); spc+=2;}
[IV]+ {strcat(sem,strcat(yytext," ")); capital+=yyleng;}
Question {qcount++; capital++; small+=7;}
[\n] {lines++; words++;}
[\t ] {words++;}
[a-z] {small++;}
[A-Z] {capital++;}
[0-9] {digits++;}
{Special} {spc++;}
%%
main(void)
{
yyin=fopen("ques_paper.txt","r");
yylex();
printf("\nDate: %s",date);
     printf("\nSem: %s",sem);
printf("\nNo. of Questions: %d",qcount);
printf("\nNo. of words: %d",words);
printf("\nNo. of lines: %d",lines);
printf("\nNo. of small letters: %d",small);
printf("\nNo. of capital letters: %d",capital);
printf("\nNo. of digits: %d",digits);
printf("\nNo. of special characters: %d",spc);
}
int yywrap()
{
return(1);
}
```

INPUT:

```
                ABC College

1/1/2000 Sem: I, II, III, IV, V, VI, VII, VIII
Question1 : What are the benefits of tree plantation?
Question2 : What is water pollution?
Question3 : What should be done to avoid road accidents?
Question4 : What are your view on noise pollution?
Question5 : Why should people adopt pets?
Question6 : What is green gym?
Question7 : What norms must pe implemented to minimize the loss from
construction to
environment?
Question8 : What is air pollution?
```

OUTPUT:


```
C:\Users\ACER\Desktop\A_27>flex E2.l

C:\Users\ACER\Desktop\A_27>gcc lex.yy.c

C:\Users\ACER\Desktop\A_27>a.exe

Date: 1/1/2000
Sem: I II III IV V VI VII VIII
No. of Questions: 7
No. of words: 73
No. of lines: 11
No. of small letters: 305
No. of capital letters: 38
No. of digits: 14
No. of special characters: 26
C:\Users\ACER\Desktop\A_27>
```

PRACTICAL NO. E3

Aim: Create a txt file to containing the following without heading: Name of Student, Company Placed in (TCS, Infosys, Wipro, Accenture, Informatica), Male/female, CGPA (floating point number), Department (CSE, IT, EC), Package (floating point number), mail id, mobile number (integer exactly 10 digits). At least 25 records must be present.

PROGRAM:

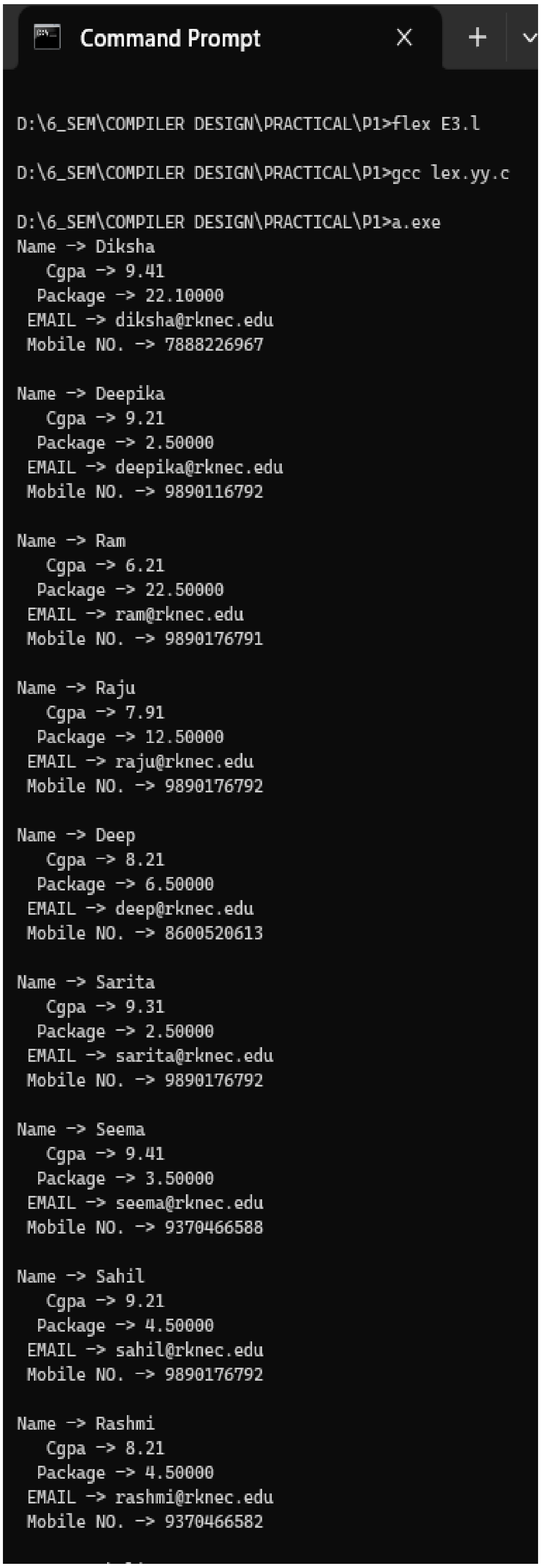
```
%{
#include<stdio.h>
int TCS=0 ,Infosys =0 ,Wipro= 0 ,Accenture=0 ,Informatica=0 ,male =0 , Female = 0 ,
CSE =0 , IT=0 , EC =0 ;
%}

%%
TCS TCS++;
Infosys Infosys++;
Wipro Wipro++;
Accenture Accenture++;
Informatica Informatica++;
male male++;
Female Female++;
CSE CSE++;
IT IT++;
EC EC++;
[a-zA-Z][a-zA-Z]* {printf("Name -> %s\n",yytext);}
[5-9][.][0-9][0-9] {printf("Cgpa -> %s\n",yytext);}
```

```
[1-9]{6} {printf("Package -> %s\n",yytext);}
[1-9]+[.][0-9]*  {printf("Package -> %s\n",yytext);}
[7-9][0-9]{9} {printf("Mobile NO. -> %s\n",yytext);}
[a-z]+@rknec".edu" {printf("EMAIL -> %s\n",yytext);}
%%

main(void)
{
 yyin = fopen("E3file.txt","r");
 yylex();
 printf("Number of students placed in TCS: %d\n",TCS);
 printf("Number of students placed in Infosys: %d\n",Infosys);
 printf("Number of students placed in Accenture: %d\n",Accenture);
 printf("Number of students placed in Informatica: %d\n",Informatica);
 printf("Number of students placed in Wipro: %d\n",Wipro);
 printf("Number of female students: %d\n",Female);
 printf("Number of male students: %d\n",male);
 printf("Number of CSE students: %d\n",CSE);
 printf("Number of CSE students: %d\n",IT);
 printf("Number of CSE students: %d\n",EC);
}
int yywrap()
{
return(1);
}
```

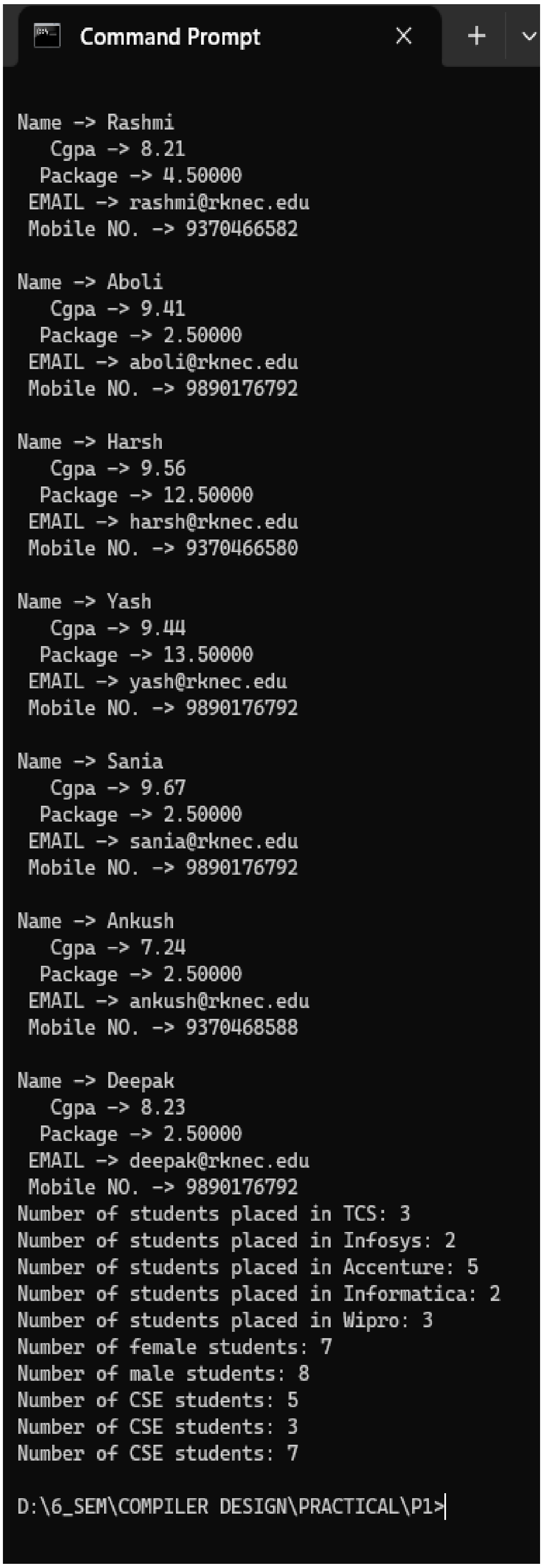
INPUT:

```
Diksha TCS Female 9.41 CSE 22.10000 diksha@rknec.edu 7888226967
Deepika Accenture Female 9.21 EC 2.50000 deepika@rknec.edu 9890116792
Ram Infosys male 6.21 EC 22.50000 ram@rknec.edu 9890176791
Raju TCS male 7.91 IT 12.50000 raju@rknec.edu 9890176792
Deep Accenture male 8.21 CSE 6.50000 deep@rknec.edu 8600520613
Sarita Wipro Female 9.31 EC 2.50000 sarita@rknec.edu 9890176792
Seema Accenture Female 9.41 IT 3.50000 seema@rknec.edu 9370466588
Sahil TCS male 9.21 CSE 4.50000 sahil@rknec.edu 9890176792
Rashmi Wipro Female 8.21 IT 4.50000 rashmi@rknec.edu 9370466582
Aboli Accenture Female 9.41 EC 2.50000 Aboli@rknec.edu 9890176792
Harsh Wipro male 9.56 CSE 12.50000 Harsh@rknec.edu 9370466580
Yash Accenture male 9.44 CSE 13.50000 Yash@rknec.edu 9890176792
Sania Infosys Female 9.67 EC 2.50000 Sania@rknec.edu 9890176792
Ankush Informatica male 7.24 EC 2.50000 Ankush@rknec.edu 9370468588
Deepak Informatica male 8.23 EC 2.50000 Deepak@rknec.edu 9890176792
```

OUTPUT:



```
D:\6_SEM\COMPILER DESIGN\PRACTICAL\P1>flex E3.l

D:\6_SEM\COMPILER DESIGN\PRACTICAL\P1>gcc lex.yy.c

D:\6_SEM\COMPILER DESIGN\PRACTICAL\P1>a.exe
Name -> Diksha
   Cgpa -> 9.41
  Package -> 22.10000
 EMAIL -> diksha@rknec.edu
 Mobile NO. -> 7888226967

Name -> Deepika
   Cgpa -> 9.21
  Package -> 2.50000
 EMAIL -> deepika@rknec.edu
 Mobile NO. -> 9890116792

Name -> Ram
   Cgpa -> 6.21
  Package -> 22.50000
 EMAIL -> ram@rknec.edu
 Mobile NO. -> 9890176791

Name -> Raju
   Cgpa -> 7.91
  Package -> 12.50000
 EMAIL -> raju@rknec.edu
 Mobile NO. -> 9890176792

Name -> Deep
   Cgpa -> 8.21
  Package -> 6.50000
 EMAIL -> deep@rknec.edu
 Mobile NO. -> 8600520613

Name -> Sarita
   Cgpa -> 9.31
  Package -> 2.50000
 EMAIL -> sarita@rknec.edu
 Mobile NO. -> 9890176792

Name -> Seema
   Cgpa -> 9.41
  Package -> 3.50000
 EMAIL -> seema@rknec.edu
 Mobile NO. -> 9370466588

Name -> Sahil
   Cgpa -> 9.21
  Package -> 4.50000
 EMAIL -> sahil@rknec.edu
 Mobile NO. -> 9890176792

Name -> Rashmi
   Cgpa -> 8.21
  Package -> 4.50000
 EMAIL -> rashmi@rknec.edu
 Mobile NO. -> 9370466582
```



```
Name -> Rashmi
   Cgpa -> 8.21
  Package -> 4.50000
 EMAIL -> rashmi@rknec.edu
 Mobile NO. -> 9370466582

Name -> Aboli
   Cgpa -> 9.41
  Package -> 2.50000
 EMAIL -> aboli@rknec.edu
 Mobile NO. -> 9890176792

Name -> Harsh
   Cgpa -> 9.56
  Package -> 12.50000
 EMAIL -> harsh@rknec.edu
 Mobile NO. -> 9370466580

Name -> Yash
   Cgpa -> 9.44
  Package -> 13.50000
 EMAIL -> yash@rknec.edu
 Mobile NO. -> 9890176792

Name -> Sania
   Cgpa -> 9.67
  Package -> 2.50000
 EMAIL -> sania@rknec.edu
 Mobile NO. -> 9890176792

Name -> Ankush
   Cgpa -> 7.24
  Package -> 2.50000
 EMAIL -> ankush@rknec.edu
 Mobile NO. -> 9370468588

Name -> Deepak
   Cgpa -> 8.23
  Package -> 2.50000
 EMAIL -> deepak@rknec.edu
 Mobile NO. -> 9890176792
Number of students placed in TCS: 3
Number of students placed in Infosys: 2
Number of students placed in Accenture: 5
Number of students placed in Informatica: 2
Number of students placed in Wipro: 3
Number of female students: 7
Number of male students: 8
Number of CSE students: 5
Number of CSE students: 3
Number of CSE students: 7

D:\6_SEM\COMPILER DESIGN\PRACTICAL\P1>
```