**Name: Aaron Emmanuel Rocque**

**Roll No: A-31**

**Practical No. 1**

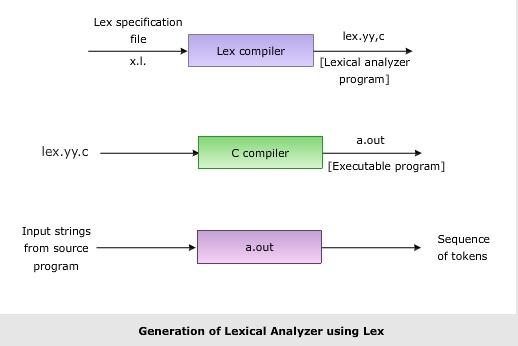
**Theory**

**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**



**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 pqwill 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.

* yywrap is a function that is used in programs that analyze text. It helps to determine what should happen when the end of the text is reached. By default, the program will stop when it reaches the end, but yywrap can be used to customize this behavior. If yywrap returns a non-zero value, the program will continue processing additional input.

1. Use of yylex function

* In simpler terms, yylex() is a function that scans input text and identifies tokens (e.g., keywords, operators, identifiers, literals) based on predefined patterns. It is often used as part of a compiler or interpreter for a programming language, where it is responsible for tokenizing the input code before passing it on to the next stage of processing.

1. What does lex.yy.c do?

* lex.yy.c is the generated code file for the lexer, which is used to tokenize input text according to specified patterns.

**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. Use File for the input.

**Program:**

/\* ----------Definition Section---------- \*/

%{

    #include<stdio.h>

%}

/\* ----------Rules Section---------- \*/

**%%**

auto|break|case|char|const|continue|default|do|double|else|enum|extern|float|for|goto|if|int|long|register|return|short|signed|sizeof|static|struct|switch|typedef|union|unsigned|void|volatile|while|\_Packed  {ECHO; printf("\tKEYWORD\n");}

\".\*\" {ECHO; printf("\tSTRING\n");}

[0-9]\* {ECHO; printf("\tINTEGER\n");}

[0-9]\*\.?[0-9]\* {ECHO; printf("\tFLOAT\n");}

[a-zA-z]([a-zA-z]|[0-9])\* {ECHO; printf("\tIDENTIFIER\n");}

[\+\-\\*\\\%<>=!&\|\^\?] {ECHO; printf("\tOPERATOR\n");}

[{};,()] {ECHO; printf("\tSYMBOL\n");}

**%%**

/\* ----------Subroutine Section---------- \*/

int yywrap() {

       return(1);

}

int main() {

    yyin = fopen("myfile.txt","r");

    yylex();

    return 0;

}

**Input:**

#include <stdio.h>

int main() {

float a = 10.24;

float b = 11.00;

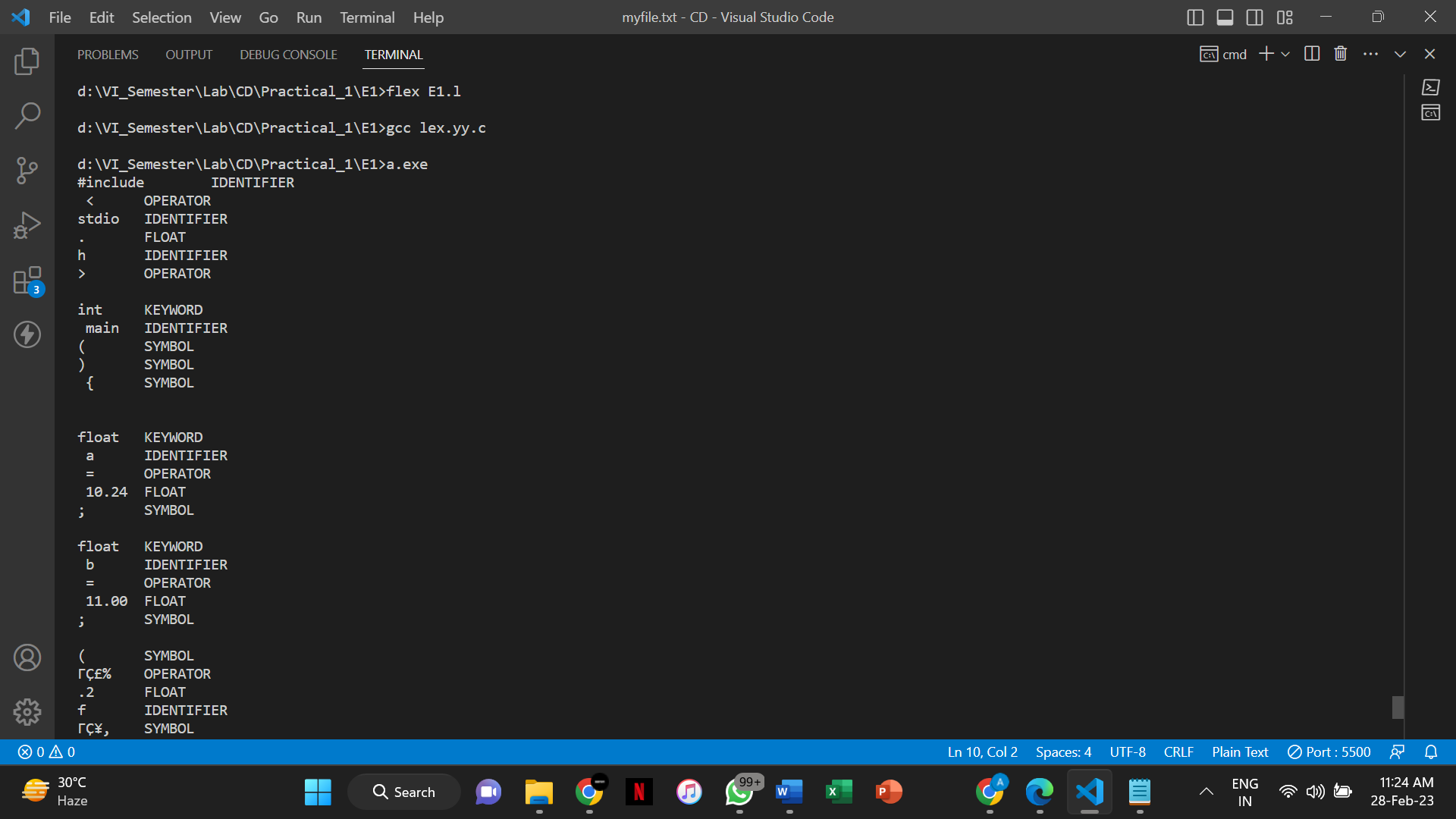
int c = 1;

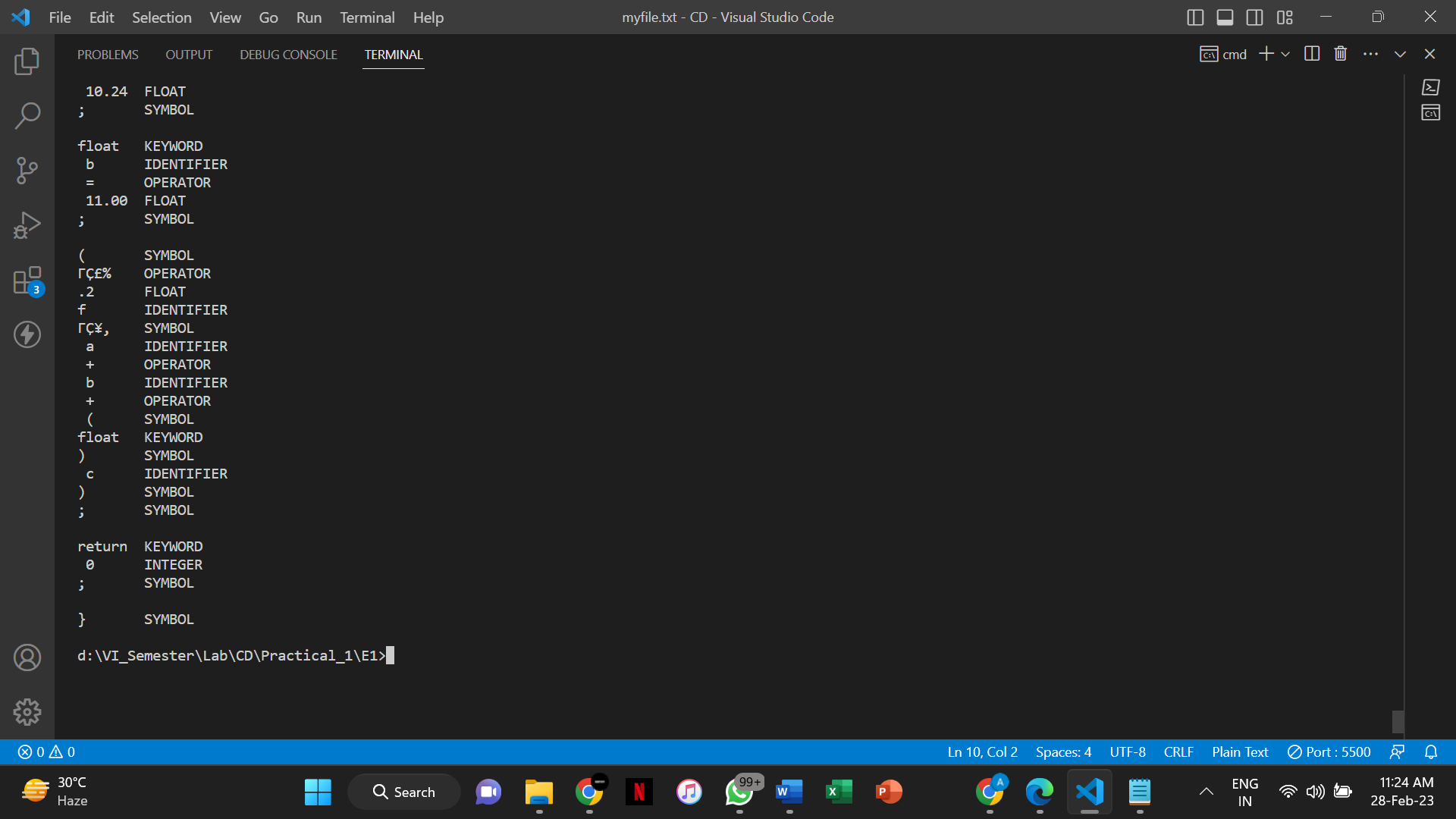
printf(“%.2f”, a + b + (float) c);

return 0;

}

**Output:**



****

**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:**

/\* ----------Definition Section---------- \*/

%{

    #include<stdio.h>

    #include<string.h>

    int words = 0;

    int qno = 0;

    int lines = 0;

    int smallLetters = 0;

    int capitalLetters = 0;

    int digits = 0;

    int specialChar = 0;

%}

digit [0-9]

special [:|,|\?|//]

/\* ----------Rules Section---------- \*/

**%%**

\n {lines++; words++;}

[\t ' '] {words++;}

[A-Z] {capitalLetters++;}

[a-z] {smallLetters++;}

{digit} {digits++;}

[0-9]\/[0-9]\/[0-9][0-9][0-9][0-9] {ECHO; printf("\tDATE\n");}

Sem.\* {ECHO; printf(" are semester numbers.\n");}

Question{digit} {qno++;}

{special} {specialChar++;}

**%%**

/\* ----------Subroutine Section---------- \*/

int yywrap() {

       return(1);

}

int main(void) {

    yyin = fopen("QuestionPaper.txt","r");

    yylex();

    printf("\nThe number of questions are %d\n", qno);

    printf("The number of words are %d\n", words);

    printf("The number of lines are %d\n", lines);

    printf("The number of small letters are %d\n", smallLetters);

    printf("The number of capital letters are %d\n", capitalLetters);

    printf("The number of digits are %d\n", digits);

    printf("The number of special characters are %d\n", specialChar);

    return 0;

}

**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?

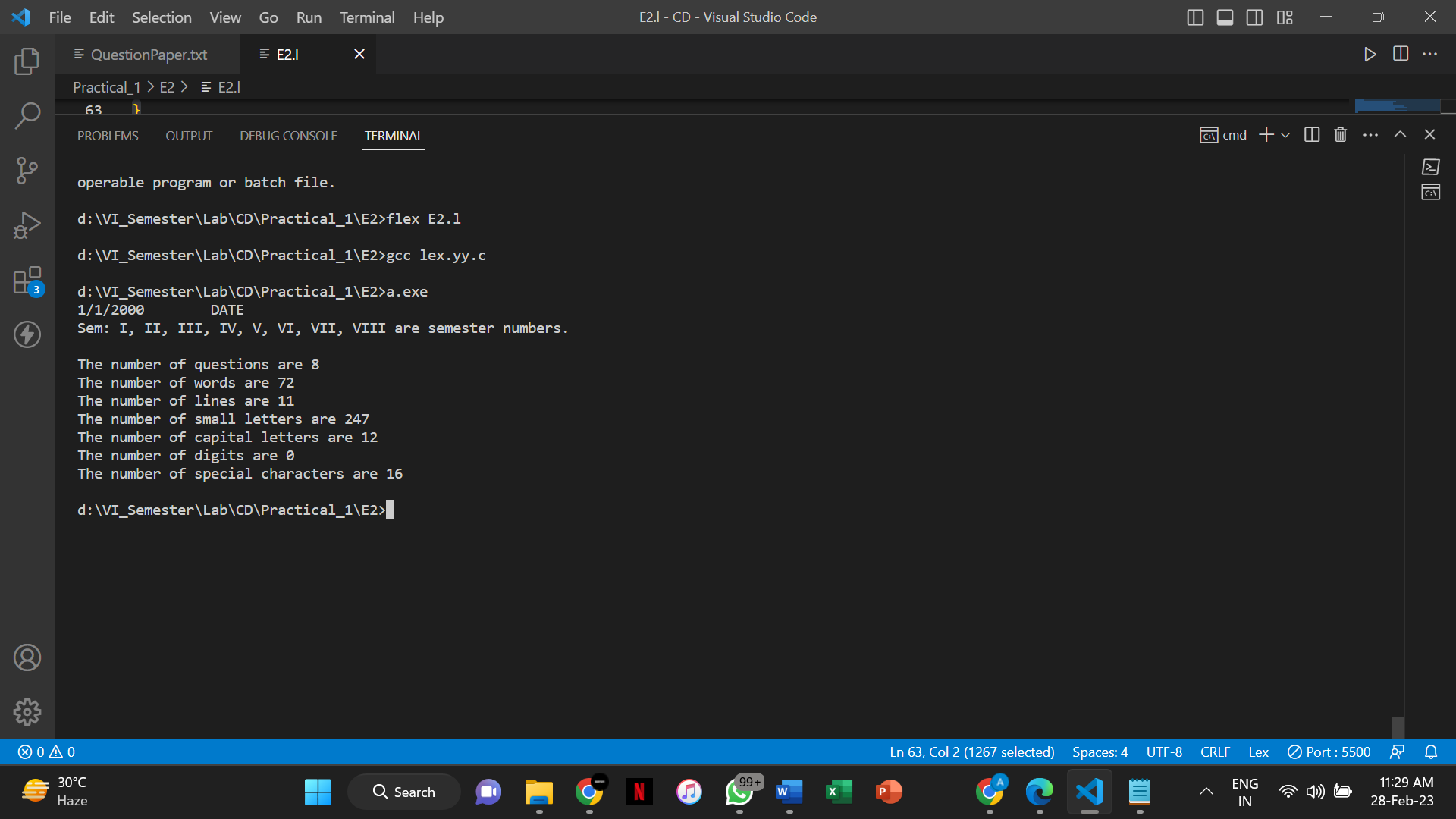
Question6 : What is green gym?

Question7 : What norms must pe implemented to minimize the loss from construction to

environment?

Question8 : What is air pollution?

**Output:**

****

**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.

Write a Lex program to find the parameters given below:

o Identify Name of student and display it.

o Identify CGPA and display (should be less than 10)

o Identify Package and display it

o Identify mail id and display

o Identify mobile number and display

o Find number of students placed in each of the company

o Number of female students

o Number of male students

o Number of CSE, IT and EC students who are placed

**Program:**

/\* ----------Definition Section---------- \*/

%{

    #include<stdio.h>

    #include<string.h>

    int male = 0;

    int female = 0;

    int tcs = 0;

    int infosys = 0;

    int wipro = 0;

    int accenture = 0;

    int informatica = 0;

    int cse = 0;

    int it = 0;

    int ec = 0;

%}

/\* ----------Rules Section---------- \*/

**%%**

"Male" {male++;}

"Female" {female++;}

"TCS" {tcs++;}

"Infosys" {infosys++;}

"Wipro" {wipro++;}

"Accenture" {accenture++;}

"Informatica" {informatica++;}

"CSE" {cse++;}

"IT" {it++;}

"EC" {ec++;}

[A-Za-z]+ {printf("\nName: %s\n", yytext);}

[0-9]\.[0-9]+ {printf("CGPA: %s\n", yytext);}

[0-9]{10} {printf("Mobile Number: %s\n", yytext);}

[0-9]+ {printf("Package: %s\n", yytext);}

[a-zA-Z0-9]+.[@].[a-zA-Z0-9]+.[/.].[a-zA-Z.]+ {printf("Email: %s\n", yytext);}

**%%**

/\* ----------Subroutine Section---------- \*/

int yywrap() {

       return(1);

}

int main(void) {

    yyin = fopen("input.txt","r");

    yylex();

    printf("\nThe number of Female students are %d\n", female);

    printf("The number of Male students are %d\n", male);

    printf("The number of placed in TCS are %d\n", tcs);

    printf("The number of placed in Infosys are %d\n", infosys);

    printf("The number of placed in Wipro are %d\n", wipro);

    printf("The number of placed in Accenture are %d\n", accenture);

    printf("The number of placed in Informatica are %d\n", informatica);

    printf("The number of CSE students placed are %d\n", cse);

    printf("The number of IT students placed are %d\n", it);

    printf("The number of EC students placed are %d\n", ec);

    return 0;

}

**Input:**

Abc TCS Female 9.4 CSE 600000 abc@rknec.edu 9999999999

Awe Wipro Female 6.2 IT 230000 awe@rknec.edu 8888888888

Bte Infosys Male 7.4 EC 590000 bte@rknec.edu 7878787878

Xaw Accenture Male 2.4 EC 990000 bxaw@rknec.edu 6666622222

Put Informatica Female 6.9 CSE 110000 put@rknec.edu 1111199999

Wte Infosys Female 7.1 CSE 630000 wte@rknec.edu 6666666674

Ntw Wipro Male 5.4 IT 870000 ntw@rknec.edu 3333444421

Mut Infosys Female 9.2 IT 900000 mut@rknec.edu 8765434532

Ntq TCS Male 6.9 IT 110000 ntq@rknec.edu 9098786543

Cqe Accenture Male 2.1 CSE 910000 cqu@rknec.edu 89778675643

Bpt Wipro Male 4.9 EC 990000 bpt@rknec.edu 9812341578

Bqe Infosys Female 7.7 EC 160000 bqe@rknec.edu 6745362764

Lop Accenture Female 5.0 CSE 710000 lop@rknec.edu 8787656546

Cwq TCS Female 3.9 EC 930000 cwq@rknec.edu 9856435645

Zut TCS Male 7.4 CSE 770000 zut@rknec.edu 9098987676

Buw Wipro Male 8.4 IT 880000 buw@rknec.edu 7673546345

Nit Wipro Male 9.0 EC 660000 nit@rknec.edu 8978767656

Mop Informatica Male 9.1 CSE 690000 mop@rknec.edu 9098987876

Vut Infosys Male 6.3 CSE 690000 vut@rknec.edu 3454545676

Cop TCS Male 5.9 CSE 440000 cop@rknec.edu 8767645565

Xit TCS Male 5.8 IT 340000 xit@rknec.edu 8987876765

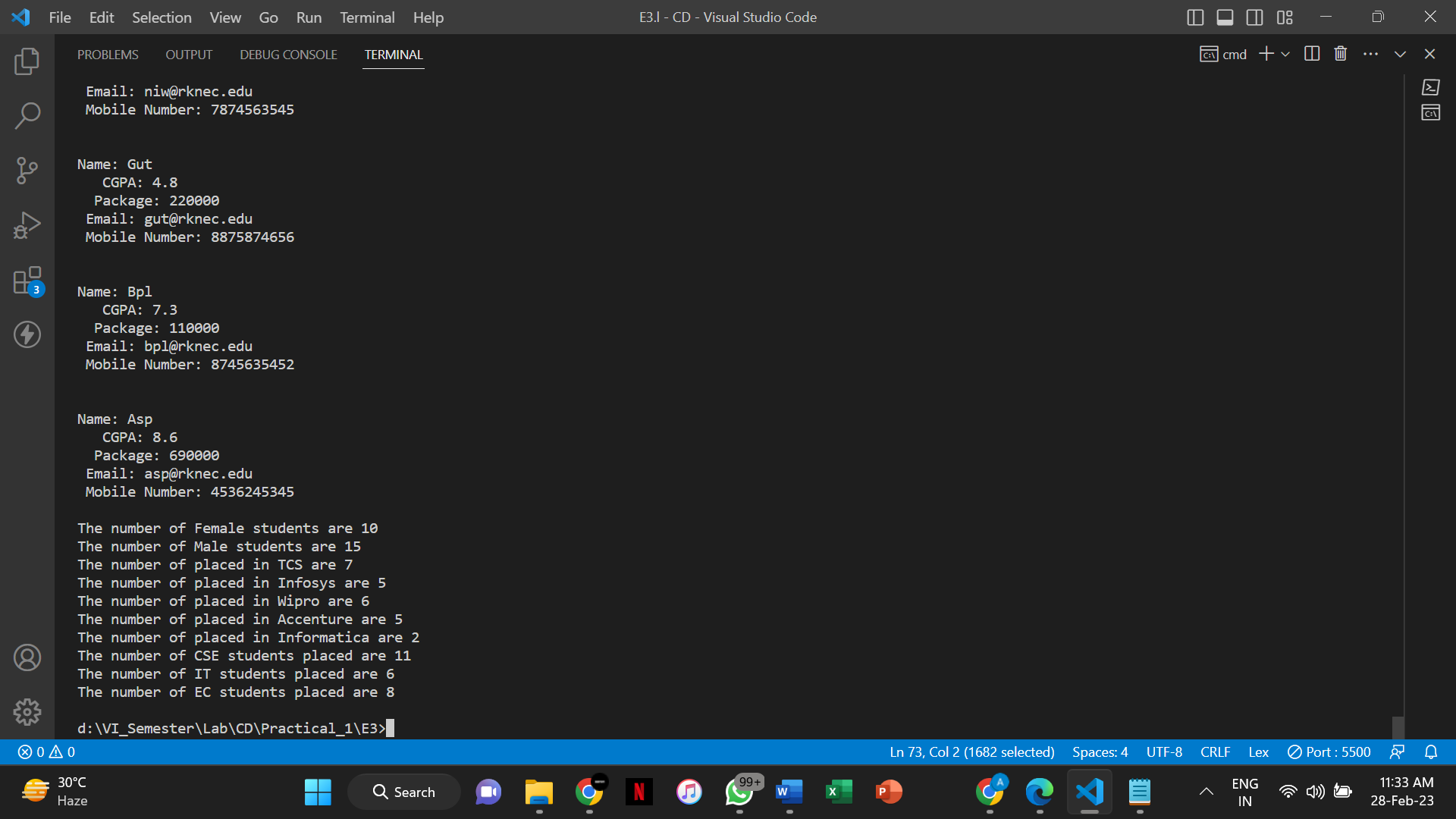
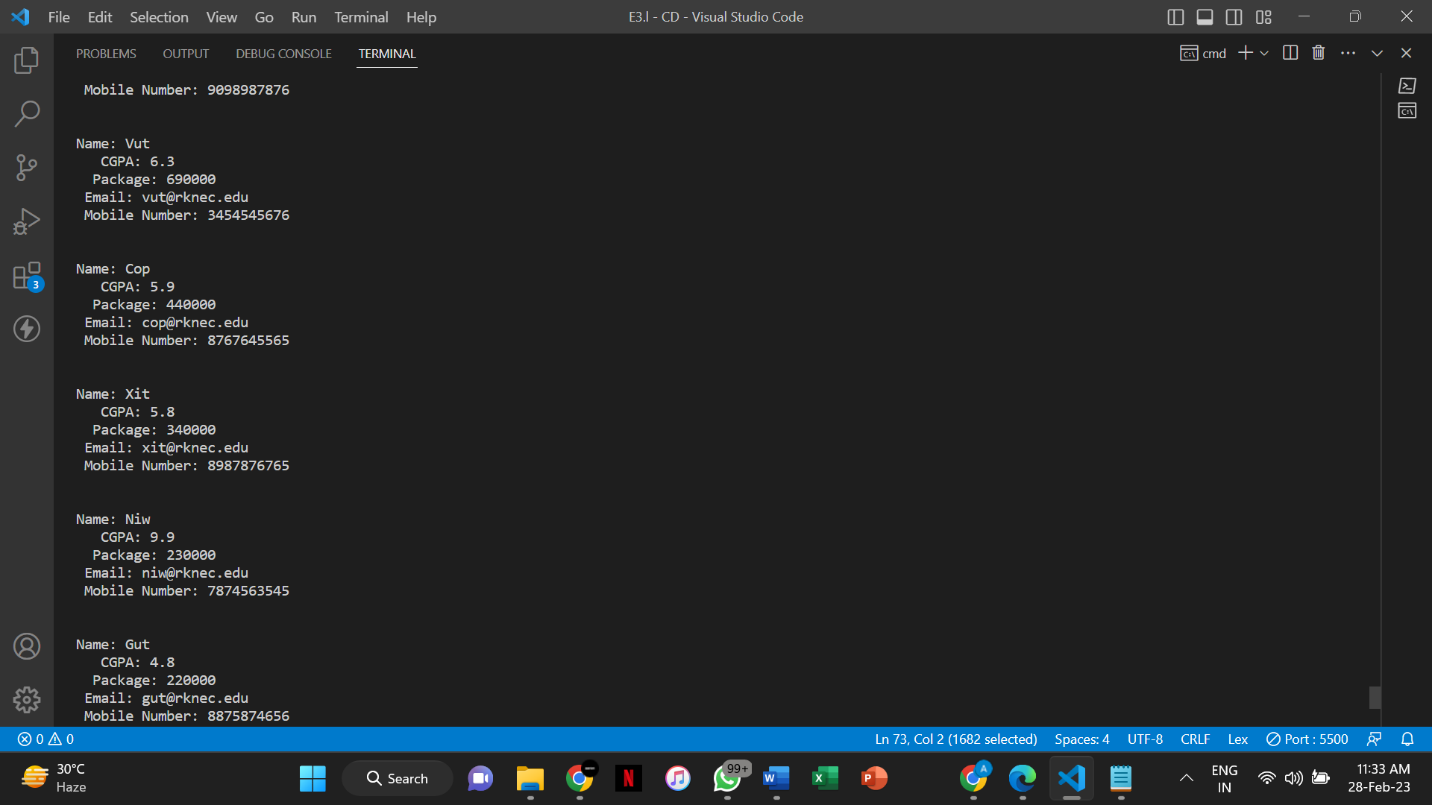
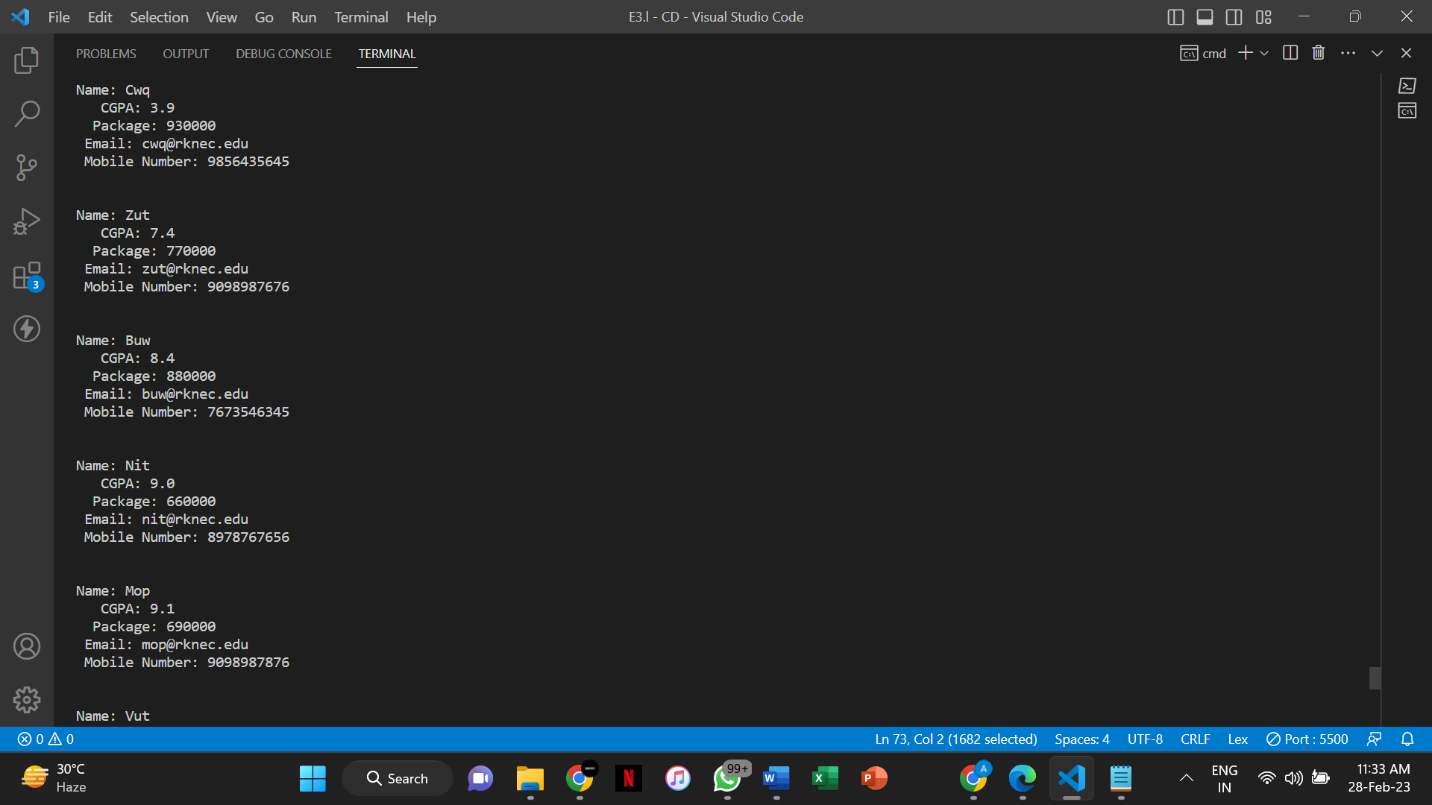
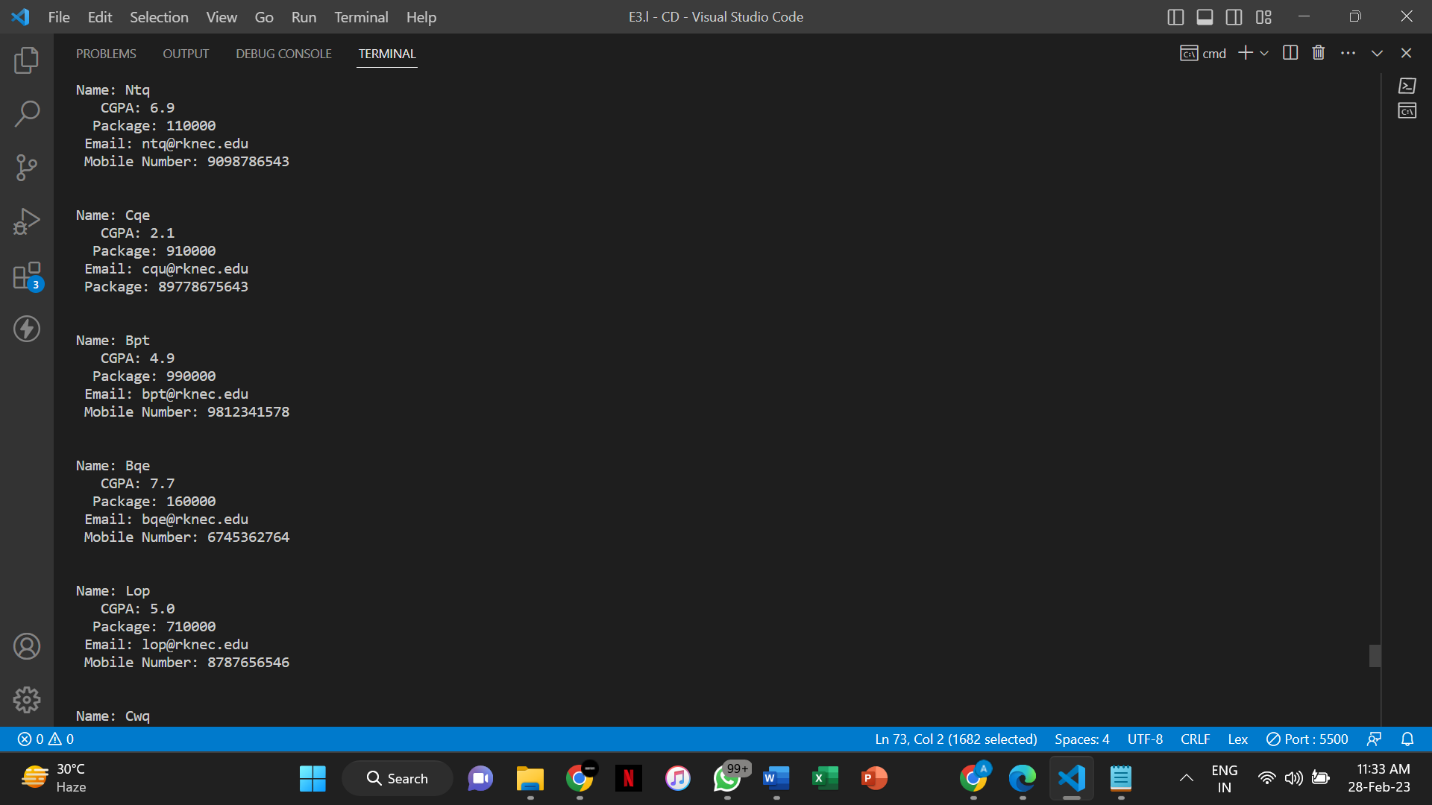
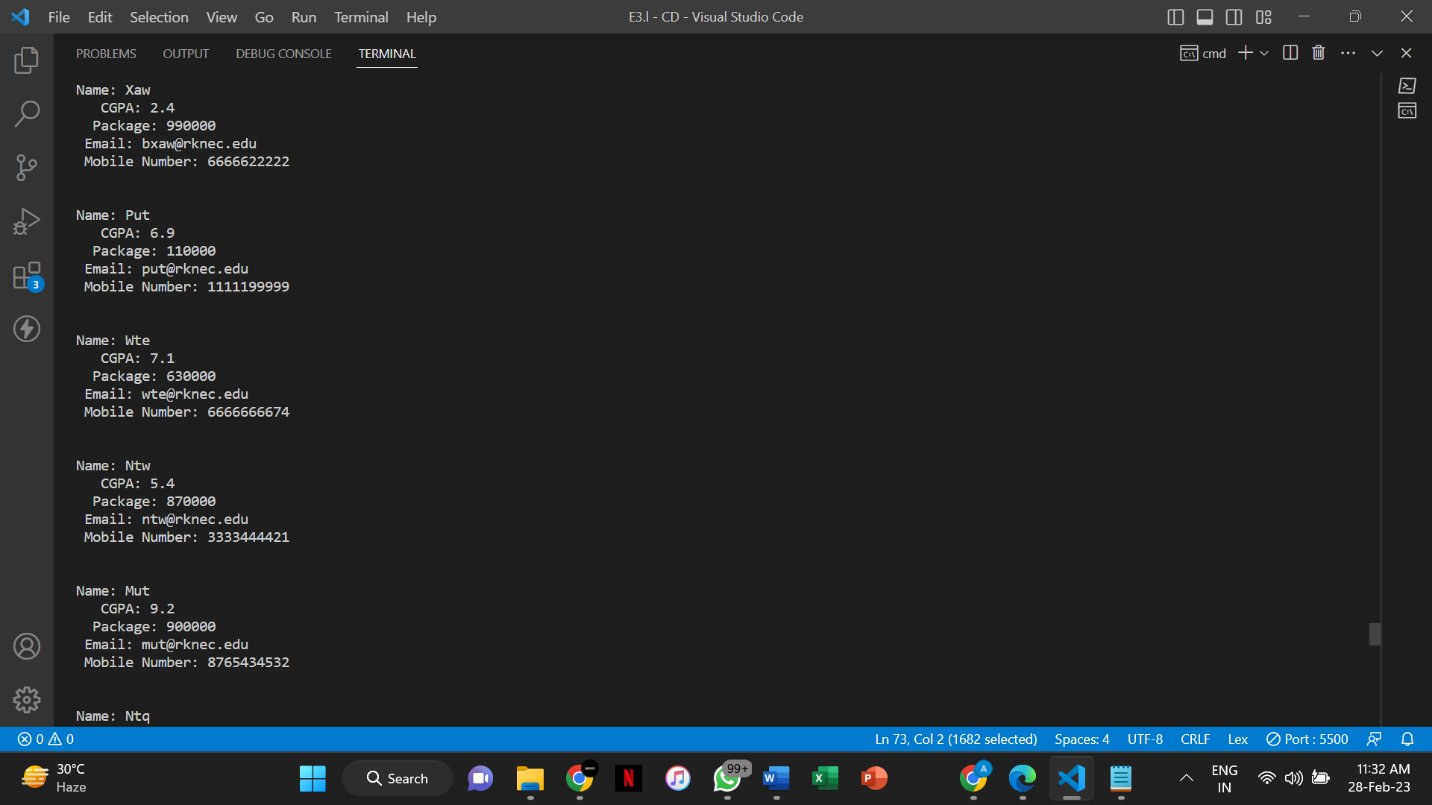
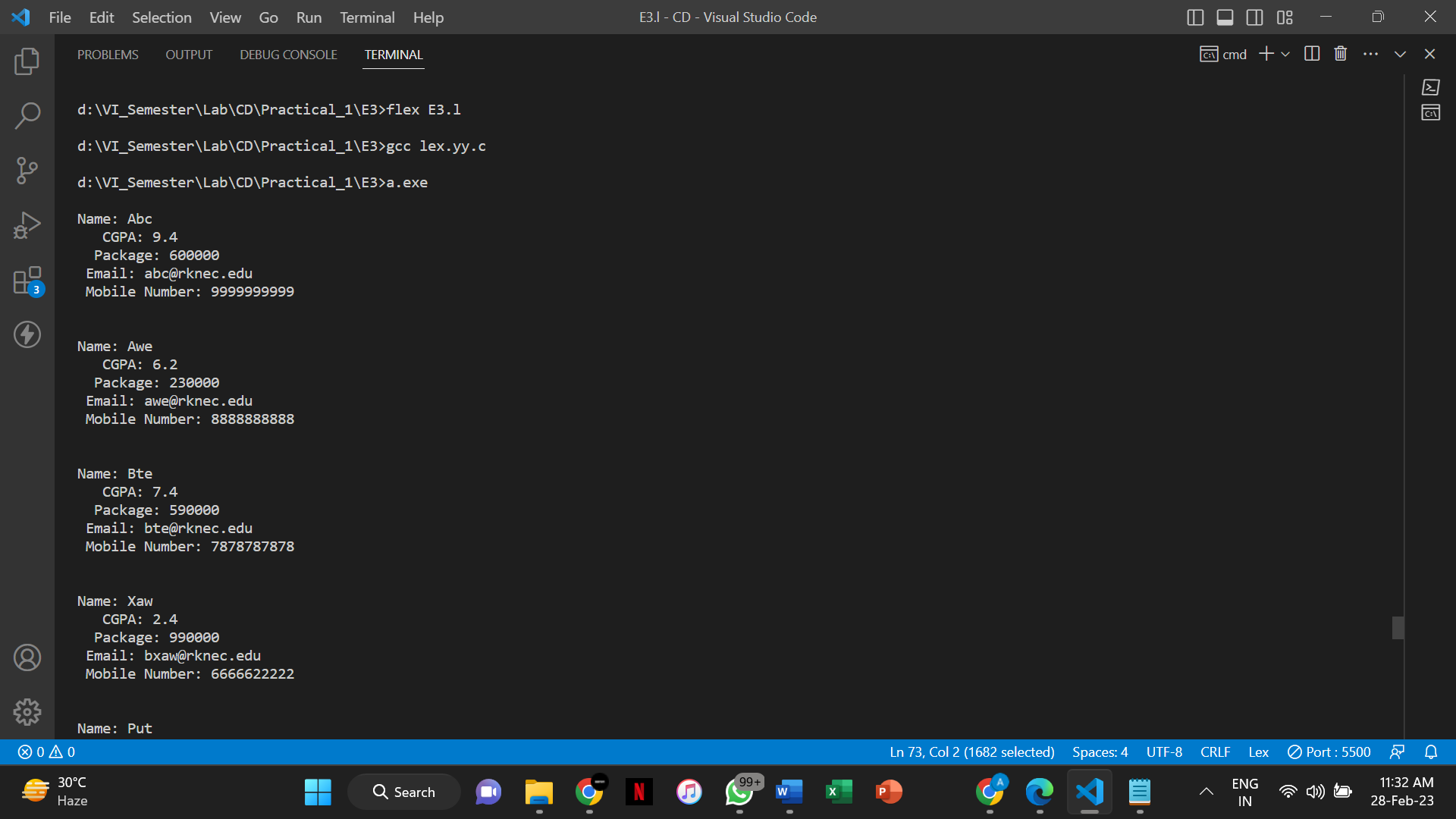
Niw Accenture Male 9.9 EC 230000 niw@rknec.edu 7874563545

Gut Accenture Female 4.8 CSE 220000 gut@rknec.edu 8875874656

Bpl Wipro Female 7.3 CSE 110000 bpl@rknec.edu 8745635452

Asp TCS Male 8.6 EC 690000 asp@rknec.edu 4536245345

**Output:**

****