MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

LAB MANUAL- M. Tech., [CSE]

ADAVANCED SYSTEM SOFTWARE

Department of Computer Science and Engineering

Compiled by

Dr. NARENDRA V G
Associate Professor – Senior Scale

January 2020

INSTRUCTIONS TO THE STUDENTS

Pre- Lab Session Instructions

1. Students should carry the Class notes, Lab Manual and the required stationery to every lab session
2. Be in time and follow the Instructions from Lab Instructors
3. Must Sign in the log register provided
4. Make sure to occupy the allotted seat and answer the attendance
5. Adhere to the rules and maintain the decorum

In- Lab Session Instructions

- Follow the instructions on the allotted exercises given in Lab Manual
- Show the program and results to the instructors on completion of experiments
- On receiving approval from the instructor, copy the program and results in the Lab record
- Prescribed textbooks and class notes can be kept ready for reference if required

General Instructions for the exercises in Lab

- The programs should meet the following criteria:

o Programs should be interactive with appropriate prompt messages, error messages if any, and descriptive messages for outputs.

o Use meaningful names for variables and procedures.

- Copying from others is strictly prohibited and would invite severe penalty during evaluation.
- The exercises for each week are divided under three sets:

o Lab exercises – to be completed during lab hours

o Additional questions – to be completed outside the lab or in the lab to enhance the skill

- In case a student misses a lab class, he/ she must ensure that the experiment is completed at students end or in a repetition class (if available) with the permission of the faculty concerned but credit will be given only to one day's experiment(s).
- Questions for lab tests and examination are not necessarily limited to the questions in the manual, but may involve some variations and / or combinations of the questions.

THE STUDENTS SHOULD NOT...

1. Bring mobile phones or any other electronic gadgets to the lab.
2. Go out of the lab without permission.

Course Objectives

• To implement the various parsing techniques.

• To implement the various distributed systems algorithms such as Logical clocks, Mutual exclusion, Election Algorithms, Chat Server and Client etc.

Course Outcomes

At the end of this course, students will have the

• Ability to write code along the lines of the algorithm.

• Ability to carry out a project to solve a given problem.

Evaluation Plan

- Internal Assessment Marks : 60 Marks

Python Programming	:	10 Marks
Lexical analyzer	:	10 Marks
Parser	:	15 Marks
Logical Clocks	:	05 Marks
Election Algorithms	:	10 Marks
Chat Server and Client	:	10 Marks

- End semester assessment : 40 Marks

✓ Duration: 2 hours

✓ Total marks : Write up : 15 Marks

Execution : 25 Marks

Note: Finally, total marks will be reduced to 50 Marks

CONTENTS

Sl. No.	Experiments	Page No.
1	Week 01 : Basics of Python Programming [Basics;Conditional and looping statements; String; Lists]	05
2	Week 02: Basics of Python Programming (contd…) [Dictionaries; Functions; Text files]	07
3	Week 03: Basics of Python Programming (contd…) [OOP; Regular expressions]	09
4	Week 04: Lexical Analyser [Introduction to PLY]	10
5	Week 05: Syntax Analysis	11
6	Week 06: Recursive Descent Parser for Decision Making and Looping Statements	13
7	Week 07: Network Programming [TCP Socket Programming]	14
8	Week 08: Network Programming [UDP Socket Programming]	15
9	Week 09: Logical clocks	16
10	Week 10: Election Algorithms	17
11	Week 11: Chat Server and Client	18
12	Week 12: Chat Server and Client (contd..)	18

Week 01 : Basics of Python Programming [Basics;Conditional and looping statements; String; Lists]

Q.1) The Fibonacci numbers are the sequence below, where the first two numbers are 1, and each number thereafter is the sum of the two preceding numbers. Write a program that asks the user how many Fibonacci numbers to print and then prints that many. [use while and for loop] 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 . . .

Q. 2) Write a Python program that computes the factorial of a number. The factorial, *n*!, of a number *n* is the product of all the integers between 1 and *n*, including *n*.
For instance, 5! = 1 x 2 x 3 x 4 x 5 = 120. [use while and for loop]

Q. 3) Write a Python program that asks the user to enter a string. The program should then print the following:
(a) The total number of characters in the string
(b) The string repeated 10 times
(c) The first character of the string (remember that string indices start at 0)
(d) The first three characters of the string
(e) The last three characters of the string
(f) The string backwards
(g) The seventh character of the string if the string is long enough and a message otherwise
(h) The string with its first and last characters removed
(i) The string in all caps
(j) The string with every a replaced with an e
(k) The string with every letter replaced by a space

Q.4) At a certain school, student email addresses end with @student.college.edu, while professor email addresses end with @prof.college.edu. Write a Python program that first asks the user how many email addresses they will be entering, and then has the user enter those addresses. After all the email addresses are entered, the program should print out a message indicating either that all the addresses are student addresses or that there were some professor addresses entered.

Q.5) Start with the list [8,9,10]. Write a Python program to do the following:
(a) Set the second entry (index 1) to 17
(b) Add 4, 5, and 6 to the end of the list
(c) Remove the first entry from the list
(d) Sort the list
(e) Double the list
(f) Insert 25 at index 3
The final list should equal [4,5,6,25,10,17,4,5,6,10,17]

Q.6) Write a Python program that asks the user to enter some text and then counts how many articles are in the text. Articles are the words 'a', 'an', and 'the'.

Additional questions:

Q.1) A number is called a perfect number if it is equal to the sum of all of its divisors, not including the number itself. For instance, 6 is a perfect number because the divisors of 6 are 1, 2, 3, 6 and 6 = 1 + 2 + 3. As another example, 28 is a perfect number because its divisors are 1, 2, 4,7, 14, 28 and 28 = 1 + 2 + 4 + 7 + 14. However, 15 is not a perfect number because its divisors are 1, 3, 5, 15 and 15 6= 1 + 3 + 5. Write a Python program that finds all four of the perfect numbers that are less than 10000.

Q.2) A simple way of encrypting a message is to rearrange its characters. One way to rearrange the characters is to pick out the characters at even indices, put them first in the encrypted string, and follow them by the odd characters. For example, the string message would be encrypted as msaeesg because the even characters are m, s, a, e (at indices 0, 2, 4, and 6) and the odd characters are e, s, g (at indices 1, 3, and 5).

(a) Write a Python program that asks the user for a string and uses this method to encrypt the string.

(b) Write a Python program that decrypts a string that was encrypted with this method.

Q.3) Write a Python program that asks the user to enter a length. The program should ask them what unit the length is in and what unit they would like to convert it to. The possible units are inches, yards, miles, millimeters, centimeters, meters, and kilometers. While this can be done with 25 if statements, it is shorter and easier to add on to if you use a two-dimensional list of conversions, so please use lists for this problem.

Week 02: Basics of Python Programming (contd…) [Dictionaries; Functions; Text files]

Q.1) Write a Python program that asks the user to enter a word. Rearrange all the letters of the word in alphabetical order and print out the resulting word. For example, abracadabra should become aaaaabbcdrr.

Q.2) Write a Python program that asks the user to enter a date in the format mm/dd/yy and converts it to a more verbose format. For example, 02/04/77 should get converted into February 4, 1977.

Q.3) Write a Python program that uses a dictionary that contains ten user names and passwords. The program should ask the user to enter their username and password. If the username is not in the dictionary, the program should indicate that the person is not a valid user of the system. If the username is in the dictionary, but the user does not enter the right password, the program should say that the password is invalid. If the password is correct, then the program should tell the user that they are now logged in to the system.

Q.4) You are given a file called students.txt. A typical line in the file looks like:

Walter melon melon@email.msmary.edu 555-3141

There is a name, an email address, and a phone number, each separated by tabs. Write a Python program that reads through the file line-by-line, and for each line, capitalizes the first letter of the first and last name and adds the area code 301 to the phone number. Your program should write this to a new file called students2.txt. Here is what the first line of the new file should look like:

Walter Melon melon@email.msmary.edu 301-555-3141

Q.5) The digital root of a number *n* is obtained as follows: Add up the digits *n* to get a new number. Add up the digits of that to get another new number. Keep doing this until you get a number that has only one digit. That number is the digital root.

For example, if *n* = 45893, we add up the digits to get 4 + 5 + 8 + 9 + 3 = 29. We then add up the digits of 29 to get 2 + 9 = 11. We then add up the digits of 11 to get 1 + 1 = 2. Since 2 has only one digit, 2 is our digital root.

Write a function in Python that returns the digital root of an integer *n*. [Note: there is a shortcut, where the digital root is equal to *n* mod 9, but do not use that here.]

Additional Questions:

Q.1) Write a Python script, which is repeatedly ask the user to enter a team name and the how many games the team won and how many they lost. Store this information in a dictionary where the keys are the team names and the values are lists of the form [wins, losses].

(a) Using the dictionary created above, allow the user to enter a team name and print out the team's winning percentage.

(b) Using the dictionary, create a list whose entries are the number of wins of each team.

(c) Using the dictionary, create a list of all those teams that have winning records.

Q.2) You are given a file namelist.txt that contains a bunch of names. Write a Python program to print out all the names in the list in which the vowels a, e, i, o, and u appear in order (with repeats possible). The first vowel in the name must be a and after the first u, it is okay for there to be other vowels. An example is Ace Elvin Coulson.

Q.3) Write a function in Python called merge that takes two already sorted lists of possibly different lengths, and merges them into a single sorted list.
(a) Do this using the sort method.
(b) Do this without using the sort method.

Week 03: Basics of Python Programming (contd…) [OOP; Regular expressions]

Q.1) Write a class in Python called Password_manager. The class should have a list called old_passwords that holds all of the user's past passwords. The last item of the list is the user's current password. There should be a method called get_password that returns the current password and a method called set_password that sets the user's password. The set_password method should only change the password if the attempted password is different from all the user's past passwords. Finally, create a method called is_correct that receives a string and returns a boolean **True** or **False** depending on whether the string is equal to the current password or not.

Q.2) Write a class in Pthon called Wordplay. It should have a field that holds a list of words. The user of the class should pass the list of words they want to use to the class. There should be the following methods:

• words_with_length(length)— returns a list of all the words of length length
• starts_with(s)— returns a list of all the words that start with s
• ends_with(s)— returns a list of all the words that end with s
• palindromes()— returns a list of all the palindromes in the list
• only(L)— returns a list of the words that contain only those letters in L
• avoids(L)— returns a list of the words that contain none of the letters in L

Q.3) Write the regular expressions in Python for the following:

i) To replace all occurrences of abc with *.
For example 'abcdef abcxyz' replace to ('abc', ' * ')
ii) To convert Roman numerals into ordinary numbers.
iii) To take a date in a verbose format, like February 6, 2011, and convert it an abbreviated format, mm/dd/yy (with no leading zeroes).
iv) To recognize the strings: "bat," "bit," "but," "hat," "hit," or "hut."
v) To match the set of all valid Python identifiers.

Additional Questions:

Q.1) Write a program in PYTHON, that removes single and multiline comments from a given input 'C' file.

Q.2) Write a program in PYTHON, that takes a file as input and replaces blank spaces and tabs by single space and writes the output to a file.

Week 04: Lexical Analyser [Introduction to PLY]

Q.1) Write a Program in Python, to Design Lexical Analyzer.

Q.2) Write a program in Python, to identify the relational operators from the given input 'C' file.

Additional questions:

Q.1) 1. Write a getNextToken ( ) to generate tokens for the perl script given below.

```
#! /usr/bin/perl
#get total number of arguments passed.
$n = scalar (@_);
$sum = 0;
foreach $item(@_) {
$sum += $item;
}
$average = $sum + $n;
```

#! Represents path which has to be ignored by getNextToken().

followed by any character other than ! represents comments.
$n followed by any identifier should be treated as a single token.
Scalar, foeach are considered as keywords.
@_, += are treated as single tokens.

Remaining symbols are tokenized accordingly.

Week 05: Syntax Analysis

Q.1) Write a simple desk calculator in Python,that reads an arithmetic expression, evaluates it, and then prints its numeric value. We shall build the desk calculator starting with the following grammar for arithmetic expressions:

i) E→ E+E | E – E | E * E | E/E | - E | number

The token number is a single digit between 0 and 9.

ii) E→ E+T | T

T → T*F | F

F → (E) | digit

The token digit is a single digit between 0 and 9.

Additional Questions:

*RECURSIVE DESCENT PARSER FOR C GRAMMAR***:**

A simple 'C' language grammar is given. Student should write/update RD parser for subset of grammar each week and integrate it lexical analyzer. Before parsing the input file, remove ambiguity and left recursion, if present and also perform left factoring on subset of grammar given. Include the functions first(X) and follow(X) which already implemented in previous week. Lexical analyzer code should be included as header file in parser code. Parser program should make a function call getNextToken() of lexical analyze which generates a token. Parser parses it according to given grammar. The parser should report syntax errors if any (for e.g.: Misspelling an identifier or keyword, Undeclared or multiply declared identifier, Arithmetic or Relational Expressions with unbalanced parentheses and Expression syntax error etc.) with appropriate line-no.

Sample C grammar:

Data Types	:	int, char
Arrays	:	1-dimensional
Expressions	:	Arithmetic and Relational
Looping statements	:	for, while
Decision statements	:	if, if – else

Note: The following grammar for C language is to be adopted with necessary corrections

Program → main () { declarations statement-list }

declarations→ data-type identifier-list; declarations | ∈

data-type → int | char

identifier-list → id | id, identifier-list | id[number] , identifier-list | id[number]

statement_list → statement statement_list | ∈

statement → assign-stat; | decision_stat | looping-stat

assign_stat → id = expn

expn→ simple-expn eprime
eprime→relop simple-expn|∈
simple-exp→ term seprime
seprime→addop term seprime |∈
term → factor tprime
tprime → mulop factor tprime |∈
factor → id | num
decision-stat → if (expn) {statement_list} dprime
dprime → else {statement_list} | ∈
looping-stat → while (expn) {statement_list} | for (assign_stat ; expn ; assign_stat) {statement_list}
relop → = = | != | <= | >= | > | <
addop → + | -
mulop → * | / | %

Grammar 5.1

Q.1) For given subset of grammar 5.1, to design Recursive Descent Parser in Python.

Program → main () { declarations statement-list }
identifier-list → id | id, identifier-list | id[number] , identifier-list | id[number]
statement_list → statement statement_list | ∈
statement → assign-stat;
assign_stat → id = expn
expn→simple-expn eprime
eprime→relop simple-expn|∈
simple-exp→ term seprime
seprime→addop term seprime |∈
term → factor tprime
tprime → mulop factor tprime |∈
factor → id | num
relop → = = | != | <= | >= | > | <
addop → + | -
mulop → * | / | %

Week 06: RECURSIVE DESCENT PARSER FOR DECISION MAKING AND LOOPING STATEMENTS

Q.1) For given subset of grammar 5.1, design Recursive Descent parser.

statement → assign-stat; | decision_stat | looping-stat

decision-stat → if (expn) {statement_list} dprime

dprime → else {statement_list} | ∈

looping-stat → while (expn) {statement_list} | for (assign_stat; expn ; assign_stat) {statement_list}

Additional Questions:

Q.1) Modify the Recursive Descent parser to handle compound expressions present in Python program.

Week 07: Network Programming [TCP Socket Programming]

Internetworking and Sockets. Implement the sample TCP client/ server programs found in the Python Library Reference documentation on the socket module and get them to work. Set up the server and then the client.

You decide the server is too boring. Update the server so that it can do much more, recognizing the following commands:

date Server will return its current date/timestamp, that is, time.ctime().

os Get OS information (os.name).

ls Give a listing of the current directory. (Hints: os.listdir() lists a directory, os.curdir is the current directory.) Extra Credit: Accept ls dir and return dir's file listing.

You do not need a network to do this assignment—your computer can communicate with itself. Be aware that after the server exits, the binding must be cleared before you can run it again. You might experience "port already bound" errors. The operating system usually clears the binding within 5 minutes, so be patient.

Algorithm:

SERVER SIDE PROGRAMMING ALGORITHM

STEP 1: Start the program.

STEP 2: Declare the variables and structure for the socket.

STEP 3: The socket is binded at the specified port.

STEP 4: Using the object the port and address are declared.

STEP 5: The listen and accept functions are executed.

STEP 6: If the binding is successful it waits for client request.

STEP 7: Execute the client program.

CLIENT DAY TIME ALGORITHM

STEP 1: Start the program.

STEP 2: Declare the variables and structure.

STEP 3: Socket us created and connect function is executed.

STEP 4: If the connection is successful then server sends the message.

STEP 5: The date and time is printed at the client side.

STEP 6: Stop the program.

Additional Questions:

Q.1) Half-Duplex Chat. Create a simple, half-duplex chat program. By half-duplex, we mean that when a connection is made and the service starts, only one person can type. The other participant must wait to get a message before being prompted to enter a message. Once a message is sent, the sender must wait for a reply before being allowed to send another message. One participant will be on the server side; the other will be on the client side.

Week 08: Network Programming [UDP Socket Programming]

Q.1) Daytime Service. Use the socket.getservbyname() to determine the port number for the "daytime" service under the UDP protocol. Check the documentation for getservbyname() to get the exact usage syntax (i.e., socket.getservbyname.__doc__). Now write an application that sends a dummy message over and wait for the reply. Once you have received a reply from the server, display it to the screen.

Additional questions:

Q.1) *Sleep Server*. Create a sleep server. A client will request to be "put to sleep" for a number of seconds. The server will issue the command on behalf of the client then return a message to the client indicating success. The client should have slept or should have been idle for the exact time requested. This is a simple implementation of a remote procedure call, where a client's request invokes commands on another computer across the network.

Week 09: Logical clocks

Q.1) Simulate the Lamport's Logical Clocks in Python

Lamport Timestamps algorithm

A Lamport logical clock is an incrementing counter maintained in each process. Conceptually, this logical clock can be thought of as a clock that only has meaning in relation to messages moving between processes. When a process receives a message, it resynchronizes its logical clock with that sender (causality).

The algorithm of Lamport Timestamps can be captured in a few rules:

- All the process counters start with value 0.
- A process increments its counter for each event (internal event, message sending, message receiving) in that process.
- When a process sends a message, it includes its (incremented) counter value with the message.
- On receiving a message, the counter of the recipient is updated to the greater of its current counter and the timestamp in the received message, and then incremented by one.

Additional Questions:

Q.2) Simulate the Lamport's Logical Clocks in Python

Week 10: Election Algorithms

Q.1) Simulate the Bully Algorithm in Python.

Algorithm – Suppose process P sends a message to the coordinator.

- If coordinator does not respond to it within a time interval T, then it is assumed that coordinator has failed.
- Now process P sends election message to every process with high priority number.
- It waits for responses, if no one responds for time interval T then process P elects itself as a coordinator.
- Then it sends a message to all lower priority number processes that it is elected as their new coordinator.

However, if an answer is received within time T from any other process Q,

(i) Process P again waits for time interval T' to receive another message from Q that it has been elected as coordinator.

(ii) If Q doesn't responds within time interval T' then it is assumed to have failed and algorithm is restarted.

Additional questions:

Q.1) Simulate the Ring Algorithm in Python.

Algorithm – If process P1 detects a coordinator failure, it creates new active list which is empty initially. It sends election message to its neighbour on right and adds number 1 to its active list.

If process P2 receives message elect from processes on left, it responds in 3 ways:

(i) If message received does not contain 1 in active list then P1 adds 2 to its active list and forwards the message.

(ii) If this is the first election message it has received or sent, P1 creates new active list with numbers 1 and 2. It then sends election message 1 followed by 2.

(iii) If Process P1 receives its own election message 1 then active list for P1 now contains numbers of all the active processes in the system. Now Process P1 detects highest priority number from list and elects it as the new coordinator.

Week 11 and 12: Chat Server and Client

Q.1) Implement the distributed chat server and client in python.

REFERENCES:

1. Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman, Compilers Principles, Techniques and Tools, Pearson Education, 2nd Edition, 2010.
2. D M Dhamdhere, "Systems Programming and Operating Systems", Tata McGraw Hill, 2nd Revised Edition, 2001.
3. Kenneth C. Louden, "Compiler Construction- Principles and Practice", Thomson, India Edition, 2007.
4. "Keywords and Identifiers",https://www.programiz.com/c-programming/c-keywords-identifier.
5. Andrew S. Tannenbaum, Maarteen Van Steen: "Distributed Systems, Principles and Paradigms", (3e), Version 3.01, 2017.
6. David M. Beazley, PLY (Python Lex-Yacc)
7. Francesco Pierfederici, "Distributed Computing with Python ", PACKT Publishing, BIRMINGHAM - MUMBAI
8. Wesley J Chun, " Core PYTHON Applications Programming", 3e, Prtice Hall, New York, 2012.
9. Vernon L. Ceder, " The Quick Python Book", Second Edition, Manning Publications Co., 2010.