SYSTEM SOFTWARE LABORATORY

Lab Hours/ Week	: 3.0	Credits :	1.5
Sub. Code	: 6CSL02	CIE Marks :	50
		SEE Marks :	50

PART-A

Execute the following programs using LEX and YACC:

1. a) Write a Lex program to count the number of characters, words, tabs, spaces, lines and operators in a given .C file.

   b) Write a Lex program to count the number of vowels and consonants in a given input string.
2. a) Write a Lex program to count the number of
   i) Positive and negative integers,
   ii) Positive and negative fractions (ex. a/b),
   iii) Positive and negative decimals (ex. a.b),
   Display error messages for the rest.

   b) Write a Lex program to count number of 'scanf' and 'printf' statements in a C program. Replace them with 'readf' and 'writef' statements respectively in a separate file.
3. Write a Lex program to count the number of comment lines in a given c program, display the count of single line and multiline comments separately. Also copy the program by eliminating comments onto another file.
4. Write a Lex program to recognize a valid arithmetic expression. Identify & print the identifiers and operators separately.
5. Write a Lex program to recognize and count the number of identifiers in a given C program.
6. a) Write a YACC program to recognize valid arithmetic expression that uses operators +,-,* and /.

   b) Write a YACC program to recognize a valid variable, which starts with a letter, followed by any number of letters or digits.
7. Write a YACC program to evaluate an arithmetic expression involving operators +,-,* and /.

8. Write a YACC program to recognize strings 'aabb', 'aaabbb', and 'ab' using the grammar (a^n b^n, n>=0).

9. Write a YACC program to recognize a string defined by the grammar (a^n b, n>=10).

10. Write a YACC program to recognize a string defined by the grammar (a^m b^n, m>n).

PART-B

Execute the following programs using C:

1. Write a program to demonstrate the following system calls of UNIX operating system: fork(), exec(), getpid(), getppid(), exit(), wait(), close(), getpriority().

2. Write a program to demonstrate the system calls for file handling mechanism such as: opendir(), readdir(), open(), read(), write().

3. Given the list of processes, their CPU burst times and arrival times, display/print the Gantt chart for preemptive SJF scheduling technique. Also compute and print the average waiting time and average turnaround time.

4. Given the list of processes, their CPU burst times and arrival times, display/print the Gantt chart for Round robin scheduling technique. Also compute and print the average waiting time and average turnaround time.

5. Program to implement Bankers algorithm to demonstrate deadlock detection and avoidance.

6. Develop an Application using the following Inter Process Communications

i. pipes
ii. shared memory

7. Implement the Producer – Consumer problem using semaphores.

8. Implement LRU paging mechanism for the given reference string and number of frames. Display the number of page faults happened.

9. Demonstrate the following memory management schemes for given hole and a list of processes.

i) First fit

ii) Best fit

iii) Worst fit.

10. Implement contiguous file allocation technique.

Example for exercises 9: Free space is maintained as a linked list of nodes with each node having the starting byte address and the ending byte address of a free block. Each memory request consists of the process-id and the amount of storage space required in bytes. Allocated memory space is again maintained as a linked list of nodes with each node having the process-id, starting byte address and the ending byte address of the allocated space. When a process finishes (taken as input) the appropriate node from the allocated list should be deleted and this free disk space should be added to the free space list. [Care should be taken to merge contiguous free blocks into one single block. This results in deleting more than one node from the free space list and changing the start and end address in the appropriate node]. For allocation use first fit, worst fit and best fit.

SOFTWARE REQUIREMENT:

Linux: Ubuntu / OpenSUSE / Fedora / Red Hat / Debian / Mint OS

CIE ASSESSEMENT

SPLIT UP OF CIE (35+15=50 marks)

- **ASSESSEMENT FOR 35 MARKS**

  Lab in charge should maintain the batch-wise assessment sheet and every student should be evaluated in all lab sessions for 35 marks (split up given below)

  - Preparedness and Writing program – 10 marks
  - Execution of program - 15 marks
  - Record and observation book – 5 marks
  - Viva-5 marks

- **ASSESSEMENT FOR 15 MARKS**

  Two tests should be conducted; each carrying 15 marks and average of two tests will be taken (split up given below)

  - Program write up - 4 marks
  - Execution – 8 marks
  - Viva - 3 marks

SEMESTER END EXAMINATIONS ASSESSMENT:

A student has to execute one program from each part for total of 50 marks. There are three parts, PART-A and PART-B and weight ages for these parts are as follows:

- PART-A - 30% of total marks(15 marks)
- PART-B - 60% of total marks(30 marks)
- Viva - 10% of total marks (5 marks)

Change of question is not permitted in one single part. If student wants a change of question, once again he/she should be given a choice in both of the parts. Change of question will be evaluated by deducting 20% of total marks (40 marks).