PIMPRI CHINCHWAD COLLEGE OF ENGINEERING & RESEARCH

DEPARTMENT OF COMPUTER ENGINEERING

SUBJECT CODE: 310248

LAB MANUAL
Laboratory Practice-I
(System Programming & Operating System)

Semester – I, Academic Year: 2021-22

Pimpri Chinchwad College of Engineering & Research,

Department of Computer Engineering

310248: Laboratory Practice-I

Teaching Scheme:
Practical: 4 Hrs/Week
Credits: 02

Examination Scheme:
Term work: 25 Marks
Practical: 25 Marks

List of Laboratory Assignments

C308.1	To design various System Software using suitable data structure.
C308.2	To implement scheduling policies and memory management concepts of operating system.
C308.3	Apply the principles of HCI to design interactive user interface.
C308.4	Apply and analyze GOMS model for suitable application.

Sr. No.	Group A	Page No.	
1	Design suitable data structures and implement pass-I of a two-pass assembler for Pseudo-machine in Java using object oriented feature. Implementation should consist of a few instructions from each category and few assembler directives.	5	C308.1
2	Implement Pass-II of two pass assembler for pseudo-machine in Java using object oriented features. The output of assignment-1 (intermediate file and symbol table) should be input for this assignment.	9	C308.1
3	Design suitable data structures and implement pass-I of a two-pass macro-processor using OOP features in Java	13	C308.1
4	Write a Java program for pass-II of a two-pass macro-processor. The output of Assignment-3 (MNT, MDT and file without any macro definitions) should be input for this assignment.	16	C308.1
	Group B		
7	Write a program to simulate CPU scheduling algorithms: FCFS , SJF (Preemptive), Priority (Non-Preemptive) and Round Robin (Preemptive)	25	C308.2
9	Write a program to simulate page replacement algorithms using 1. FIFO 2. Least Recently Used (LRU) 3.Optimal algorithm	31	C308.2
10	Bankers Algorithm (Deadlock Avoidance Algorithm)		C308.2
	Part II : Elective II Group B		

1.	Design a paper prototype for selected Graphical User Interface.		C308.3
2.	Implement GOMS (Goals, Operators, Methods and Selection rules) modeling technique to model user's behavior in given scenario.		C308.4
3.	Design a User Interface in Python.		C308.3
4.	To redesign existing Graphical User Interface with screen complexity.		C308.3

Instructions

Guidelines for Student's Laboratory Journal

The laboratory assignments are to be submitted by student in the form of journal. Journal consists of Certificate, table of contents, and handwritten write-up of each assignment (Title, Date of Completion, Objectives, Problem Statement, Software and Hardware requirements, Assessment grade/marks and assessor's sign, Theory- Concept in brief, algorithm, flowchart, test cases, Test Data Set(if applicable), mathematical model (if applicable), conclusion/analysis. Program codes with sample output of all performed assignments are to be submitted as softcopy. As a conscious effort and little contribution towards Green IT and environment awareness, attaching printed papers as part of write-ups and program listing to journal must be avoided. Use of DVD containing students programs maintained by Laboratory In-charge is highly encouraged. For reference one or two journals may be maintained with program prints in the Laboratory.

Guidelines for Laboratory /Term Work Assessment

Continuous assessment of laboratory work should be based on overall performance of Laboratory assignments by a student. Each Laboratory assignment assessment will assign grade/marks based on parameters, such as timely completion, performance, innovation, efficient codes, punctuality and

Guidelines for Practical Examination

Problem statements must be decided jointly by the internal examiner and external examiner. During practical assessment, maximum weightage should be given to satisfactory implementation of the problem statement. Relevant questions may be asked at the time of evaluation to test the student's understanding of the fundamentals, effective and efficient implementation. This will encourage, transparent evaluation and fair approach, and hence will not create any uncertainty or doubt in the minds of the students. So, adhering to these principles will consummate our team efforts to the promising start of student's academics.

Guidelines for Laboratory Conduction

The instructor is expected to frame the assignments by understanding the prerequisites, technological aspects, utility and recent trends related to the topic. The assignment framing policy need to address the average students and inclusive of an element to attract and promote the intelligent students. Use of open source software is encouraged. Based on the concepts learned. Instructor may also set one assignment or mini-project that is suitable to respective branch beyond the scope of syllabus. For the elective subjects students should form group of 3-4 students. The faculty coordinator will take care that all the assignment should be assigned to class and minimum two assignments are compulsory for each group.

Programming tools recommended: -

Human computer Interface-GUI in python

Internet of Things and Embedded System- Raspberry Pi/Arduino Programming; Arduino IDE/Python Interfacing. Other IoT devices

Software project management-MS project/Gantt Project/Primavera

Assignment No.: 01

Problem Statement: Design suitable data structures and implement pass-I of a two-pass assembler for pseudo machine in Java/C++ using object oriented feature. Implementation should consist of a few instructions from each category and few assembler directives.

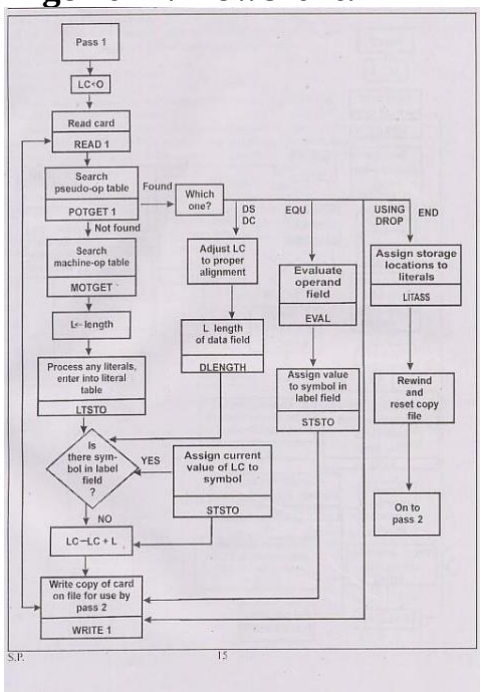
Objectives:

1. To study the design and implementation of 1st pass of two pass assembler.
2. To study the categorized instruction set of assembler.
3. To study the data structure used in assembler implementation.

Theory:

1. Explain various Data and Instruction formats of assembly language programming.
2. Explain the design of Pass- I of assembler with the help of flowchart and example.
3. Discuss various Data structure used in Pass-I along with its format and significance of each field.

Algorithm/Flowchart:



Design diagrams (if any):

1. Class Diagram
2. Use case Diagram
3. ER Diagram

Input:

Source code of Assembly Language

```
SAMPLE    START    100
          USING    *, 15
          L        1, FOUR
```

```
                A       1, =F'3'
                ST      1, RESULT
                SR      1, 2
                LTORG
                L       2, FIVE
                A       2, =F'5'
                A       2, =F'7'
FIVE            DC      F'5'
FOUR            DC      F'4'
RESULT          DS      1F
                END
```

Output:

```
100     SAMPLE  START   100
100             USING   *, 15
100             L       1, FOUR
104             A       1, =F'3'
108             ST      1, RESULT
112             SR      1, 2
114             LTORG
124             L       2, FIVE
128             A       2, =F'5'
132             A       2, =F'7'
136     FIVE    DC      F'5'
140     FOUR    DC      F'4'
144     RESULT  DS      1F
152             5
156             7
160             END
```

Machine Opcode Table (MOT)

Mnemonic	Hex / Binary Code	Length (Bytes)	Format
L	5A	4	RX
A	1B	4	RX
ST	50	4	RX
SR	18	2	RR

Pseudo Opcode Table (POT)

Pseudo op	Address / Name of Procedure to implement pseudo operation
START	PSTART
USING	PUSING
DC	PDC
DS	PDS
LTORG	PLTORG
END	PEND

Symbol Table (ST)

Sr. No	Symbol name	Address	Value	Length	Relocation
1	SAMPLE	100	--	160	R
2	FIVE	136	5	4	R
3	FOUR	140	4	4	R
4	RESULT	144	__	4	R

Literal Table (LT)

Sr. No	Literal	Address	Length
1	3	120	4
2	5	152	4
3	7	156	4

Instructions :

Not specific

Test Cases:

1. Check syntax of instruction (Correct and wrong)
2. Symbol not found
3. Wrong instruction
4. Duplicate symbol declaration
5. Test the output of program by changing value of START pseudo opcode.
6. Test the output of program by changing position of LTORG pseudo-op.

Software Requirement:

1. Fedora
2. Eclipse
3. JDK

Hardware Requirement:

Not specific

Frequently Asked Questions:

1. What is two pass assembler?
2. What is the significance of symbol table?
3. Explain the assembler directives EQU, ORIGIN.
4. Explain the assembler directives START, END, LTORG.
5. What is the use of POOLTAB and LITTAB?
6. How literals are handled in pass I?
7. What are the tasks done in Pass I?
8. How error handling is done in pass I?
9. Which intermediate data structures are designed and implemented in PassI?
10. What is the format of a machine code generated in PassII?
11. What is forward reference? How it is resolved by assembler?
12. How error handling is done in pass II?
13. What is the difference between IS, DL and AD?

14. What are the tasks done in Pass II?

Conclusion:

Input assembly language program is processed by applying Pass-I algorithm of assembler and intermediate data structures, Symbol Table, Literal Table, MOT, POT, BT, etc. are generated.

Assignment No.: 02

Problem Statement:

Implement Pass-II of two pass assembler for pseudo-machine in Java/C++ using object oriented features. The output of assignment-1 (intermediate file and symbol table) should be input for this assignment.

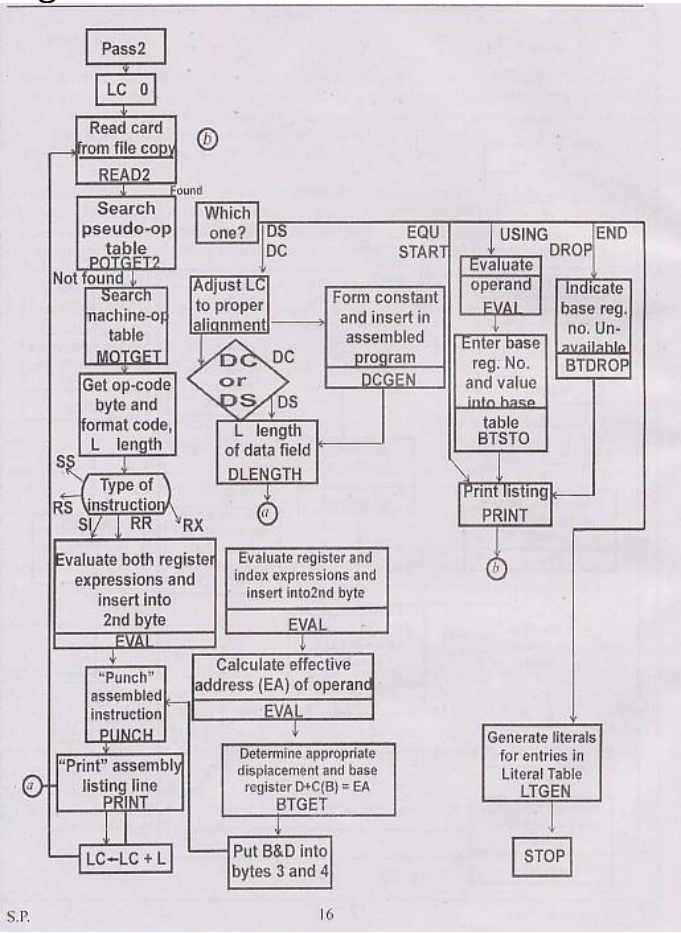
Objectives:

1. To study the design and implementation of 2nd pass of two pass assembler.
2. To study the data structure used in Pass-2 of assembler implementation.

Theory:

1. Explain the design of Pass- II of assembler with the help of flowchart and example.

Algorithm/Flowchart:



Design diagrams (if any):

1. Class Diagram
2. Use case Diagram
3. ER Diagram

Input:

Intermediate code of pass-1.

LC LABEL INSTR. OPERANDS

```
-------------------------------------------------------------
100     SAMPLE      START       100
100                 USING       *, 15
100                 L           1, FOUR
104                 A           1, =F'3'
108                 ST          1, RESULT
112                 SR          1, 2
114                 LTORG
124                 L           2, FIVE
128                 A           2, =F'5'
132                 A           2, =F'7'
136     FIVE        DC          F'5'
140     FOUR        DC          F'4'
144     RESULT      DS          1F
152                 5
156                 7
160                 END
```

Machine Opcode Table (MOT)

Mnemonic	Hex / Binary Code	Length (Bytes)	Format
L	5A	4	RX
A	1B	4	RX
ST	50	4	RX
SR	18	2	RR

Pseudo Opcode Table (POT)

Pseudo op	Address / Name of Procedure to implement pseudo operation
START	PSTART
USING	PUSING
DC	PDC
DS	PDS
LTORG	PLTORG
END	PEND

Symbol Table (ST)

Sr. No	Symbol name	Address	Value	Length	Relocation
1	SAMPLE	100	--	160	R
2	FIVE	136	5	4	R
3	FOUR	140	4	4	R
4	RESULT	144	__	4	R

Literal Table (LT)

Sr. No	Literal	Address	Length
1	3	120	4
2	5	152	4
3	7	156	4

Output:

Base Table (BT)

Register no	Availability	Value/ Contents
1	N	--
:	:	:
:	:	:
:	:	:
15	Y	100

Object Code

LC	OPCODE	OPERAND
100	5A	1,40(0,15)
104	1B	1,20(0,15)
108	50	1,44(0,15)
112	18	1,2
124	5A	2,36(0,15)
128	1B	2,52(0,15)
132	1B	2,56(0,15)

Instructions :

1.
2.
3.

Test Cases:

1. Check syntax of instruction (Correct and wrong)
2. Symbol not found
3. Wrong instruction
4. Duplicate symbol declaration
5. Test the output of program by changing value of START & USING pseudo opcode.

Software Requirement:

1. Fedora
2. Eclipse
3. JDK

Hardware Requirement:

Frequently Asked Questions:

1. What is two pass assembler?
2. What is the significance of symbol table?
3. Explain the assembler directives EQU, ORIGIN.
4. Explain the assembler directives START, END, LTORG.
5. What is the use of POOLTAB and LITTAB?
6. How literals are handled in pass I?
7. What are the tasks done in Pass I?
8. How error handling is done in pass I?
9. Which intermediate data structures are designed and implemented in PassI?
10. What is the format of a machine code generated in PassII?
11. What is forward reference? How it is resolved by assembler?
12. How error handling is done in pass II?
13. What is the difference between IS, DL and AD?

Conclusion:

The intermediate data structures generated in Pass-I of assembler are given as input to the Pass-II of assembler, processed by applying Pass-II algorithm of assembler and machine code is generated

Assignment No.: 03

Problem Statement: Design suitable data structures and implement Pass-I of a two pass macro processor using OOP features in Java/C++. The output of Pass-I (MNT, MDT, ALA & Intermediate code file without any macro definitions) should be input for Pass-II.

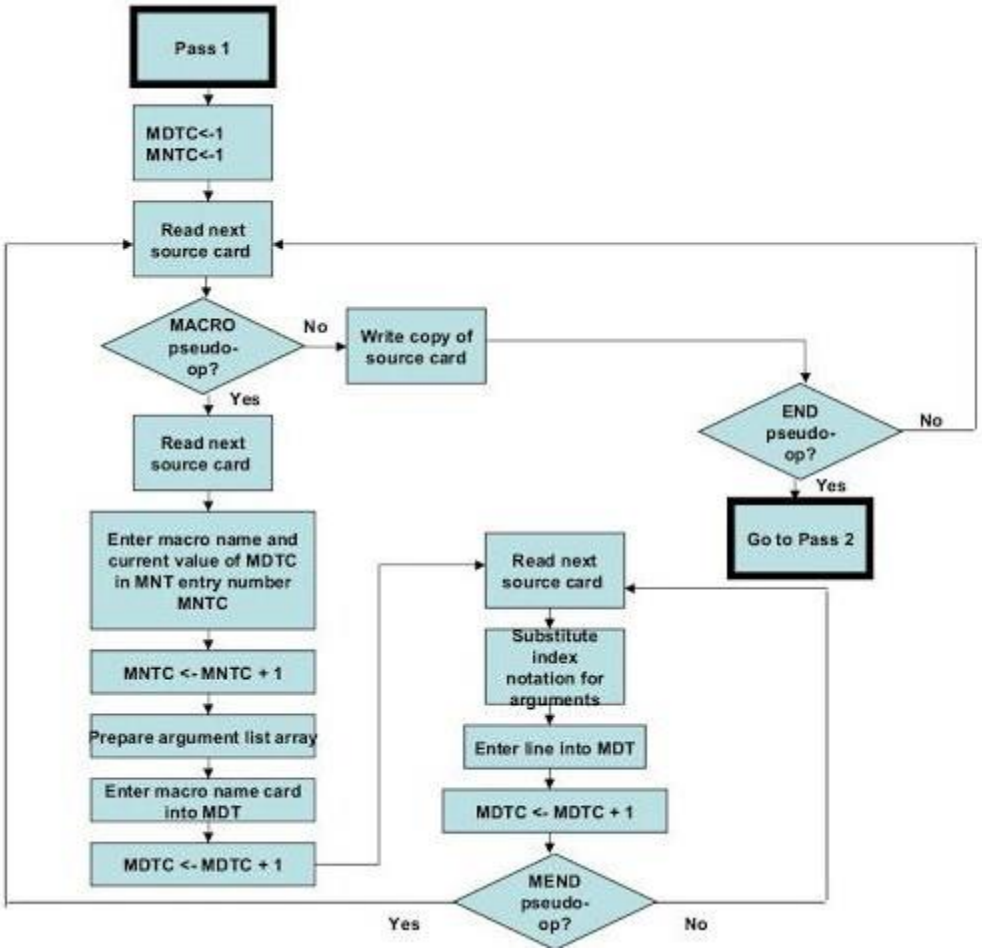
Objectives:

1. To identify and design different data structure used in macro-processor implementation
2. To apply knowledge in implementation of two pass microprocessor.

Theory:

1. What is macro processor?
2. Differentiate Macro and Function?
3. Explain the design of Pass- I of macro-processor with the help of flowchart?
4. Explain the design of Data structure used in Pass-I?
5. Explain the data structures used in Pass-I?

Algorithm/Flowchart:



Design diagrams (if any):

1. Class diagram
2. Sequence diagram
3.

Input:

Small assembly language program with macros written in file input.asm.

```
        MACRO
&lab    ADDS &arg1,&arg2
&lab    L 1, &arg1
        A 1, &arg2
        MEND
PROG START 0
        BALR 15,0
        USING *,15
LAB ADDS DATA1, DATA2
        ST 4,1
DATA1 DC F'3'
DATA2 DC F'4'
        END
```

Output:

Assembly language program without macro definition but with macro call.

Note: Follow the following templates during implementation

Macro Name Table (MNT) :

Index	Macro Name	MDT Index
1	ADDS	15

Macro Definition Table (MDT) :

Index	Macro Definition Card entry		
15	&lab	ADDS	&arg1, &arg2
16	#0	L	1, #1
17		A	1, #2
18		MEND	

Argument List Array (ALA) :

Index	Arguments
0	&lab
1	&agr1
2	&arg2

Instructions :

1.

2.

3.

Test Cases:

1. Check macro end not found.
2. Duplicate macro name found.
3. Check program output by changing macro name and parameter list.
4. Handle label in macro definition.
5. Handle multiple macro definitions and calls

Software Requirement:

1. Fedora
2. Eclipse
3. JDK

Hardware Requirement: N/A

Frequently Asked Questions:

1. Define macro?
2. Define purpose of pass-1 of two pass macro processor
3. List out types of macro arguments
4. What is the use of MDT-index field in MNT?
5. What we store in ALA?

Conclusion: We have successfully completed implementation of Pass-I of macro processor.

Assignment No.: 04

Problem Statement: Design suitable data structures and implement Pass-II of a two pass macro processor using OOP features in Java/C++. The output of Pass-I (MNT, MDT, ALA & Intermediate code file without any macro definitions) should be input for Pass-II.

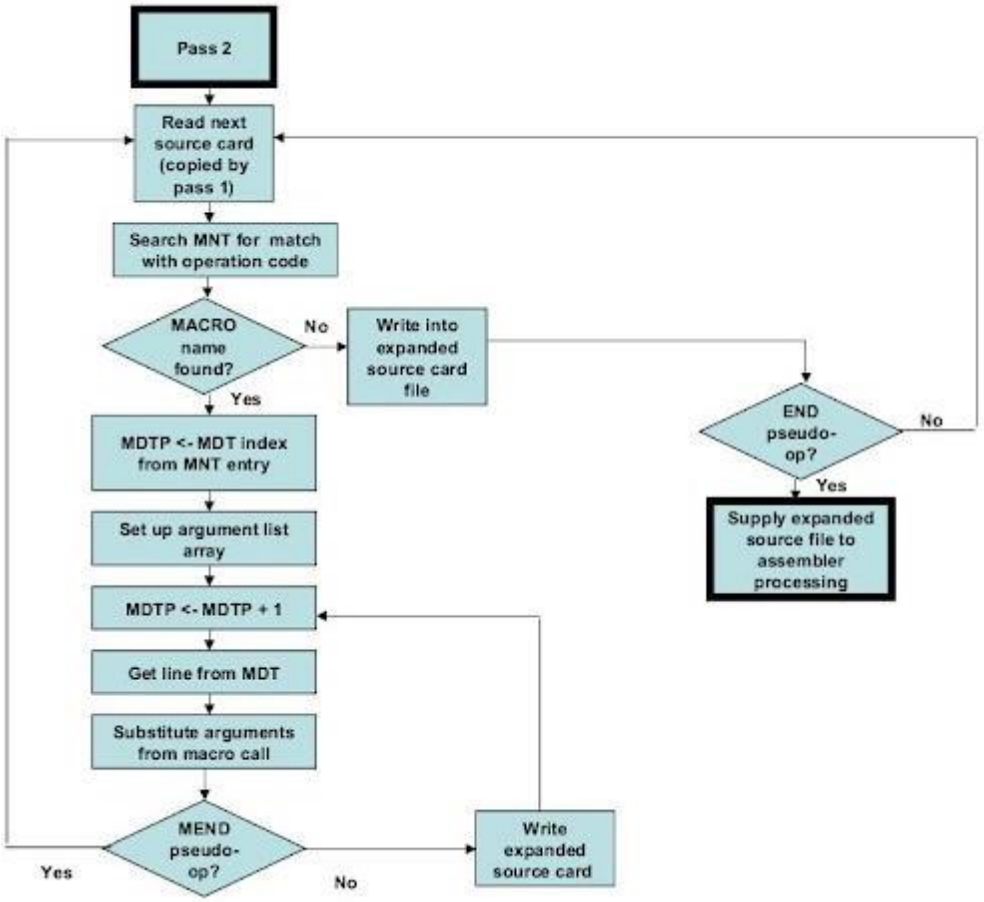
Objectives:

1. To identify and design different data structure used in macro-processor implementation
2. To apply knowledge in implementation of pass-2 of two pass microprocessor.

Theory:

1. Explain design steps of two pass microprocessor, types of statements, data structures required and flowcharts.

Algorithm/Flowchart:



Design diagrams (if any):

1. Class diagram
2. Sequence diagram
3.

Input: Output of pass-1 (Intermediate File) given as a input to pass-2.

```
PROG START 0
BALR 15,0
USING *,15
LAB ADDS DATA1, DATA2
ST 4,1
DATA1 DC F'3'
DATA2 DC F'4'
END
```

Output:

Assembly language program without macro definition and macro call.

```
PROG START 0
BALR 15,0
USING *,15
LAB L 1, DATA1
A 1, DATA2
ST 4,1
DATA1 DC F'3'
DATA2 DC F'4'
END
```

Macro Name Table (MNT):

Index	Macro Definition Card entry		
15	&lab	ADDS	&arg1, &arg2
16	#0	L	1, #1
17		A	1, #2
18		MEND	

Macro Definition Table (MDT):

Index	Macro Name	MDT Index
1	ADDS	15

Argument List Array (ALA) :

Index	Arguments
0	LAB
1	DATA2
2	DATA3

Instructions :

1.
2.
3.

Test Cases:

1. Check macro definition not found.
2. Check program output by changing parameter list in macro call.

Software Requirement:

1. Fedora
2. Eclipse
3. JDK

Hardware Requirement: N/A

Frequently Asked Questions:

1. What is macro expansion?
2. Define purpose of pass-2 of two pass macro processor
3. What is positional arguments?
4. What is the use of MDT-index field in MNT?
5. What is the use of MNT table while processing macro call?

Conclusion: We have successfully completed implementation of Pass-II of macro processor.

Assignment No.: 05

Problem Statement:

Write a program to simulate CPU Scheduling Algorithms: FCFS, SJF (Preemptive), Priority (Non-Preemptive) and Round Robin (Preemptive).

Objectives:

1. To study the process management and various scheduling policies viz. Preemptive and Non preemptive.
2. To study and analyze different scheduling algorithms.

Theory :

1. Define process. Explain need of process scheduling.
2. Explain different scheduling criteria and policies for scheduling processes.
3. Explain possible process states
4. Explain FCFS, SJF(Preemptive), Priority (Non-Preemptive) and Round Robin (Preemptive) and determine waiting time, turnaround time, throughput using each algorithm.

Algorithm/Flowchart:

1. FCFS

1. Input the processes along with their burst time (bt).
2. Find waiting time (wt) for all processes.
3. As first process that comes need not to wait so waiting time for process 1 will be 0 i.e. wt[0] = 0.
4. Find **waiting time** for all other processes i.e. for process i -> wt[i] = bt[i-1] + wt[i-1] .
5. Find **turnaround time** = waiting_time + burst_time for all processes.
6. Find **average waiting time** = total_waiting_time / no_of_processes.
7. Similarly, find **average turnaround time** = total_turn_around_time / no_of_processes.

1. SJF

1. Traverse until all process gets completely executed.

a) Find process with minimum remaining time at every single time lap.

b) Reduce its time by 1.

c) Check if its remaining time becomes 0

d) Increment the counter of process completion.

e) Completion time of current process = current_time +1;

e) Calculate waiting time for each completed process.

wt[i]= Completion time - arrival_time-burst_time

f) Increment time lap by one.

2. Find turnaround time (waiting_time+burst_time).

3. Priority

1- First input the processes with their burst time and priority.

2- Sort the processes, burst time and priority according to the priority.

3- Now simply apply FCFS algorithm.

4. RR

1- Create an array **rem_bt[]** to keep track of remaining burst time of processes. This array is initially a copy of bt[] (burst times array)
2- Create another array **wt[]** to store waiting times of processes. Initialize this array as 0.
3- Initialize time : t = 0
4- Keep traversing the all processes while all processes are not done. Do following for i'th process if it is not done yet.
a- If rem_bt[i] > quantum
(i) t = t + quantum
(ii) bt_rem[i] -= quantum;
c- Else // Last cycle for this process
(i) t = t + bt_rem[i];
(ii) wt[i] = t - bt[i]
(ii) bt_rem[i] = 0; // This process is over

Design diagrams (if any):

Class Diagram
Use Case Diagram
Sequence Diagram

Input:

1. Enter the number of processes
2. Enter burst time and arrival time of each process

Output:

1. Compute Waiting time, turnaround time, average waiting time, average turnaround time and throughput.

For each algorithm display result as follows:

Process	Burst Time	Arrival Time	Waiting Time	Turnaround Time
P1				
P2				
P3				
-				

Calculate

1. Average waiting time=
2. Average turnaround time=
3. Throughput=

Instructions :

1.
2.
3.

Test Cases:

1. Check arrival time of all process should not be same.

Software Requirement:

1. Fedora
2. Eclipse
3. JDK

Hardware Requirement: for simulation no dependency

Frequently Asked Questions:

1. What are the types of CPU scheduler?
2. What is the difference between long and short term scheduling?
3. Logic of program?
4. What is preemptive and non-preemptive scheduling?
5. What are types of scheduling algorithms?
6. Why Priority scheduling may cause low-priority processes to starve?
7. What are the goals of scheduling?
8. Define the difference between preemptive and non-preemptive scheduling.
9. Which scheduling algorithm is best? Why?

Conclusion:

CPU policies implemented successfully

Assignment No.: 06

Problem Statement:

Write a Java Program (using OOP features) to implement paging simulation using

1. FIFO
2. Least Recently Used (LRU)
3. Optimal algorithm

Objectives:

1. To study page replacement policies to understand memory management.
2. To understand efficient frame management using replacement policies.

Theory:

CONCEPT OF PAGE REPLACEMENT:

1. Page Fault: Absence of page when referenced in main memory during paging leads to a page fault.
2. Page Replacement: Replacement of already existing page from main memory by the required new page is called as page replacement. And the techniques used for it are called as page replacement algorithms.

NEED OF PAGE REPLACEMENT:

Page replacement is used primarily for the virtual memory management because in virtual memory paging system principal issue is replacement i.e. which page is to be removed so as to bring in the new page, thus the use of the page replacement algorithms. Demand paging is the technique used to increase system throughput. To implement demand paging page replacement is primary requirement. If a system has better page replacement technique it improves demand paging which in turn drastically yields system performance gains.

PAGE REPLACEMENT POLICIES:

1. Determine which page to be removed from main memory.
2. Find a free frame.
   1) If a frame is found use it
   2) if no free frame found, use page replacement algorithm to select a victim frame.
   3) Write the victim page to the disk.
3. Read the desired page into the new free frame, change the page and frame tables.
4. Restart the user process.

PAGE REPLACEMENT ALGORITHMS:

1. **FIFO**

   This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

2. **OPTIMAL PAGE REPLACEMENT ALGORITHM:** Replace the page that will not be used for longest period of time as compared to the other pages in main memory. An optimal page replacement algorithm has lowest page fault rate of all algorithm. It is called as OPT or MIN.

   ADVANTAGE:

   1) This algorithm guarantees the lowest possible page-fault rate for a fixed no. of frames.

   DISADVANTAGE:

   1) The optimal page replacement algorithm is very difficult to implement, as it requires the knowledge of reference strings i.e. strings of memory references.

3. **LEAST RECENTLY USED (LRU):** LRU algorithm uses the time of the page's last usage. It uses the recent past as an approximation of the near future, then we can replace the page that has not been used for the longest period of the time i.e. the page having larger idle time is replaced.

   ADVANTAGE:

   1) The LRU policy is often used for page replacement and is considered to be good.

   DISADVANTAGES:

   1) It is very difficult to implement.
   2) Requires substantial hardware assistance.
   3) The problematic determination of the order for the frames defined by the time of last usage

Algorithm/Flowchart:

1. FIFO :

1. Start the process

2. Read number of pages n

3. Read number of pages no

4. Read page numbers into an array a[i]

5. Initialize avail[i]=0 .to check page hit

6. Replace the page with circular queue, while re-placing check page availability in the frame

Place avail[i]=1 if page is placed in the frame Count page faults

7. Print the results.

8. Stop the process.

2. **LEAST RECENTLY USED**

1. Start the process

2. Declare the size

3. Get the number of pages to be inserted

4. Get the value

5. Declare counter and stack

6. Select the least recently used page by counter value

7. Stack them according the selection.

8. Display the values

9. Stop the process

3. **OPTIMAL**

ALGORTHIM:

1. Start Program

2. Read Number Of Pages And Frames

3.Read Each Page Value

4. Search For Page In The Frames

5. If Not Available Allocate Free Frame

6. If No Frames Is Free Replace The Page With The Page That Is Least Used

7.Print Page Number Of Page Faults

8.Stop process.

Design diagrams (if any):

1. Class Diagram

Input:

1. Number of frames
2. Number of pages

3. Page sequence

Output:

1. Sequence of allocation of pages in frames (for each algorithm)
2. Cache hit and cache miss ratio.

Instructions :

Test Cases:

1. Test the page hit and miss ratio for different size of page frames.
2. Test the page hit and miss ratio for both algorithms with different page sequences.

Software Requirement:

1. Fedora
2. Eclipse
3. JDK

Hardware Requirement:

Frequently Asked Questions:

1. What is virtual memory?
2. Explain working of LRU page replacement algorithm
3. Explain working of OPTIMAL page replacement algorithm
4. Which Page replacement algorithm is best?
5. Explain what is Belody's Anomaly?
6. Explain the scenario in which page replacement algorithm is used?
7. Explain what is page fault?
8. Explain what is paging scheme?
9. Explain what is counting based page replacement algorithms?

Conclusion: Successfully implemented all page replacement policies.

Assignment No.: 07

Problem Statement:

To write a program to implement Bankers algorithm for detection & Avoidance of deadlock.

Objectives:

1. To study bankers algorithm to avoid deadlock.
3. To study the safety algorithm for finding out whether or not a system is in safe state.

Theory:

Deadlock: A set of processes is in deadlock state when every process in the set is waiting for an Event that can only caused by another process in the set. Examples of such processes are resources acquisition and release. Deadlock prevention algorithm prevent deadlock situation by restraining how requests can be made. The restrains ensure that at least one of the necessary conditions for deadlock can't occur and hence deadlock can't hold. A side effect of this method is low device utilization and reduced system throughput. An alternative method requires additional information about resources available and resources currently allocated to each process, also the future requires and release of each process to decide if the current request can be satisfied or must wait to avoid a possible future deadlock. There are various algorithms for deadlock avoidance, which differ in the amount and type of information required.

The Banker's Algorithm:

This is a deadlock avoidance algorithm. The name was chosen since this algorithm can be used in a banking system to ensure that the bank never allocate its available cash in such a way that it can no longer satisfy further request for cash.

When a new process enters a system, it must declare the maximum no. Of instances of each resource type that it may need. This no. May not exceed the total number. Of resources in the system. When a user requests a set of resources, the system must determine whether the allocation of this resource will leave the system in a safe state. The resource are allocated otherwise, the process must wait until some another process releases enough resource.

Required Database:

Let n be the no. Of processes in the system and m be the no. Of resource type.

Available: A vector of length m indicates the no. Of available resources of each type. If

Available [j] =k, there are k instances of resource type Rj available.
Max: An n*m matrix defines the maximum demand of each process. If Max [i, j] =k, then Pi May request almost k instances of resource type Rj.
Allocation: An n*m matrix defines the no. Of resources of each type currently allocated to Each process. If Allocation [i,j]=k, then process Pi is currently allocated k instances of resource type Rj.
Need: An n*m matrix indicates the remaining resource need of each process. If need [i, j] =k, Then Pi may need k more instance of resource type Rj to compute its task. Need [i, j] =Max [i, j]-Allocation [i, j].

Safety algorithm:

The algorithm for finding out whether or not a system is in a safe state can be describes as follows:
Let Work and Finish be vectors of length m & n respectively. Initialize Work: =Available And Finish[i]:= false for i=1,2,...,n.
Find an i such that both
Finish[i] =false
Need[i] <=Work
If no such i exists, go to step 4.
Work: =Work
Allocation[i].
Finish[i]:=true go
to step 2.
If finish[i] =true for all i, then the system is in safe state.

Resource-Request algorithm:

Let request[i] be the request vector for process P[i], If request [i, j] =k, then process P[i] wants k Instance of resource type R[j].When a request for resource is made by process p[i], the following Actions are taken:
If Request[i] <=Need[i], go to step 2.Otherwise, raise an error condition since the process has Exceeded its maximum claim.
If Request[i] <=Available, go to step 3.otherwise, P[i] must wait, since the resource are not Available.
Have the system pretend to have allocated the request resource to process P[i] by modifying The state as follows:

Available: =Available-Request[i];

Allocation[i] :=Allocation[i]+Request[i];

Need[i] :=Need[i]-Request[i];

If the resulting resource-Allocation state is safe the transaction is completed and process P[i] is Allocated its resources. However, if new state is unsafe, then P[i] must wait for request[i] and the old resource-allocation state is resorted.

Conclusion:

The goal of this algorithm is to handle all the requests without entering into the unsafe state. So that this algorithm is not widely used in the real world is because to use it, the operating system must know the maximum amount of resources that every process is going to need at all time.

Subject Name: Human Computer Interface

Subject Code: 310245

Lab Assignments

Assignment No : 1

Title: Paper prototype-Design elements of GUI

Problem Statement: Design a paper prototype for a selected Graphical User Interface

Learning Objectives:

- ✓ To observe the user view point of elements on a GUI
- ✓ To study the elements of impact by noting the observations.
- ✓ To help in refining the drawbacks of the existing system

Learning Outcomes:

- ✓ Details of elements present on a specific GUI.
- ✓ Studying Importance of these elements and their impact on the user interface
- ✓ Use the analysis for developing a better GUI

Requirements:

- ✓ Notepad and pencil
- ✓ Any selected GUI (Any website like e-shopping , 2 pages)

Theory:

- Shneiderman's Eight Golden Rules of Interface Design.
- Interaction Style

The selected GUI must be studied carefully for all the following elements:

1. Font color's used in background, text
2. Images and their size with respect to the whole Interface
3. Options for selection of data (Menu, arrows, sub-menus, highlighted text)
4. Text boxes, User support options
5. Use of icons for representations, menu selections.

After noting the above details, a chart should be prepared in following manner:

Title of Paper prototyping:-

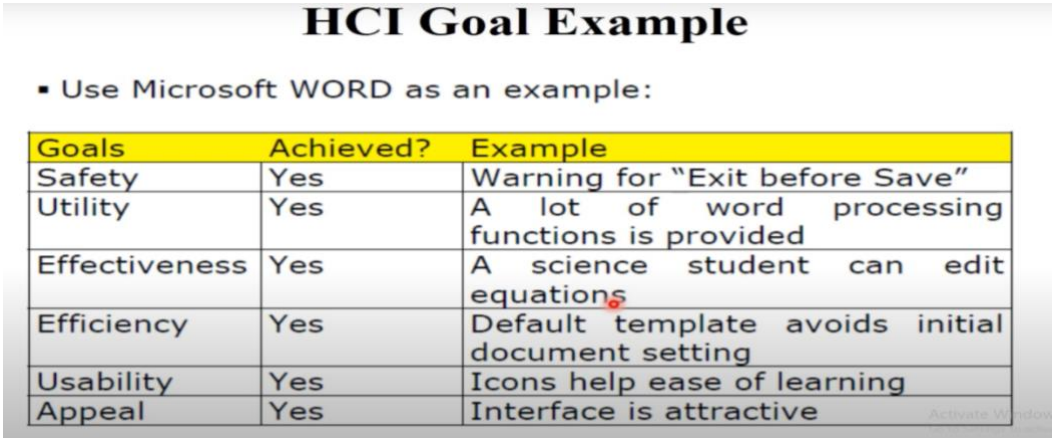
Sr. No	Element	Observation	Improvement/ corrections
1	--	--	--

Sr No.	Principles	Achieved	Justification

Interaction Styles used:

Input and output Channel:

Group of User:



Sample to refer

Documents to attach:
Paper prototyping Model(Black chart paper with your creativity)
Designing suing Figma, Canva, UI,UX, wireframe tool.

Conclusion: Thus students are able to identify the elements of GUI from user perspective to develop paper prototype.

Assignment No : 2
(Variation in lab Assignment)

Title: Comparison - GUI and Web design with a real time example.

Problem Statement: Analyze different websites to understand the principles of good design

Learning Objectives:

- ✓ Study the different interaction styles
- ✓ Categorize these interaction styles according to the requirement of the product
- ✓ Study the characteristics of web user interface.

Learning Outcomes:

- ✓ Study the various interaction styles and their advantages over other styles
- ✓ Study the interaction styles important in web interfaces.

Requirements:

Selected GUI of different websites, example. ebay, amazon, flipkart, zovi, myntra.

Theory:

Perform a comparative analysis on the selected GUI's to understand how each one caters to the goal, the interactions and flow of the payment system and prepare a report on the same.

Consider any 8 HCI principles and prepare the following table evaluating the websites.

Sr. No.	Principles	Poor	Average	Good	Very good	Excellent
1	Aesthetically pleasing					
2	..					

Conclusion:

Thus students were able to analyze the difference between GUI and web interface by considering the attributes.

Assignment No : 3

Title: GOMS model - Adding items to a cart of e-shopping website.

Problem Statement: Implement GOMS modeling technique to model user's behaviour in given scenario

Learning Objectives:

- ✓ Usability assessment of a given interface
- ✓ Model user behavior in terms of GOMS (Goals, Operators, Methods and Selection rules)
- ✓ Learn how to predict time it will take a user to carry out a goal using GOMS Model.

Learning Outcomes:

- ✓ Improve human-computer interaction efficiency by eliminating useless or unnecessary interactions.
- ✓ Using GOMS modeling for usability information when the system is in the earliest of design phases.
- ✓ Improve the performance of a cognitive skill, eliminate unnecessary operators from the method used to do the task.
- ✓ Provides hierarchical task description for a specific activity.

Requirements:
Specific scenario of user-interaction – Adding items in a cart of e-shopping website.

Theory:
Goals, operators, methods, and selection rules is a method derived from human-computer interaction (HCI) and constructs a description of human performance. The level of granularity will vary based on the needs of the analysis.

- ✓ The **Goal** is what the user wants to accomplish.
- ✓ The **Operator** is what the user does to accomplish the goal.
- ✓ The **Method** is a series of operators that are used to accomplish the goal.
- ✓ **Selection** rules are used if there are multiple methods, to determine how one was selected over the others.

Implementing GOMS for given Scenario:
Define the User's Top-Level Goal - Adding items in a cart
Goal: Select an item and Add it to cart

Operator: Amount of mouse clicks required

Methods:
1. Check list the item, click on quantity, click on "add to cart"
2. Click on quick view and add to cart

Selection:
Based on time taken to achieve the goal, select the appropriate method.

Conclusion: Thus Students were able to apply the concept of GOMS model for selected interface.

Assignment No : 4

Title: GUI in Python

Problem Statement: Design a user interface in Python

Learning Objectives:
- ✓ To design a user interface in Python
- ✓ To learn simplicity, user centric approach of a GUI in designing

Learning Outcomes:
A simple GUI designed using Tkinter library in Python.

Requirements:
Tkinter - standard GUI library for Python

Implementation Steps:

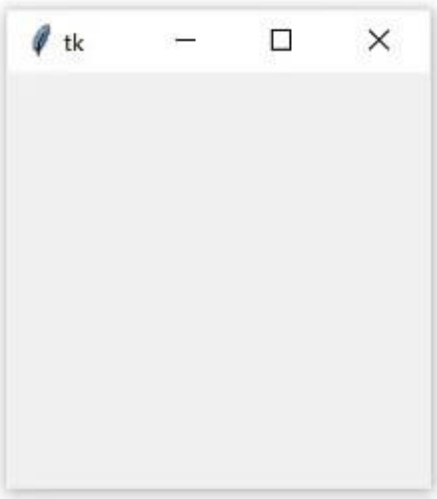
import the Python GUI Tkinter module:

```
>>> import tkinter as tk
```

A **window** is an instance of Tkinter's Tk class. Go ahead and create a new window and assign it to the variable window:

```
>>> window = tk.Tk()
```

When you execute the above code, a new window pops up on your screen. How it looks depends on your operating system:



(a) Windows

Adding a Widget

Use the tk.Label class to add some text to a window. Create a Label widget with the text "Hello, Tkinter" and assign it to a variable called greeting:

```
>>>
>>> greeting = tk.Label(text="Hello, Tkinter")
```

Working With Widgets

Each **widget** in Tkinter is defined by a class. Here are some of the widgets available:

Widget Class	Description
Label	A widget used to display text on the screen
Button	A button that can contain text and can perform an action when clicked
Entry	A text entry widget that allows only a single line of text
Text	A text entry widget that allows multiline text entry
Frame	A rectangular region used to group related widgets or provide padding between wid

Displaying Tex and Images With Label Widgets

Label widgets are used to **display text or images**. The text displayed by a Label widget can't be edited by the user. It's for display purposes only. As you saw in the example at the beginning of this tutorial, you can create a Label widget by instantiating the Label class and passing a string to the text parameter:

```
label = tk.Label(text="Hello, Tkinter")
```

abel widgets display text with the default system text color and the default system text background color. These are typically black and white, respectively, but you may see different colors if you have changed these settings in your operating system.

You can control Label text and background colors using
the foreground and background parameters:

```
label = tk.Label(
    text="Hello, Tkinter",
    foreground="white", # Set the text color to white
    background="black"  # Set the background color to black
)
```

ou can also control the width and height of a label with the width and height parameters:

```
label = tk.Label(
    text="Hello, Tkinter",
    fg="white",
    bg="black",
    width=10,
    height=10
```

Here's what this label looks like in a window:

Displaying Clickable Buttons With Button Widgets

```
button = tk.Button(
    text="Click me!",
    width=25,
    height=5,
    bg="blue",
    fg="yellow",
)
```

Getting User Input With Entry Widgets

he following code creates a widget with a blue background, some yellow text, and a width of 50 text units:

```
entry = tk.Entry(fg="yellow", bg="blue", width=50)
```

The best way to get an understanding of Entry widgets is to create one and interact with it. Open up a Python shell and follow along with the examples in this section. First, import tkinter and create a new window:

```
>>> import tkinter as tk
>>> window = tk.Tk()
```

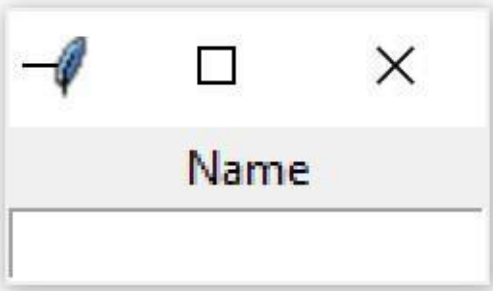
Now create a Label and an Entry widget:

```
>>> label = tk.Label(text="Name")
>>> entry = tk.Entry()
```

The Label describes what sort of text should go in the Entry widget. It doesn't enforce any sort of requirements on the Entry, but it tells the user what your program expects them to put there. You need to .pack() the widgets into the window so that they're visible:

```
>>> label.pack()
>>> entry.pack()
```

Here's what that looks like:



Conclusion: GUI created using Python

Assignment No : 5

Title: GUI for screen complexity

Problem Statement: To Redesign existing Graphical User Interface with screen complexity

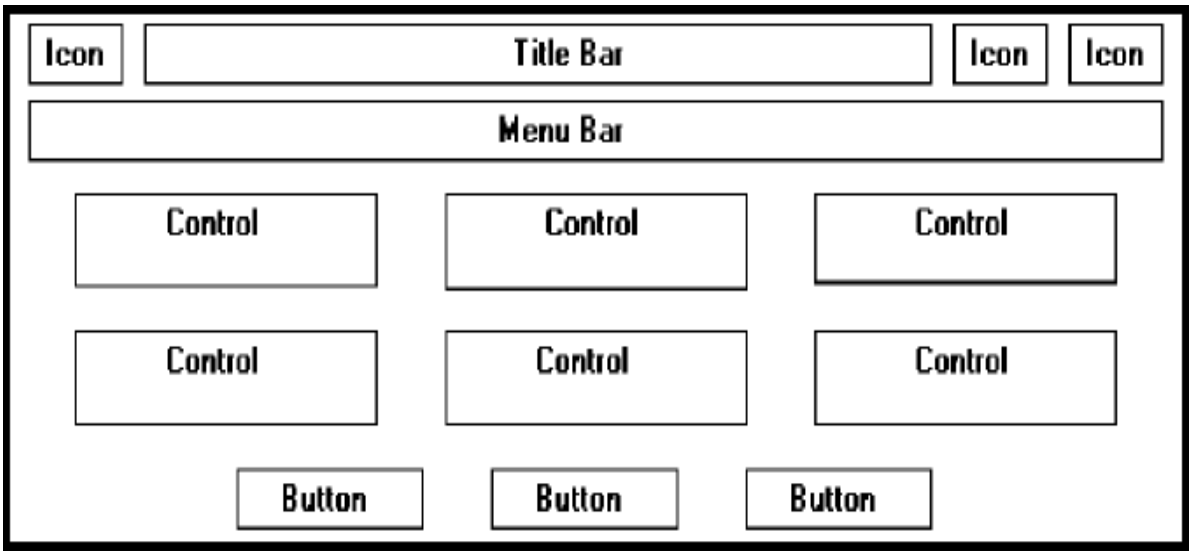
Learning Objectives:

- ✓ To study principles of Good screen design
- ✓ To apply the screen complexity rules to a GUI to improvise it.
- ✓ To analyse the human considerations in Interface and screen design.

Learning Outcomes:

- ✓ Design better screens in interfaces based on visually pleasing structure
- ✓ Learn to organize the elements on an interface screen by properly calculating the screen complexity.
- ✓ Learning the factors that affect the screen design quality with respect to user expectations

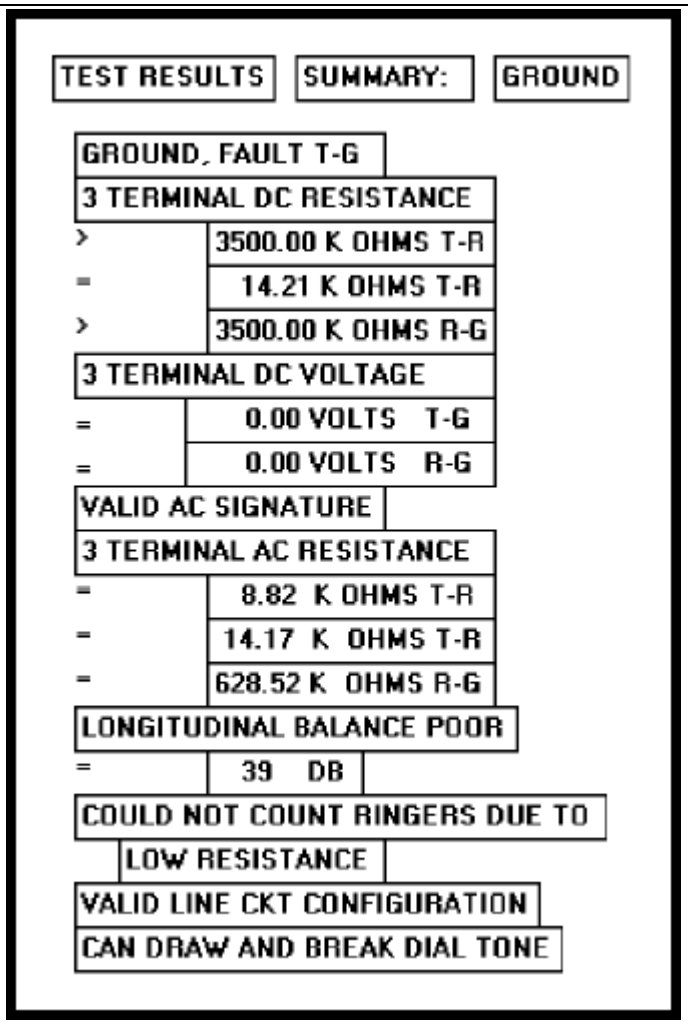
Requirements:
Any GUI screen from a selected application.
General structure of the elements on the screen to measure complexity factors.



To calculate the complexity first determine the following:

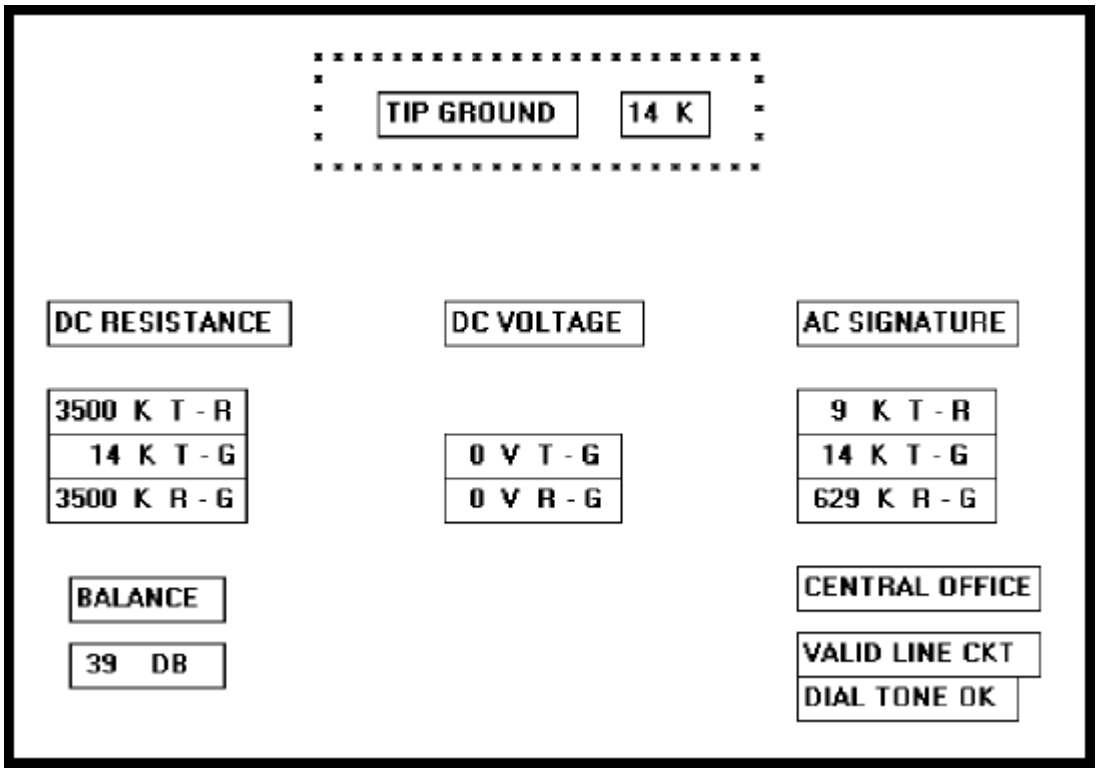
(1) the number of elements on the screen
(2) the number of horizontal (column) alignment points
(3) the number of vertical (row) alignment points

An example is given below:



Original Design of the GUI

- ✓ In the above screen the elements are not placed in a proper symmetry, which creates user confusion and loss of interest in the interface.
- ✓ The first requirement is to identify the text boxes and their places on the screen and then place them in a proper order , also group them as per requirement.
- ✓ The re-designed screen for the above example is shown in the figure below.
- ✓ To validate the improved screen, complexity of the screen is calculated which shows the optimization of screen space as well as the user friendly interface.



Re-designed Screen design

Calculation of complexity:
Original Design

1. 22 elements
2. 6 horizontal (column) alignment points
3. 20 vertical (row) alignment points
4. 48 = complexity (addition of all counts))

Redesigned Screen

1. 18 elements
2. 7 horizontal (column) alignment points
3. 8 vertical (row) alignment points
4. 33 = complexity

Students have to choose an application like the above GUI and calculate the screen complexity and re-design it by re-arranging the elements on the screen

Conclusion:
Brief description of the studied method for improving screen design complexity and the improvement in design by applying this method.