Name - Jayesh Devne
Roll No: 36-TE-A

Carnlin Page
Date / /

Experiment No: 1

Aim - To implement Pass-I Assembler

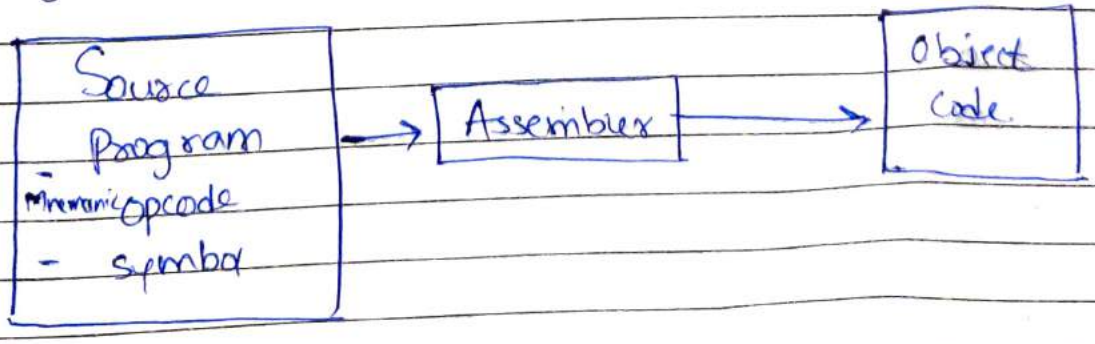
Problem statement - Design suitable data structure and implement Pass-I of a two-pass assembler for pseudo machines in Java using object oriented feature Implementation of few instructions from each category and few assembler directives.

Theory -

Assembly language - It is a low level language for a computer or other programmable device, each assembly language is specific to particular Computer architecture Assembly language uses a mnemonic to represent low-level machine.

Assembler - Assembly language is converted into executable machine by a utility program referred to an assembler the conversion process is referred to a assembly.

An assembler is a translator that translates an assembler program into a machine language program Then assembler proceeds to the next instruction. In this way, the entire machine code program is created.



Assembler directives -

If directives are pseudo instructions. They will not be translated into machine instructions. ~~They will not be~~ They only provide instructions / direction to the assembly.

Basic assembler directives are:

1. START - Specify name and starting address for program.
2. END - Indicate end of source program.
3. EQU - Replace a number by a symbol.

Main data structure -

1. operation code table (OPTAB)
2. location counter (LOCCTR)
3. Symbol table (SYMTAB)

One-pass-assembler -

A one pass assembler passes over the source code file exactly once, in the same pass collecting the labels resolving future references & doing the actual reference.

Data structure for assembler -

1. Opcode table.
2. looked up for the translation of mnemonic code.

Algorithm for pass-I assembler -

Begin

If starting address is given

LOCCTR = Starting address;
else
LOCCTR = 0;
while OPCODE != END do OR EOF
begin
read a line from a code
if there is a label
if these label is in SYMTAB error
else insert (label, LOCCTR) into SYMTAB
Search OPTAB for opcode
if found
LOCCTR + = M (M → instruction length)
else if this is an assembly direction
update LOCCTR as directed.
else error
write line to intermediate file.
end
program size = LOCCTR - Starting address
end.

Input -

```
START 200
MOVER AREG = '4'
MOVEM AREG, A
MOVER BREG = '1'
LOOP MOVER CREG, B
LTOR G
ADD CREG = '6'
STOP
A DS 1
B DS 1
END.
```

Expected o/p = Symbol table

A	208
LOOP	203
B	209

Intermediate key -

AD	01	C	200	
IS	04	1	L	1
IS	05	1	S	1
IS	04	2	L	2
IS	04	3	S	3
AD	05			
IS	01	3	2	3
IS	00			
DL	02	C	1	
DL	02	C	1	
AD	02			

Conclusion -

Thus, we have implemented PASS-I assembler using object oriented features.

Name- Jayesh Devre
Roll No: 36-TE-A

Experiment No: 2

Aim - To design data structure for pass-2 assembler

Problem statement - Implement Pass-II of two pass assembler for pseudo in Java using object oriented features. The output of assignment-I should be input for that assignment.

Theory -

Two pass assembler - The two pass assembler perform two passes over the source program. In the first pass, it reads the entire source program, all labels are collected & placed in symbol table. In the second pass, the instructions are again read & the assembled using symbol table.

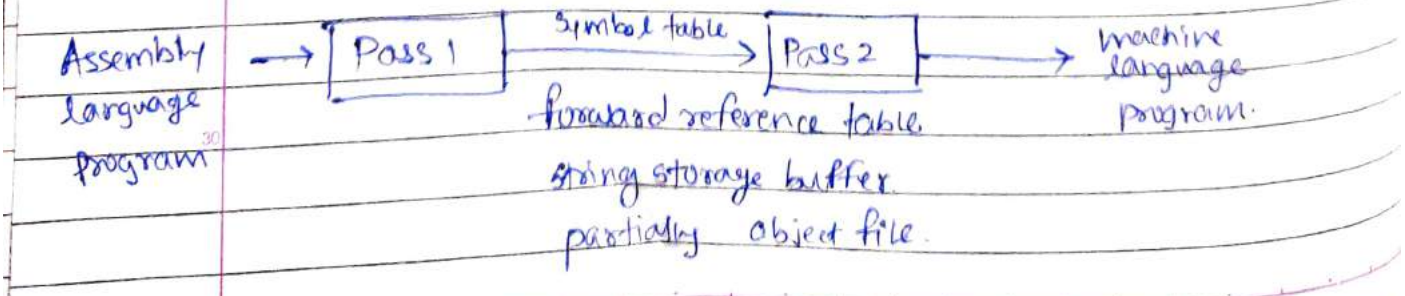
A two pass assembler perform two sequential scans over source code.

Pass 1: Symbols and literals are defined

Pass 2: object program is generated.

Data Structures -

1. Location Counter
2. Opcode translation table
3. Symbol table
4. String storage buffer
5. forward reference table



Solved example-

1	START	200		200] +04	1	211
2	MOVER	AREG='5'		201] +05	1	217
3	MOVEM	AREG, A		202] +04	1	218
4	LOOP MOVER	AREG, A		203] +05	3	218
5	MOVER	CREG, B		204] +01	3	212
6	ADD	CREG='1'		.		
7	...			210] +07	6	214
12	BC	ANY, NEXT		.		
13	LTORG			211] +00	0	005
		='5'		212] +00	0	001
		='1'				
14				214] +02	1	219
15	NEXT SUB	AREG='1'		215] +07	1	202
16	BC	LT, BACK		216] +00	0	000
17	LAST STOP					
18	ORIGIN	LOOP+2		204] +03	3	218
19	MULT	CREG, B				
20	ORIGIN	LAST+1		217]		
21	A DS	1				
22	Back EQU	LOOP		218]		
23	B DS	1				
24	END			219] +00	0	001
25						
25		='1'				
20	ORIGIN	LAST+1				

Conclusion-

Thus we have generated machine code for Solve program.

Name - Jayesh Devre
Roll No: 36 - TE A

Experiment No - 3

Aim - To design data structure for microprocessor.

Problem statement - Design suitable data structure and implement pass-I of a two-pass-macro-processor using OOP features in Java.

Theory -

1] Macro-processor -

It is a program that reads a files and scans them for certain keywords when a keyboard is found, it is replaced by some text.

2] Basic tasks performed by macro processor -

a] Recognize macro defination
b] Save defination
c] Recognize call
d] Expanded calls and substitute arguments.

3] macro defination part -

a] macro prototype statement
b] model statement
c] Preprocessor statement.

4] Data structure for macro defination processing -

a] Macro name table (MNT)
b] parameter Name table (PNTAB)
c] Keywords parameter default table (KPDTAB)
d] macro Defination table (MDT)

Algorithm -

```
begin {macro processor}
  EXPANDING = FALSE
    while OPCODE ≠ 'END'
    begin
      GETLINE
       PROCESSLINE
      end (while)
    end {macro processor}
  procedure PROCESSLINE
begin
    search NAMTAB for OPCODE
     if found then
       EXPAND
    else if OPCODE = 'MACRO' then
     DEFINE
     else write source line to expanded file
    end {PROCESSLINE}
```

Solved example -

Source

```
STRG   MACRO
       STA    DATA 1
       STB    DATA 2
       STX    DATA 3
       MEND
  .
STRG
STRG
```

Expanded Source

```
  .
  .
  .
       { STA    Data1
         STB    Data2
         STX    Data3

       { STA    Data1
         STB    Data2
         STX    Data3
```

Name - Jayesh Devre
Roll No: 36-TE-A

Experiment No. 4

Aim - Design of MACRO PASS-2

Problem statement - write a Java program for PASS-II of a two pass macro processors. The output of assignment is (MNT, MDT & file without any macro definations) should be input for this assignment.

Theory -

① MACRO PROCESSOR - It is a program that reads a files and scans them for certain keywords. when a keyword is found it is replaced by some text. The keyword / text combination is called MACRO

② Basic tasks performed by macro processor -
a) Recognize macrodefination
b) Save the defination
c) Recognize call
d) expanded calls & substitute arguments.

Pass 2 : macro calls and expansion
Pass2 algorithm examines the operation code of every input line to check whether it exist in MNT or not.

Conclusion -
Thus Pass-II of macroprocessor is implemented and ALA file is generated.

Name - Jayesh Devre
Roll No: 36 - TE - A

Experiment No: 5

Aim - Design lex program to generate token of given input file.

Problem statement - write a program using lex specifications to implement lexical analysis phase of compiler to generate tokens of subset of Java program.

Pre requisite - LEX110, LEX120, LEX130, LEX140, LEX150, LEX160

Theory - lexical analyzer is a tool for generating Scanner. scanners are programmers that recognize lexical patterns in text. A matched regular expression may have an associated action. lex turns the users expressions and actions into ~~that~~ host general purpose language the generated program is named yylex.

Regular expression in lex: A regular expression is a pattern description using a meta language. An expression is made up of symbols.

Programming lex - It can be divided in 3 steps-

1. Specify the pattern - associated actions in a form than lex can understood.
2. Run lex over this file to generate C code for the scanner.
3. Compile and link the code to procedure the execute scanner

lex program is divided in 3 sections-
1] Global C and lex declarations.

2] Patterns.
3] Supplement C functions.
The sections are delimited by %%.

A Character class define a single character ituo operators supported in a character class are hyphen ("-") and circumflex ("^")

.... definations...
%%
.... rules....
%%
... subroutines...

Input to lex is divided in 3 sections with %%. %% dividing the section %% %%. Input is copied to output one character at a time. The first %% is always required as there always must be sections. If we don't specify any ruler then the default action is to match everything and copy is to output Defaults for input & output are stdin & stdout.

Conclusion -

Thus we studied lexical and Implemented application for it to generate tokens.

Name - Jayesh Devre
Roll No: 36-TE-A

Experiment No-6

Aim - Design the program to count no. of words lines and character of given input file.

Problem statement - write a program using lex specifications to implement lexical analyser phase of compiler to count no. of words, lines and characters of given input file.

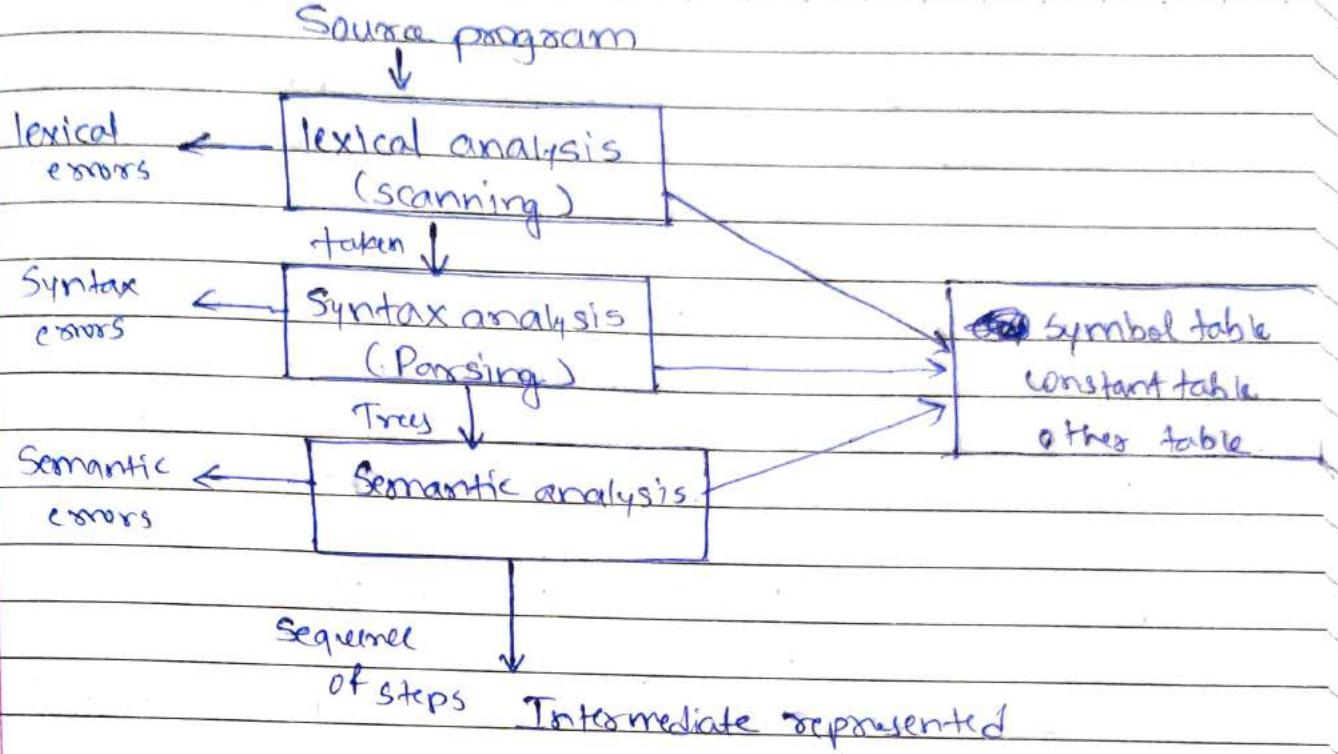
Pre-requisite - lex Basics.

Software requirements - ubuntu os with lextool (lex)

Theory - How the input is matched?

When the generator scanner runs it analyses it input looking for strings which match any of it patterns. If it finds more than one match it takes one matching the most text. If it finds two or more matches of same length the rule listed first in the flex input file is chosen.

Once the match is determined, the text corresponding to match is made available in yytext and its length is yyleng. If no. match is found then default rule is executed.



Conclusion-

Thus we have studied lexical analysis and implemented an application for lexical analyzer to count total number of words characters & lines etc.

Name - Jayesh Devre
Roll No: 36-TE-A

Experiment No. 7

Aim - Design and Lex & Yacc program to validate type and syntax of variable declaration in Java.

Problem statement - write a program using Yacc specifications to implement lexical analysis phase of Compiler to validate type and syntax of variable declaration in Java.

Pre requisits - LEX 110, LEX 120, LEX 130, LEX 140, LEX 160, 250.

Software requirements - ubuntu OS, Flex, YAcc (Lex & YAcc)

Theory - Yacc (Yet Another Compiler-Compiler) is a computer program for the UNIX operating system. developed by Stephen C. Johnson. It is a look ahead left to right prior generator., the part of a compiler that tries to make syntatic sense of source code. Based on analytic grammer written in a notation similar job is to analyze the structure of input stream & operate of the big picture.

Structure of YAcc file -

```
.... definations...
%. %.
.... rules....
%. %.
... code.
```

Definations As with lex, all code between %. and %. is copied to begining of resulting (file. Rules as with lex a number of combination of pattern

and action. Input to you is divided into three sections. The definition system consist of token declarations and c code bracketed by "%{ and %}" The BNF grammer is placed in rulers sections If can be used to express context free languages

Ex. E → E + E
E → E * E
E → id

Translating, Compiling & executing a Yacc program

lex program file consists of lex specification and should be named .l and the Yacc program consists of yacc specifications and should be named .y

Command to generate passes
lex <filename>.l
yacc -d <filename>.y
.cc lex y.c y.tab.c -ll
.% a.out

The execution of parser begins from the main function, which will be ultimately call yyparse() to run parser

Lexical analyzer for Yacc-
The user must supply a lexical analyzer to read input stream and communicate tokens. If there is a value associated with that token It should be assigned to yylval.
The portion of lexical analyzer might look like:

```
yylex (){
 extern int yylval;
   int c;
   c = getchar ();
   ... switch(c){
        case '0'
     Case '1':
  case '9':
     yylval = c - '0';
     return (DIGIT);
      ...}
```

Say name of token in 'DIGIT'

Comprising sentence types → Sentences give structure to that language. They come in four types- Simple, compound, complex & compound-complex.

- Simple sentence is as independent clause with one subject and one verb.
- The compound sentence is two or more independent clause, joined with comma, semicolon & conjunctions

Conclusion - Here we have studied lexical analyzer syntax and implemented lex & Yacc application for syntax analyzer to validate given infix expression.

Name - Jayesh Devre
Roll No: 36-TE-A

Experiment No. 8

Aim - To implement syntax analysis phase of compiler.

Problem statement -
write a program using Yacc specifications to implement syntax analysis phase of compiler to validate type and syntax of variable declaration in Java.

Theory -
YACC (Yet another Compiler compiler).
It is a standard parser generator for the UNIX as An open source program, Yacc generater for the parser in c programming language
The program is usually rendered in lowercase ~~The acronym is usually~~ re but it is ocassionally been as Yacc or YACC this original version of Yacc was written by Stephen Johnson at american telephone & telegraph. [AT&T].

Yacc file format -
```
%{
    C declaration
%}
    Yacc declarations
%%
```

Grammer rules -
```
%%
Additional C code (/* user subroutines */)
```

* Algorithm.

1. Include Header lines
2. ~~Return~~ Declare rules
3. Return Datatype
4. Return comma
5. Return SC
6. Return NL
7. Return ID
8. END

Conclusion -

Thus we implement syntax analysis phase of compiler to validate type & syntax of variable declaration in Java.

Name - Jayesh Derre
Roll No: 36-TE-A

Experiment No - 9

Aim - To implement Yacc specification with simple & compounded sentences.

Problem statement - write a program using Yacc specification to implement syntax analysis phase of compile to recognize simple & compound sentences.

Theory -

Syntax Analyzers -

Syntax analyzes of parsing is the second phase ie after lexical analysis. It checks the syntactical structure of given input. i.e whether the given i/p is correct syntax or not. It does so by building the data structure, called parse tree or syntax tree the input string is found to be in correct syntax.

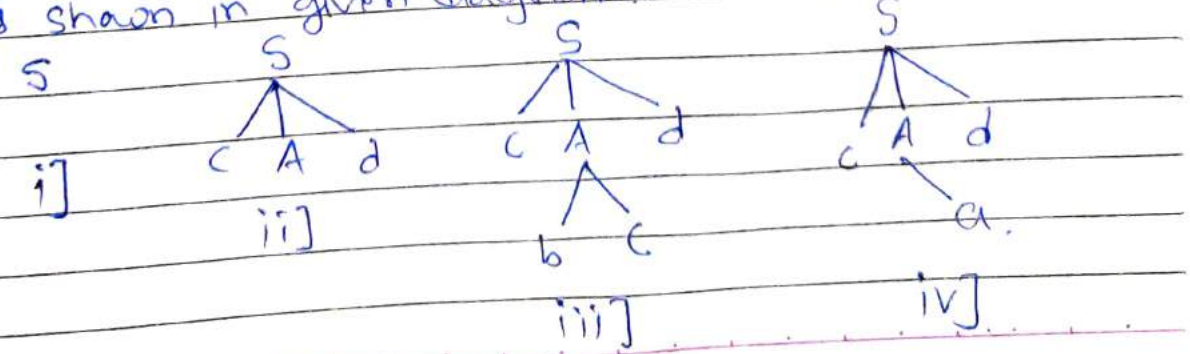
e.g. Suppose production rules for the grammar of 'c' language are

$S \rightarrow cAd$

$A \rightarrow b/a$.

And the input string is "cad"

Now the parser attempts to construct syntax tree from this grammer for the given input string. It used the given production rules & applies those as needed to generate the string. To generate string 'cad' it uses the rules as shown in given diagram.



Algorithm-

Begin
include header files
define compound = 0
define rules
define yy.conap()
take input
yy.lex();
Check if (compound == 1)
print ('sentense is compound');
else
print ('Sentense is simple');
Return
End;

Conclusion-

Thus we have implemented syntax analyze phases of compiler to recognize simple & Compound statement given input file

Name - Jayesh. Devre
Roll No: 36 - TE - A

Experiment No - 10

Aim - Implement Job scheduling algorithm
i] FCFS ii] SJF iii] Priority iv] RR.

Problem statement - write a Java program (using oop features) to implement following scheduling algorithm FCFS, SJF, Priority, RR.

Theory - 1. First come first serve -
The process that request the CPU first, is the one to which it is allocated first. The algorithm is implemented using a Job queue. when a process request the CPU it is added at the tail of Job queue. The CPU is allocated to the process at head of queue.

Algorithm -
① Input the process along with their burst time
② Find waiting time for all process.
③ As first process that comes first 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 process i.e for process 1 → wt[1] = bt[i-1] + wt[i-1]
⑤ Find fundamental time = waiting time + burst time
⑥ find average waiting time = total waiting time / no of processes
⑦ Similarly find average time = total_turn_around_time / no. of processes.

2. Shortest Job first -

This algorithm associates with it the length of next CPU burst. when the CPU is available it is assigned to that job with the smallest CPU burst. This algorithm provides minimum average waiting time. The major problem with this knows the CPU burst of a job.

Algorithm - 1. Sort all the processes in increased order according to burst time.
2. Then simply, apply FCFS.

3. Priority Scheduling -

① It is non preemptive algorithm.
② It is most common scheduling algorithm in batch systems.
③ Process with same priority are executed on FCFS basics.
④ Priority can be decided based on memory requirement time requirements or any other resource requirement

Algorithm -
① First input the process with their bt & priority
② Start the processes burst time & process priority & according to the priority
③ Now simply apply FCFS algorithm.

4. Round Robin Scheduling -

① RR is a CPU scheduling algorithm where each process is assigned a 1st time slot in cyclic way.

Camlin Page
Date / /

② It is primptive as process are assigned CPU only for a fixed slice of time at most.
③ Each process is provided a fixed time to execute is called a quantum.
④ once a process is executed for given time period it is preempted and other process executes for a given period of time.

Conclusion -

Thus we have implemented Job scheduling algorithm FCFS, RR, SJF & priority.

Name - Jayesh
Roll No: 36-TE-A

Experiment No. 11

Aim - Banker's algorithm for deadlock detection and avoidance.

Problem statement - Write a program to implement Banker's algorithm.

Theory - The banker's algorithm is a resource allocation & deadlock avoidance algorithm that tests for safety by simulating allocation for predetermined maximum possible amounts of all resources. then makes an 'S-State', check the test for possible activities, before deciding whether allocation should be allowed to continue.

Following data structures are used to implement the banker's algorithm.
let 'n' be the number of processes in the system & 'm' be number of resources types.

Available:
It is 1-d array of size 'm' indicating the number of available resources of each type
Available [j] = K means there are 'K' instances of resources type Rj.

Max - It is a 2-d array of size 'nxm' that defines the maximum demand of each process in a system. Max [i, j] = K means process Pi may request to at most 'K' instances of resource type 'Rj'

Allocation -
It is a 2-d array of size 'n x m' that defines the number of resources of each type currently allocated to each types.
Allocation [i,j] = k means process Pi is currently allocated 'k' instances of resource type Rj

Need - It is 2-d array of size 'n x m' that indicates the remaining source need of each process.
Need [i,j] = max [i,j] - Allocation [i,j]

Safety algorithm -

1. let work and finish be vectors of length 'm' & 'n' resp. Initialize work = Available.
   finish [i] = false : for i = 1, 2, 3 . . . n
2. finish [i]s i such that both
   a] finish [i] = false
   b] Need i <= work.
   if NO such i exists goto step(4)
3. work = work + Allocation [i]
   finish [i] = true
   goto step (2)
4. If finish [i] = true for all i
   then system is in safe state.

Conclusion -
Thus we have studied implementation of banker's algorithm for deadlock attention & avoidance.

Name- Jayesh
Roll No: 36-TE-A

Experiment No. 12

Aim- Implement UNIX system calls like for process management.

Problem statement- To write a program to implement UNIX system calls like for process management.

Theory-

SYSTEM CALL- ① When a programin user mode requires access to RAM or a Hardware resource, it must be ask the kernel to provide access to that resource. This is done via something called a system call.
② When a program makes a system call, the mode is switched from user mode to kernel mode this is called Context Switch

KERNEL MODE-

① When CPU is in kernel mode, the code being executed can access any memory address, and any hardware resource.
② That means system call will be in safe state even if a program in user mode crashes.
③ Hence, most program in an OS runs in user mode

Unix system calls-

PS- command- This command is used to provide information about the the currently running processes including their process identification number (PID).

Camlin Page
Date / /

Fork command - This command is used to provide information about the currently running process including their one is child process & other is parent.

Join command - It is command line utility for joining lines of two files in command field.

Exec() command - It is also used to create process exec() call replaces the address space text segment, data segment etc.

Conclusion -

Thus the process system calls program is implement and studied various system calls

Name - Jayesh Devre
Roll No: 36-TE-A

Experiment No. 13

Aim - Study assignment on process scheduling algorithm in Android & Tizen.

Problem statement - Study Assignment on scheduling algorithm in android & Tizen.

Software Requirement - Android, SDK

Theory -

① Android OS -
Android is mobile os developed by google based on a modified of linux kernel & other open source software version designed primarily for touch screen mobiles devices such as smartphones & tablets.

② Tizen OS - It is mobile os developed by Samsung that runs on a wide range of Samsung os devices, including smartphones, tablets.

Process scheduling algorithms on Android and Tizen OS

① Normal scheduling -
Android is based on linux & uses the linux kernel's scheduling mechanisms for determining scheduling policies. the linux's time sliced

Scheduling policy combines static & dynamic priorities-

- Real time scheduling-

The standard linux kernel provides two-real time scheduling policies, Sched-FIFO & Schen-RR the main real-time policy is sched-FiFo. It implements FIFO algorithm non real time that use the Sched-Normal scheduling policy.

Thread scheduling-

A thread schedular decides which threads in the Android system should run, when and for how long Android thread schedular uses two main factors to determine the scheduling.

Priority based Pre-Emptive task scheduling for Android operating system-

Thus scheduling is particularly used for mobile os as the CPU utilization is medium turnaround & response time is high

Conclusion- Thus we have studied the concept of process scheduling of Android & lizen operating system.

Name - Jayesh Devre
Roll No: 36 - TE - A

Experiment No. 14

Aim - Implementing page replacement algorithm 1] LRU 2] optimal.

Problem statement -
To write a java program (using oop) to implement LRU & optimal algo. for page replacement.

Theory - ① LRU Page replacement -
The main difference between FIFO and optimal page replacement is that FIFO algorithm uses the time when page was brought into memory & the optimal algorithm uses the time when a page is to be used. If we use the recent past as an approximation of future then we will replace the page that has not been used for largest period of time. This approach is called as least recently used algo. Now consider reference string 7, 0, 1, 2, 0, 3, 4, 2, 3, 0, 3 with three memory frames or blocks.

Algorithm -
I. Start traversing the pages -
II. If set holds less pages than capacity
a] Insert page into the set one by one until the size of set reaches capacity or all page request are processed.
b] Simultaneously maintain the recent occured index.
c] Increament page fault.
III]
else

If current page is present in set, do nothing else
a) Find the page in set that was least recently used. We find it using index array. We basically need to replace the place with minimum index.
b) Replace the found page with current page.
c) Increament page faults
d) update index of current page.
2. Return page faults.

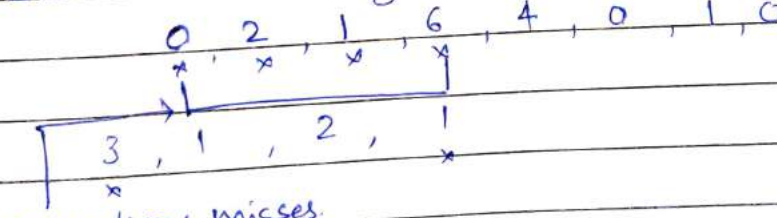
- ~~after called B~~

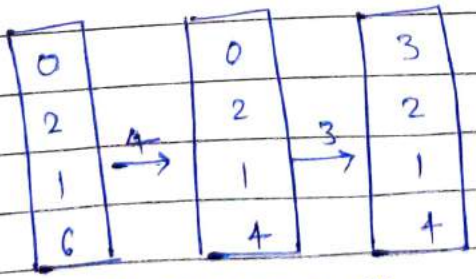
3. Optimal Replacement -

The algorithm has lowest ~~type~~ page fault rate of all algorithm. This algorithm state that Replace the page which will not be used for longest period of time.

- often called Balady's min Basic idea. Replace the page that will not be offered for the longest time.
- Impossible to implement.

- Consider the following reference string.





Fault rate = 6/12 = 0.50

Algorithm-

1. Start the process
2. Declare the size
3. Get number of pages to be inserted
4. get the value.
5. Compare counter label & stack.
6. Select optimal page by counter value.
7. Stack them according the selection.
8. Print pages with fault pages.
9. Stop the processes.

Conclusion -- Thus, we have implemented page replacement algorithm LRU & optimal

Name: Jayesh Bharat Devre **Roll**: TE-A-36

Assignment No. 01 [Pass 1 Assembler]

Problem Statement: 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

---

1. Pass 1 Program:

```
package com.company;

import java.io.BufferedReader;
import java.io.*;
import java.io.IOException;
import java.util.*;

public class Pass1 {
    public static void main(String[] args) {

        BufferedReader br = null;
        FileReader fr = null;

        FileWriter fw = null;
        BufferedWriter bw = null;

        try {
            String inputfilename = "C:\\Users\\Jayesh\\Desktop\\spos\\Input.txt";
            fr = new FileReader(inputfilename);
            br = new BufferedReader(fr);

            String OUTPUTFILENAME = "C:\\Users\\Jayesh\\Desktop\\spos\\IC.txt";
            fw = new FileWriter(OUTPUTFILENAME);
            bw = new BufferedWriter(fw);

            Hashtable<String, String> is = new Hashtable<String, String>();
            is.put("STOP", "00");
            is.put("ADD", "01");
            is.put("SUB", "02");
            is.put("MULT", "03");
            is.put("MOVER", "04");
            is.put("MOVEM", "05");
            is.put("COMP", "06");
            is.put("BC", "07");
            is.put("DIV", "08");
            is.put("READ", "09");
            is.put("PRINT", "10");

            Hashtable<String, String> dl = new Hashtable<String, String>();
            dl.put("DC", "01");
            dl.put("DS", "02");

            Hashtable<String, String> ad = new Hashtable<String, String>();

            ad.put("START", "01");
            ad.put("END", "02");
            ad.put("ORIGIN", "03");
            ad.put("EQU", "04");
            ad.put("LTORG", "05");

            Hashtable<String, String> symtab = new Hashtable<String, String>();
            Hashtable<String, String> littab = new Hashtable<String, String>();
            ArrayList<Integer> pooltab = new ArrayList<Integer>();

            String sCurrentLine;
            int locptr = 0;
            int litptr = 1;
            int symptr = 1;
```

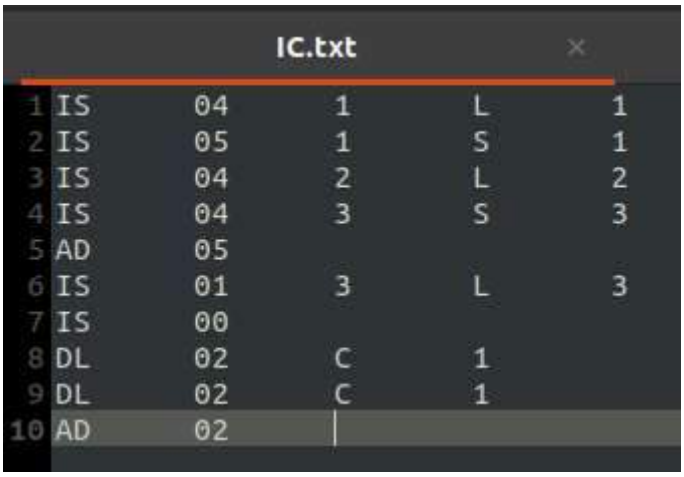
```
            int pooltabptr = 1;

            sCurrentLine = br.readLine();

            String s1 = sCurrentLine.split(" ")[1];
            if (s1.equals("START")) {
                bw.write("AD \t 01 \t");
                String s2 = sCurrentLine.split(" ")[2];
                bw.write("C \t" + s2 + "\n");
                locptr = Integer.parseInt(s2);
            }

            while ((sCurrentLine = br.readLine()) != null) {
                int mind_the_LC = 0;
                String type = null;

                int flag2 = 0; // checks whether addr is assigned to current symbol

                String s = sCurrentLine.split(" |\\,")[0]; // consider the first word in
the line

                for (Map.Entry m : symtab.entrySet()) { // allocating addr to arrived
symbols
                    if (s.equals(m.getKey())) {
                        m.setValue(locptr);
                        flag2 = 1;
                    }
                }
                if (s.length() != 0 && flag2 == 0) { // if current string is not " " or
addr is not assigned,
                    // then the current string must be a new symbol.
                    symtab.put(s, String.valueOf(locptr));
                    symptr++;
                }

                int isOpcode = 0; // checks whether current word is an opcode or not

                s = sCurrentLine.split(" |\\,")[1]; // consider the second word in the line

                for (Map.Entry m : is.entrySet()) {
                    if (s.equals(m.getKey())) {
                        bw.write("IS\t" + m.getValue() + "\t"); // if match found in
imperative stmt
                        type = "is";
                        isOpcode = 1;
                    }
                }

                for (Map.Entry m : ad.entrySet()) {
                    if (s.equals(m.getKey())) {
                        bw.write("AD\t" + m.getValue() + "\t"); // if match found in
Assembler Directive
                        type = "ad";
                        isOpcode = 1;
                    }
                }
                for (Map.Entry m : dl.entrySet()) {
                    if (s.equals(m.getKey())) {
                        bw.write("DL\t" + m.getValue() + "\t"); // if match found in
declarative stmt
                        type = "dl";
                        isOpcode = 1;
                    }
                }

                if (s.equals("LTORG")) {
                    pooltab.add(pooltabptr);
                    for (Map.Entry m : littab.entrySet()) {
                        if (m.getValue() == "") { // if addr is not assigned to the literal
                            m.setValue(locptr);
                            locptr++;
                            pooltabptr++;
                            mind_the_LC = 1;
                            isOpcode = 1;
                        }
                    }
```

```
                }

                if (s.equals("END")) {
                    pooltab.add(pooltabptr);
                    for (Map.Entry m : littab.entrySet()) {
                        if (m.getValue() == "") {
                            m.setValue(locptr);
                            locptr++;
                            mind_the_LC = 1;
                        }
                    }
                }

                if (s.equals("EQU")) {
                    symtab.put("equ", String.valueOf(locptr));
                }

                if (sCurrentLine.split(" |\\,").length > 2) { // if there are 3 words
                    s = sCurrentLine.split(" |\\,")[2]; // consider the 3rd word

                    // this is our first operand.
                    // it must be either a Register/Declaration/Symbol
                    if (s.equals("AREG")) {
                        bw.write("1\t");
                        isOpcode = 1;
                    } else if (s.equals("BREG")) {
                        bw.write("2\t");
                        isOpcode = 1;
                    } else if (s.equals("CREG")) {
                        bw.write("3\t");
                        isOpcode = 1;
                    } else if (s.equals("DREG")) {
                        bw.write("4\t");
                        isOpcode = 1;
                    } else if (type == "dl") {
                        bw.write("C\t" + s + "\t");
                    } else {
                        symtab.put(s, ""); // forward referenced symbol
                    }
                }

                if (sCurrentLine.split(" |\\,").length > 3) { // if there are 4 words

                    s = sCurrentLine.split(" |\\,")[3]; // consider 4th word.
                    // this is our 2nd operand
                    // it is either a literal, or a symbol
                    if (s.contains("=")) {
                        littab.put(s, "");
                        bw.write("L\t" + litptr + "\t");
                        isOpcode = 1;
                        litptr++;
                    } else {
                        symtab.put(s, ""); // Doubt : what if the current symbol is already
present in SYMTAB?
                        // Overwrite?
                        bw.write("S\t" + symptr + "\t");
                        symptr++;

                    }
                }

                bw.write("\n"); // done with a line.

                if (mind_the_LC == 0)
                    locptr++;
            }

            String f1 = "C:\\Users\\lenovo\\eclipse-workspace\\GroupA\\src\\SYMTAB.txt";
            FileWriter fw1 = new FileWriter(f1);
            BufferedWriter bw1 = new BufferedWriter(fw1);
            for (Map.Entry m : symtab.entrySet()) {
                bw1.write(m.getKey() + "\t" + m.getValue() + "\n");
                System.out.println(m.getKey() + " " + m.getValue());
            }

            String f2 = "C:\\Users\\lenovo\\eclipse-workspace\\GroupA\\src\\LITTAB.txt";
```

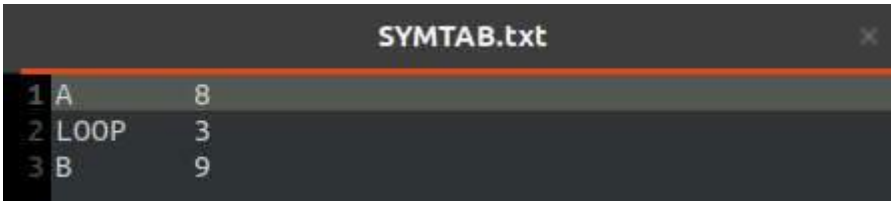
```
            FileWriter fw2 = new FileWriter(f2);
            BufferedWriter bw2 = new BufferedWriter(fw2);
            for (Map.Entry m : littab.entrySet()) {
                bw2.write(m.getKey() + "\t" + m.getValue() + "\n");
                System.out.println(m.getKey() + " " + m.getValue());
            }

            String f3 = "C:\\Users\\lenovo\\eclipse-workspace\\GroupA\\src\\POOLTAB.txt";
            FileWriter fw3 = new FileWriter(f3);
            BufferedWriter bw3 = new BufferedWriter(fw3);
            for (Integer item : pooltab) {
                bw3.write(item + "\n");
                System.out.println(item);
            }

            bw.close();
            bw1.close();
            bw2.close();
            bw3.close();

        } catch (IOException e) {
            e.printStackTrace();
        }

    }

}
```

PASS 1 - ASSEMBLER OUTPUT:

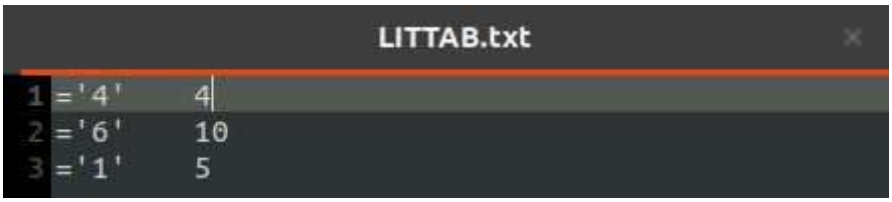
IC.txt



SYMTAB.txt



LITTAB.txt



Name: Jayesh Bharat Devre **Roll**: TE-A-36

Assignment No. 02 [Pass 2 Assembler]

Problem Statement: 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.

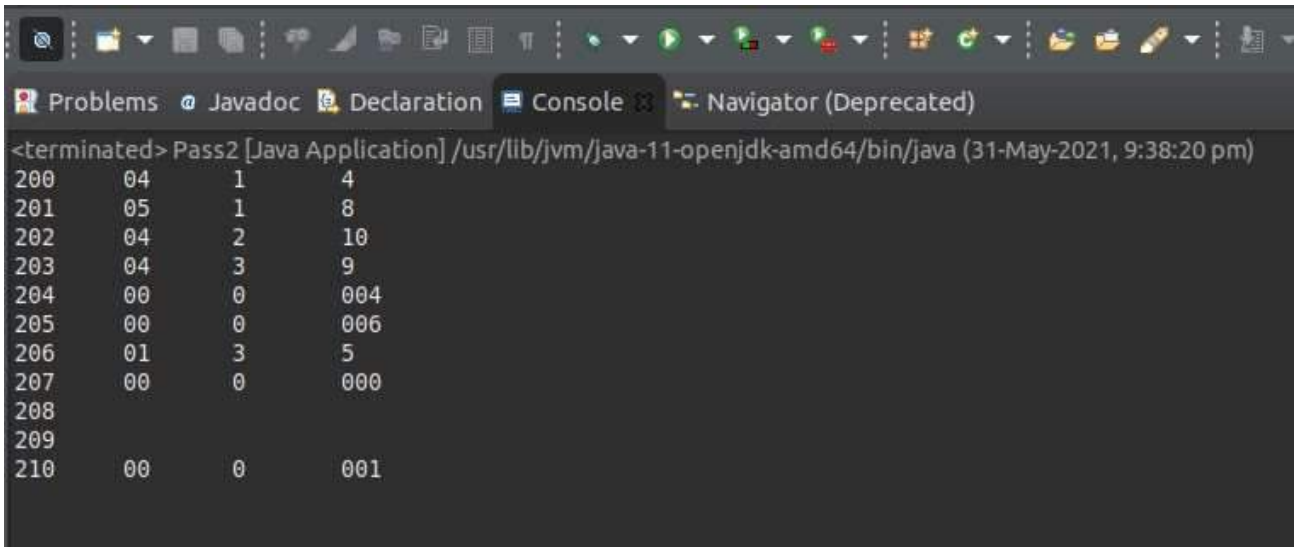
```
package com.company;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.lang.reflect.Array;
import java.util.ArrayList;
import java.util.Hashtable;
import java.util.Map;

public class Pass2{

    public static void main(String[] args) {

        try {

            //1. Read Intermediate code file
            String f ="C:\\Users\\Jayesh\\Desktop\\spos\\IC_new.txt";
            FileReader fw =new FileReader(f);
            BufferedReader IC_file=new BufferedReader(fw);

            //2.Read Symbol table file
            String f1="C:\\Users\\Jayesh\\Desktop\\spos\\SYMTAB.txt";
            FileReader fs=new FileReader(f1);
            BufferedReader symtab_file=new BufferedReader(fs);
            symtab_file.mark(500);

            //3.Read Literal table file
            String f2=" C:\\Users\\Jayesh\\Desktop\\spos\\LITTAB.txt";
            FileReader fl=new FileReader(f2);
            BufferedReader littab_file=new BufferedReader(fl);
            littab_file.mark(500);


            //4.create  littab array and hashtable for symbol table


            String littab[][]=new String[10][2] ;

            Hashtable<String, String> symtab = new Hashtable<String, String>();
            String str;
            int z=0;
            //5.Read LITTAB.txt
            while ((str = littab_file.readLine()) != null) {

                littab[z][0]=str.split("\\s+")[0]; //first word
                littab[z][1]=str.split("\\s+")[1]; //second word
                z++;
            }
            //6.Read SYMTAB.txt

            while ((str = symtab_file.readLine()) != null) {
                symtab.put(str.split("\\s+")[0], str.split("\\s+")[1]);
            }
            //7.Read POOLTAB.txt
            String f3 = "C:\\Users\\lenovo\\eclipse-workspace\\GroupA\\src\\POOLTAB.txt";
            FileReader fw3 = new FileReader(f3);
            BufferedReader pooltab_file = new BufferedReader(fw3);
```

```
ArrayList<Integer> pooltab = new ArrayList<Integer>();
String t;
while ((t = pooltab_file.readLine()) != null) {
    pooltab.add(Integer.parseInt(t));
}


int pooltabptr = 1;
int temp1 = pooltab.get(0);          //dry run
int temp2 = pooltab.get(1);

//7.Read IC.txt
String sCurrentLine;
sCurrentLine = IC_file.readLine();
int locptr=0;
//locptr=Integer.parseInt(sCurrentLine.split("\\s+")[3]);
locptr=Integer.parseInt(sCurrentLine.split("\t")[3]);

while ((sCurrentLine = IC_file.readLine()) != null) {


    System.out.print(locptr+"\t");


    String s0 = sCurrentLine.split("\t")[0]; //contains statement type

    String s1 = sCurrentLine.split("\t")[1]; //contains statement code

    if (s0.equals("IS")) {

        System.out.print(s1+"\t");
        if (sCurrentLine.split("\t").length == 5) {


            System.out.print(sCurrentLine.split("\t")[2] + "\t");
            //7.2 if third character is L
            if (sCurrentLine.split("\t")[3].equals("L")) {
                int add = Integer.parseInt(sCurrentLine.split("\t")[4]);

                //machine_code_file.write(littab[add-1][1]);
                System.out.print(littab[add-1][1]);

            }

            //7.3 or if third character is S
            if (sCurrentLine.split("\t")[3].equals("S")) {
                int add1 = Integer.parseInt(sCurrentLine.split("\t")[4]);

                //search for the 4th word in symbol table
                int i = 1;
                String l1;
                for (Map.Entry m : symtab.entrySet()) {
                    if (i == add1) {

                        System.out.print((String) m.getValue());
                    }
                    i++;
                }

            }
        } else {

            System.out.print("0\t000");
        }
    }



    //DRY RUN is a must

    if (s0.equals("AD")) {
        littab_file.reset();
        if (s1.equals("05")) {        //if it is LTORG
            int j = 1;
            while (j < temp1) {
```

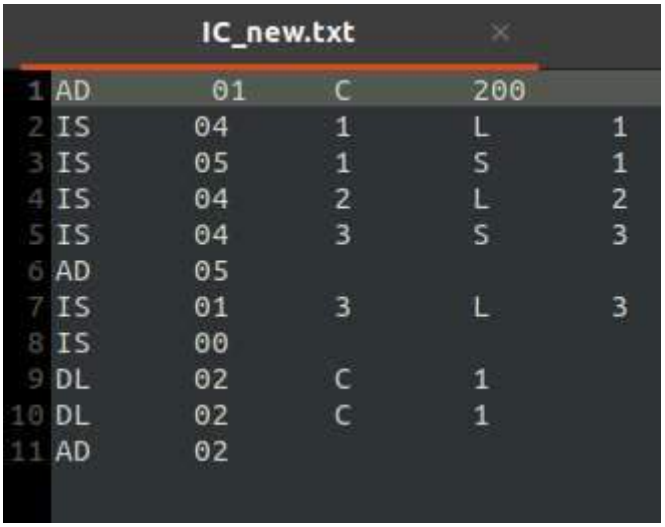
```
                            littab_file.readLine();
                        }
                        while (temp1 < temp2) {

                            System.out.print("00\t0\t00" +
littab_file.readLine().split("'")[1]);
                            if(temp1<(temp2-1)){
                                locptr++;

                                System.out.println();

                                System.out.print(locptr+"\t");
                            }
                            temp1++;
                        }
                        temp1 = temp2;
                        pooltabptr++;
                        if (pooltabptr < pooltab.size()) {
                            temp2 = pooltab.get(pooltabptr);
                        }
                    }
                    int j = 1;
                    if (s1.equals("02")) {        //if it is "END" stmt
                        String s;
                        while ((s = littab_file.readLine()) != null) {
                            if (j >= temp1)

                                System.out.print("00\t0\t00" + s.split("'")[1]);
                            j++;
                        }
                    }
                }



                if(s0.equals("DL")&&s1.equals("01")){      //if it is DC stmt

                    System.out.print("00\t0\t00"+sCurrentLine.split("'")[1]);

                }



                locptr++;

                System.out.println();
            }
            IC_file.close();
            symtab_file.close();
            littab_file.close();
            pooltab_file.close();

        } catch (IOException e) {
            e.printStackTrace();
        }

    }
}
```

PASS 2 - ASSEMBLER OUTPUT:

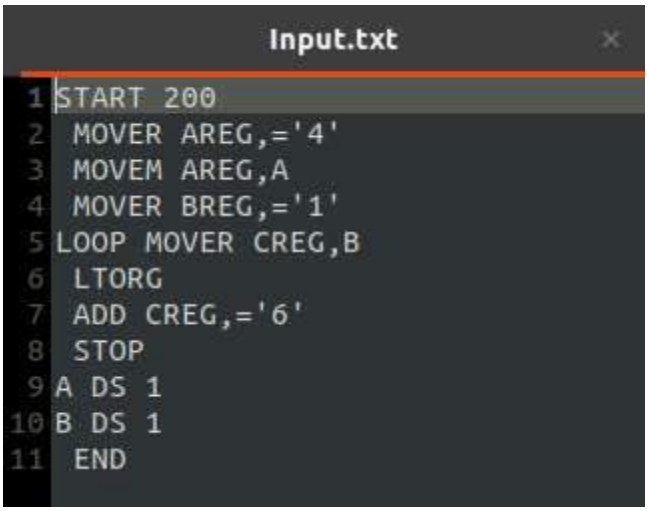
PASS- 2 OUTPUT:



IC_New.txt

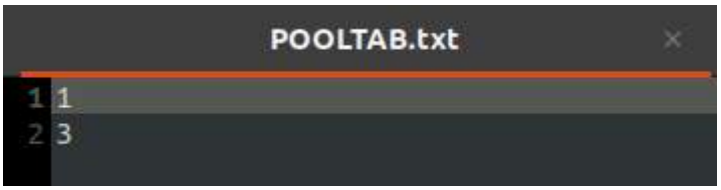


Input.txt



LITTAB.txt

POOLTAB.txt



SYMTAB.txt

Name: Jayesh Bharat Devre **Roll**: TE-A-36

Assignment No. 03 [PASS-1 Macro-processor]

Problem Statement: Design suitable data structures and implement pass-I of a two-pass macro-processor using OOP features in Java.

```
package com.company;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Hashtable;
import java.util.Map;

public class Pass1macro{


    public static void main(String[] args) {
        BufferedReader input_file = null;
        FileReader fr = null;

        FileWriter fw = null;
        BufferedWriter mdt_file = null;

        try {
            String filename = "C:\\Users\\Jayesh\\Desktop\\spos\\input.txt";
            fr = new FileReader(filename);
            input_file = new BufferedReader(fr);

            String filename1 = "C:\\Users\\Jayesh\\Desktop\\spos\\MDT.txt";
            fw = new FileWriter(filename1);
            mdt_file = new BufferedWriter(fw);

            String f1 = "C:\\Users\\Jayesh\\Desktop\\spos\\PNTAB.txt";
            FileWriter fw1 = new FileWriter(f1);
            BufferedWriter pn_file = new BufferedWriter(fw1);

            String f2 = "C:\\Users\\Jayesh\\Desktop\\spos\\EVNTAB.txt";
            FileWriter fw2 = new FileWriter(f2);
            BufferedWriter evn_file = new BufferedWriter(fw2);

            String f3 = "C:\\Users\\Jayesh\\Desktop\\spos\\SSNTAB.txt";
            FileWriter fw3 = new FileWriter(f3);
            BufferedWriter ssN_file = new BufferedWriter(fw3);

            String f4 = "C:\\Users\\Jayesh\\Desktop\\spos\\MNT.txt";
            FileWriter fw4 = new FileWriter(f4);
            BufferedWriter mnt_file = new BufferedWriter(fw4);

            String f5 = "C:\\Users\\Jayesh\\Desktop\\spos\\KPDTAB.txt";
            FileWriter fw5 = new FileWriter(f5);
            BufferedWriter kpd_file = new BufferedWriter(fw5);

            String f6 = "C:\\Users\\Jayesh\\Desktop\\spos\\SSTAB.txt";
            FileWriter fw6 = new FileWriter(f6);
            BufferedWriter ss_file = new BufferedWriter(fw6);

            String s;

            s = input_file.readLine();     //MACRO
            s = input_file.readLine();     //Macro name and params list

            String mnt[][] = new String[7][2];
            mnt[0][0] = "name";
            mnt[1][0] = "#PP";
            mnt[2][0] = "#KP";
            mnt[3][0] = "#EV";        //0th column for headers
            mnt[4][0] = "#MDTP";
            mnt[5][0] = "#KPDTP";
```

```
            mnt[6][0] = "#SSTP";

            mnt[1][1] = Integer.toString(0);
            mnt[2][1] = Integer.toString(0);
            mnt[3][1] = Integer.toString(0);       //1st column for values
            mnt[4][1] = Integer.toString(0);
            mnt[5][1] = Integer.toString(0);
            mnt[6][1] = Integer.toString(0);

            ArrayList<String> PNTAB = new ArrayList<String>();
            ArrayList<String> EVNTAB = new ArrayList<String>();
            ArrayList<String> SSNTAB = new ArrayList<String>();
            ArrayList<Integer> SSTAB = new ArrayList<Integer>();

            Hashtable<String, String> KPDTAB = new Hashtable<String, String>();

            int pp = 0;
            int kp = 0;
            int ev = 0;
            int mdt = 1;

            mnt[4][1] = Integer.toString(mdt);     //only one macro. so enter the MDT loc as
1.

            mnt[0][1] = s.split(" ")[1];


            int i = s.split("&|\\,").length;


            for (int j = 1; j < i; j += 2)            //+2, because "," and "&" contain a
null string between them. :  ,<null>&
            {
                String s1;
                if ((s1 = s.split("&|\\,")[j]).contains("="))
                {
                    kp++;
                    PNTAB.add(s1.split("=")[0]);               //param name
                    KPDTAB.put(s1.split("=")[0], s1.split("=")[1]);    //param default
value
                }
                else
                {
                    pp++;
                    PNTAB.add(s.split("&|\\,")[j]);
                }

            }


            mnt[1][1] = Integer.toString(pp);
            mnt[2][1] = Integer.toString(kp);

            while ((s = input_file.readLine()) != null)
            {
                mdt_file.write(mdt + "\t");        //always write mdt

                if (s.equals("MEND"))
                {
                    mdt_file.write("MEND");
                }
                else
                {

                    String s2;

                    if ((s.split(" ")[1]).equals("LCL"))
                    {
                        ev++;
                        s2 = s.split(" ")[2];      //LCL varname
                        EVNTAB.add(s2.split("&")[1]);
                        mnt[3][1]=Integer.toString(Integer.parseInt(mnt[3][1])+1);
                        mdt_file.write(s.split(" ")[1] + "\tE\t" +
(EVNTAB.indexOf(s2.split("&")[1]) + 1));
                    }
```

```
                    else if ((s2 = s.split(" ")[1]).equals("SET") || (s2 = s.split("
")[1]).equals("AIF")
                            || (s2 = s.split(" ")[1]).equals("AGO"))
                    {
                        if ((s.split(" ")[1]).equals("SET"))
                        {
                            s2 = s.split(" ")[0];

                            mdt_file.write("E\t" + (EVNTAB.indexOf(s2.split("&")[1]) + 1) +
"\t" + s.split(" ")[1] + "\t");

                            for (int z = 2; z < s.split(" ").length; z++)
                            {
                                String a = s.split(" ")[z];
                                if (a.contains("&"))
                                {
                                    if (EVNTAB.contains(a.split("&")[1]))
                                    {

                                        mdt_file.write("E\t" +
(EVNTAB.indexOf(a.split("&")[1]) + 1) + "\t");
                                    }

                                    else
                                    {
                                        mdt_file.write(s.split(" ")[z] + "\t");
                                    }
                                }

                                else
                                {
                                    mdt_file.write(a + "\t");
                                }
                            }
                        } // end of (SET)

                        else if ((s.split(" ")[1]).equals("AIF") || (s.split("
")[1]).equals("AGO"))
                        {
                            mdt_file.write(s.split(" ")[1] + "\t");


                            for (int y = 2; y < s.split(" ").length; y++)  //why 2? -
>answer.
                            {
                                String s3 = s.split(" ")[y];
                                if (s3.contains("&"))
                                {
                                    if (EVNTAB.contains(s3.split("&|\\)")[1]))
                                    {
                                        mdt_file.write("E\t" +
(EVNTAB.indexOf(s3.split("&|\\)")[1]) + 1) + "\t");
                                    }

                                    if (PNTAB.contains(s3.split("&|\\)")[1]))
                                    {
                                        mdt_file.write("P\t" +
(PNTAB.indexOf(s3.split("&|\\)")[1]) + 1) + "\t");
                                    }

                                }

                                else if (s3.contains("."))
                                {
                                    if (!(SSNTAB.contains(s3.split(" |\\.")[1])))  //if not
present in ssntab, add it! (fwd ref)
                                    {
                                        SSNTAB.add(s3.split(" |\\.")[1]);

                                    }

                                    mdt_file.write("S\t" + ((SSNTAB.indexOf(s3.split("
|\\.")[1]) + 1) + "\t"));

                                }
```

```
                                else
                                {
                                    mdt_file.write(s3 + "\t");
                                }

                            }//end of for loop

                        }  //end of (AIF||AGO)
                    }     //end of (SET||AIF||AGO)

                    else   // it is neither of LCL,SET,AIF,AGO.
                    {
                        String s4;
                        int len;
                        len = s.split(" |\\,").length;
                        for (int q = 0; q < len; q++)
                        {
                            s4 = s.split(" |\\,")[q];

                            if (s4.contains("."))
                            {
                                if (!(SSNTAB.contains(s4.split(" |\\.")[1])))
                                {
                                    SSNTAB.add(s4.split(" |\\.")[1]);

                                }
                            }

                            else if (s4.contains("&"))
                            {
                                if (PNTAB.contains(s4.split("&")[1]))
                                {
                                    mdt_file.write("P\t" + (PNTAB.indexOf(s4.split("&")[1])
+ 1) + "\t");

                                }

                                else if (EVNTAB.contains(s4.split("&")[1]))
                                {
                                    mdt_file.write("E\t" +
(EVNTAB.indexOf(s4.split("&")[1]) + 1) + "\t");
                                }
                            }

                            else if (s4.length() > 0)
                            {
                                mdt_file.write(s4 + "\t");
                            }
                        } //end of for loop
                    }


                    String l=s.split(" ")[0];
                    if((s.split(" ")[0].contains("."))){

                        int index=SSNTAB.indexOf(l.split(" |\\.")[1]);

                        SSTAB.add(index,mdt);     //write the value of MDT in the SSTAB, at
the index of the symbol in SSNTAB.
                    }
                }

                mdt++;
                mdt_file.write("\n");


            } //end of while loop



            for (int p = 0; p < PNTAB.size(); p++) {
                pn_file.write(PNTAB.get(p) + "\n");
            }
            for (int p = 0; p < EVNTAB.size(); p++) {
                evn_file.write(EVNTAB.get(p) + "\n");
            }
            for (int p = 0; p < SSNTAB.size(); p++) {
```

```
                ssN_file.write(SSNTAB.get(p) + "\n");
            }
            for (int z = 0; z < 7; z++) {
                mnt_file.write(mnt[z][0] + "\t" + mnt[z][1]+"\n");
            }

            for (Map.Entry m : KPDTAB.entrySet()) {
                kpd_file.write(m.getKey() + "\t" + m.getValue()+"\n");
            }

            for (int p = 0; p < SSTAB.size(); p++) {
                ss_file.write(SSTAB.get(p) + "\n");
            }


            mnt[3][1] = Integer.toString(ev);
            input_file.close();
            mdt_file.close();
            pn_file.close();
            evn_file.close();
            ssN_file.close();
            mnt_file.close();
            kpd_file.close();
            ss_file.close();
        } catch (

                IOException e) {
            e.printStackTrace();
        }

    }
}
```

Name: Jayesh Bharat Devre **Roll**: TE-A-36

Assignment No. 04 [PASS-2 Macro-processor]

Problem Statement: 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.

---

1. Pass 2 Macro Code:

```
package com.company;
import java.io.*;
import java.util.HashMap;
import java.util.Vector;

public class macroPass2 {
    public static void main(String[] Args) throws IOException{
        BufferedReader b1 = new BufferedReader(new FileReader("intermediate.txt"));
        BufferedReader b2 = new BufferedReader(new FileReader("mnt.txt"));
        BufferedReader b3 = new BufferedReader(new FileReader("mdt.txt"));
        BufferedReader b4 = new BufferedReader(new FileReader("kpdt.txt"));
        FileWriter f1 = new FileWriter("Pass2.txt");
        HashMap<Integer,String> aptab=new HashMap<Integer,String>();
        HashMap<String,Integer> aptabInverse=new HashMap<String,Integer>();
        HashMap<String,Integer> mdtpHash=new HashMap<String,Integer>();
        HashMap<String,Integer> kpdtpHash=new HashMap<String,Integer>();
        HashMap<String,Integer> kpHash=new HashMap<String,Integer>();
        HashMap<String,Integer> macroNameHash=new HashMap<String,Integer>();
        Vector<String>mdt=new Vector<String>();
        Vector<String>kpdt=new Vector<String>();
        String s,s1;
        int i,pp,kp,kpdtp,mdtp,paramNo;
        while((s=b3.readLine())!=null)
            mdt.addElement(s);
        while((s=b4.readLine())!=null)
            kpdt.addElement(s);
        while((s=b2.readLine())!=null){
            String word[]=s.split("\t");
            s1=word[0]+word[1];
            macroNameHash.put(word[0],1);
            kpHash.put(s1,Integer.parseInt(word[2]));
            mdtpHash.put(s1,Integer.parseInt(word[3]));
            kpdtpHash.put(s1,Integer.parseInt(word[4]));
        }
        while((s=b1.readLine())!=null){
            String b1Split[]=s.split("\\s");
            if(macroNameHash.containsKey(b1Split[0])){
                pp= b1Split[1].split(",").length-b1Split[1].split("=").length+1;
                kp=kpHash.get(b1Split[0]+Integer.toString(pp));
                mdtp=mdtpHash.get(b1Split[0]+Integer.toString(pp));
                kpdtp=kpdtpHash.get(b1Split[0]+Integer.toString(pp));
                String actualParams[]=b1Split[1].split(",");
                paramNo=1;
                for(int j=0;j<pp;j++){
                    aptab.put(paramNo, actualParams[paramNo-1]);
                    aptabInverse.put(actualParams[paramNo-1],paramNo);
                    paramNo++;
                }
                i=kpdtp-1;
                for(int j=0;j<kp;j++){
                    String temp[]=kpdt.get(i).split("\t");
                    aptab.put(paramNo,temp[1]);
                    aptabInverse.put(temp[0],paramNo);
                    i++;
                    paramNo++;
                }
                i=pp+1;
                while(i<=actualParams.length){
                    String initializedParams[]=actualParams[i-1].split("=");

aptab.put(aptabInverse.get(initializedParams[0].substring(1,initializedParams[0].length()))
```

```
,initializedParams[1].substring(0,initializedParams[1].length()));
                    i++;
                }
                i=mdtp-1;
                while(mdt.get(i).compareToIgnoreCase("MEND")!=0){
                    f1.write("+ ");
                    for(int j=0;j<mdt.get(i).length();j++){
                        if(mdt.get(i).charAt(j)=='#')
                            f1.write(aptab.get(Integer.parseInt("" +
mdt.get(i).charAt(++j))));
                        else
                            f1.write(mdt.get(i).charAt(j));
                    }
                    f1.write("\n");
                    i++;
                }
                aptab.clear();
                aptabInverse.clear();
            }
            else
                f1.write("+ "+s+"\n");
        }
        b1.close();
        b2.close();
        b3.close();
        b4.close();
        f1.close();
    }
}
```

/*
OUTPUT:

```
Intermediate - -
M1 10,20,&b=CREG
M2 100,200,&u=&AREG,&v=&BREG



Kpdt--
a     AREG
b     -
u     CREG
v     DREG

pass2 --
+ MOVE AREG,10
+ ADD AREG,='1'
+ MOVER AREG,20
+ ADD AREG,='5'
+ MOVER &AREG,100
+ MOVER &BREG,200
+ ADD &AREG,='15'
+ ADD &BREG,='10'


MNT --
M1    2     2     1     1
M2    2     2     6     3


MDT --
MOVE #3,#1
ADD #3,='1'
MOVER #3,#2
ADD #3,='5'
MEND
```

```
MOVER #3,#1
MOVER #4,#2
ADD #3,='15'
ADD #4,='10'
MEND
*/
```

Name: Jayesh Bharat Devre **Roll**: TE-A-36

Assignment No. 05 [LEX Program]

Problem Statement: Write a program using Lex specifications to implement lexical analysis Phase of compiler to generate tokens of subset of Java program.

---

1. Code b2.l:

```
2.
  %{
          FILE* yyin;
          %}

          DATATYPE "int"|"char"|"float"|"double" KEYWORDS "class"|"static"
          DIGIT [0-9] NUMBER {DIGIT}+ TEXT [a-zA-Z]
          IDENTIFIER {TEXT}({DIGIT}|{TEXT}|"_")*
          ACCESS "public"|"private"|"protected" CONDITIONAL "if"|"else"|"else
  if"|"switch" LOOP "for"|"while"|"do"
          FUNCTION {ACCESS}{DATATYPE}{IDENTIFIER}"("({DATATYPE}{IDENTIFIER})*")"

          %%
          [ \n\t]+ ;
          {DATATYPE} {printf("%s == DATATYPE\n",yytext);}
          {KEYWORDS} {printf("%s == KEYWORDS\n",yytext);}
          {NUMBER} {printf("%s == NUMBER\n",yytext);}
          {IDENTIFIER} {printf("%s == IDENTIFIER\n",yytext);}

          {CONDITIONAL} {printf("%s == CONDITIONAL\n",yytext);}

          {FUNCTION} {printf("%s == FUNCTION\n",yytext);}
          . ;
          %%

          int yywrap(){

          }


          int main(int argc,char* argv[]){ yyin= fopen(argv[1],"r"); yylex();
          fclose(yyin); return 0;
          }
  2. Demo.java Code:

          import java.io.BufferedReader;
          import java.io.InputStreamReader;
          import java.util.Arrays;

  public class demo
  {


      public static void main(String[] args)throws Exception { int hit=0;
          int miss=0;

          BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

          System.out.println("Enter total no of frames"); int
  noFrames=Integer.parseInt(br.readLine());

          int[] frames=new int[noFrames]; int[] lruTime=new int[noFrames];

          System.out.println("Enter total no of pages"); int totalPages =
  Integer.parseInt(br.readLine());

          for(int i=0;i<totalPages;i++){
              System.out.println("Enter page value"); int page=
  Integer.parseInt(br.readLine());
              int searchIndex=isPresent(frames, page ); if(searchIndex!=-1){
  // page fonud
                  hit++; lruTime[searchIndex]=i;
                  System.out.println("Page Hit");
```

```
            }
            else{
                System.out.println("Page Miss"); miss++;

// page not found
                int emptyindex=isEmpty(frames); if(emptyindex!=-1){
// if frame is empty
                    frames[emptyindex]=page;




//user lru algo to find replace location


                }
                else{

                    lruTime[emptyindex]=i;

                    int minLocationIndex=lru(lruTime);

                    System.out.println("Replace "+
                            frames[minLocationIndex]);

                    frames[minLocationIndex]=page; lruTime[minLocationIndex]=i;

                }

            }

        }


        System.out.println("Total page hit" + hit); System.out.println("Total Page miss
" + miss); System.out.println(Arrays.toString(frames));

    }

    public static int lru(int[] lruTime){
        int min = 9999; int index = -1;
        for(int i=0;i<lruTime.length;i++){

            if(min>lruTime[i]){
                min=lruTime[i]; index=i;
            }

        }

        return index;
    }

    public static int isEmpty(int[] frames){

        for(int i=0;i<frames.length;i++)
        {
            if(frames[i]==0){
                return i;
            }
        }

        return -1;
    }

    public static int isPresent(int[] frames, int search){

        for(int i=0;i<frames.length;i++){ if(frames[i]==search)

            return i;
        }

        return -1;
    }
```

```
    }
```

3. OUTPUT:

```
import == IDENTIFIER
java == IDENTIFIER
io == IDENTIFIER
BufferedReader == IDENTIFIER
import == IDENTIFIER
java == IDENTIFIER
io == IDENTIFIER
InputStreamReader == IDENTIFIER
import == IDENTIFIER
java == IDENTIFIER
util == IDENTIFIER
Arrays == IDENTIFIER
public == IDENTIFIER
```

class == KEYWORDS
demo == IDENTIFIER
public == IDENTIFIER
static == KEYWORDS
void == IDENTIFIER
main == IDENTIFIER
String == IDENTIFIER
args == IDENTIFIER
throws == IDENTIFIER
Exception == IDENTIFIER
int == DATATYPE
hit == IDENTIFIER 0
== NUMBER
int == DATATYPE miss
== IDENTIFIER 0 ==
NUMBER
BufferedReader == IDENTIFIER
br == IDENTIFIER
new == IDENTIFIER
BufferedReader == IDENTIFIER
new == IDENTIFIER
InputStreamReader == IDENTIFIER
System == IDENTIFIER
in == IDENTIFIER
System == IDENTIFIER
out == IDENTIFIER
println == IDENTIFIER
Enter == IDENTIFIER
total == IDENTIFIER no
== IDENTIFIER
of == IDENTIFIER
frames == IDENTIFIER
int == DATATYPE
noFrames == IDENTIFIER
Integer == IDENTIFIER
parseInt == IDENTIFIER br
== IDENTIFIER
readLine == IDENTIFIER
int == DATATYPE
frames == IDENTIFIER
new == IDENTIFIER int
== DATATYPE
noFrames == IDENTIFIER
int == DATATYPE
lruTime == IDENTIFIER
new == IDENTIFIER
int == DATATYPE
noFrames == IDENTIFIER
System == IDENTIFIER out
== IDENTIFIER
println == IDENTIFIER
Enter == IDENTIFIER

total == IDENTIFIER
no == IDENTIFIER of
== IDENTIFIER
pages == IDENTIFIER
int == DATATYPE
totalPages == IDENTIFIER
Integer == IDENTIFIER
parseInt == IDENTIFIER br
== IDENTIFIER
readLine == IDENTIFIER
for == IDENTIFIER
int == DATATYPE i
== IDENTIFIER 0
== NUMBER
i == IDENTIFIER
totalPages == IDENTIFIER i
== IDENTIFIER
System == IDENTIFIER
out == IDENTIFIER
println == IDENTIFIER
Enter == IDENTIFIER
page == IDENTIFIER
value == IDENTIFIER
int == DATATYPE page
== IDENTIFIER
Integer == IDENTIFIER
parseInt == IDENTIFIER
br == IDENTIFIER
readLine == IDENTIFIER
int == DATATYPE
searchIndex == IDENTIFIER
isPresent == IDENTIFIER
frames == IDENTIFIER page
== IDENTIFIER
if == IDENTIFIER
searchIndex == IDENTIFIER
1 == NUMBER
page == IDENTIFIER
fonud == IDENTIFIER
hit == IDENTIFIER
lruTime == IDENTIFIER
searchIndex == IDENTIFIER i
== IDENTIFIER
System == IDENTIFIER
out == IDENTIFIER
println == IDENTIFIER
Page == IDENTIFIER
Hit == IDENTIFIER else
== IDENTIFIER System
== IDENTIFIER out ==
IDENTIFIER
println == IDENTIFIER

Page == IDENTIFIER
Miss == IDENTIFIER
miss == IDENTIFIER
page == IDENTIFIER
not == IDENTIFIER
found == IDENTIFIER
int == DATATYPE
emptyindex == IDENTIFIER
isEmpty == IDENTIFIER
frames == IDENTIFIER
if == IDENTIFIER
emptyindex == IDENTIFIER
1 == NUMBER
if == IDENTIFIER
frame == IDENTIFIER
is == IDENTIFIER
empty == IDENTIFIER
frames == IDENTIFIER
emptyindex == IDENTIFIER
page == IDENTIFIER
lruTime == IDENTIFIER
emptyindex == IDENTIFIER i
== IDENTIFIER
else == IDENTIFIER
user == IDENTIFIER
lru == IDENTIFIER
algo == IDENTIFIER
to == IDENTIFIER
find == IDENTIFIER
replace == IDENTIFIER
location == IDENTIFIER
int == DATATYPE
minLocationIndex == IDENTIFIER
lru == IDENTIFIER
lruTime == IDENTIFIER
System == IDENTIFIER
out == IDENTIFIER
println == IDENTIFIER
Replace == IDENTIFIER
frames == IDENTIFIER
minLocationIndex == IDENTIFIER
frames == IDENTIFIER
minLocationIndex == IDENTIFIER
page == IDENTIFIER
lruTime == IDENTIFIER
minLocationIndex == IDENTIFIER i
== IDENTIFIER
System == IDENTIFIER
out == IDENTIFIER
println == IDENTIFIER
Total == IDENTIFIER
page == IDENTIFIER

hit == IDENTIFIER
hit == IDENTIFIER
System == IDENTIFIER
out == IDENTIFIER
println == IDENTIFIER
Total == IDENTIFIER
Page == IDENTIFIER
miss == IDENTIFIER
miss == IDENTIFIER
System == IDENTIFIER
out == IDENTIFIER
println == IDENTIFIER
Arrays == IDENTIFIER
toString == IDENTIFIER
frames == IDENTIFIER
public == IDENTIFIER
static == KEYWORDS int
== DATATYPE
lru == IDENTIFIER
int == DATATYPE
lruTime == IDENTIFIER int
== DATATYPE
min == IDENTIFIER
9999 == NUMBER int
== DATATYPE
index == IDENTIFIER
1 == NUMBER
for == IDENTIFIER
int == DATATYPE
i == IDENTIFIER
0 == NUMBER
i == IDENTIFIER
lruTime == IDENTIFIER
length == IDENTIFIER
i == IDENTIFIER if
== IDENTIFIER
min == IDENTIFIER
lruTime == IDENTIFIER i
== IDENTIFIER
min == IDENTIFIER
lruTime == IDENTIFIER i
== IDENTIFIER
index == IDENTIFIER i
== IDENTIFIER
return == IDENTIFIER
index == IDENTIFIER
public == IDENTIFIER
static == KEYWORDS
int == DATATYPE
isEmpty == IDENTIFIER
int == DATATYPE
frames == IDENTIFIER

for == IDENTIFIER
int == DATATYPE
i == IDENTIFIER
0 == NUMBER
i == IDENTIFIER
frames == IDENTIFIER
length == IDENTIFIER i
== IDENTIFIER
if == IDENTIFIER
frames == IDENTIFIER i
== IDENTIFIER
0 == NUMBER
return == IDENTIFIER i
== IDENTIFIER
return == IDENTIFIER
1 == NUMBER
public == IDENTIFIER
static == KEYWORDS
int == DATATYPE
isPresent == IDENTIFIER
int == DATATYPE
frames == IDENTIFIER
int == DATATYPE
search == IDENTIFIER
for == IDENTIFIER
int == DATATYPE i
== IDENTIFIER 0
== NUMBER
i == IDENTIFIER
frames == IDENTIFIER
length == IDENTIFIER i
== IDENTIFIER
if == IDENTIFIER
frames == IDENTIFIER i
== IDENTIFIER
search == IDENTIFIER
return == IDENTIFIER i
== IDENTIFIER
return == IDENTIFIER
1 == NUMBER

Name: Jayesh Bharat Devre **Roll**: TE-A-36

Assignment No. 06 [LEX Program]

Problem Statement: Write a program using Lex specifications to implement lexical analysis Phase of compiler to count no. of words, lines and characters of given Input file.

---

1. Code b3.l:

```
%{
int no_line=0;
int no_space=0;
int no_char=0;
int no_words=0;
#include<string.h>
%}

%%
([a-zA-Z])+ {no_words++; no_char+=strlen(yytext);}
[" "] {no_space++;}
["\n"] {no_line++;}
. ;

%%


int yywrap(){

}

int main(int argc,char* argv[]){
   yyin=fopen("test.txt","r");
   yylex();
   printf("Total Spaces %d\n",no_space);
   printf("Total Words %d\n",no_words);
   printf("Total Line %d\n",no_line);
   no_char+=no_space;
   printf("Total Char %d\n",no_char);
   fclose(yyin);
}
```

2.text.txt File**:**
// *Content of text.txt File*

The earliest foundations of what would become computer science predate the invention of the modern digital computer. Machines for calculating fixed numerical tasks such as the abacus have existed since antiquity, aiding in computations such as multiplication and division. Algorithms for performing computations have existed since antiquity, even before the development of sophisticated computing equipment.

Computer science, the study of computers and computing, including their theoretical and algorithmic foundations, hardware and software, and their uses for processing information. The discipline of computer science includes the study of algorithms and data structures, computer and network design, modeling data and information processes, and artificial intelligence. Computer science draws some of its foundations from mathematics and engineering and therefore incorporates techniques from areas such as queueing theory, probability and statistics, and electronic circuit design. Computer science also makes heavy use of hypothesis testing and experimentation during the conceptualization, design, measurement, and refinement of new algorithms, information structures, and computer architectures.

Output:

Total spaces 155
Total words 157
Total Line 3
Total Char 1180

Name: Jayesh Bharat Devre **Roll**: TE-A-36

Assignment No. 07 [LEX Program]

Problem Statement: Write a program using YACC specifications to implement syntax analysis phase of compiler to recognize simple and compound sentences given in input file

```
%{
   #include<stdio.h>
   int simple=0;
%}

%%
[ \t\n][aA][nN][dD][ \t\n] {simple=1;}
[ \t\n][bB][uU][tT][ \t\n] {simple=1;}
[ \t\n][oO][rR][ \t\n] {simple=1;}
. ;
%%

int yywrap(){

}

int main(){
   printf("Enter sentence: \n");
   yylex();
   if(simple==1){
      printf("compound\n\n");
   }
   else{
      printf("simple\n\n");
   }
   return 0;
}
```

Output:
Enter sentence:
Hi Friends

Simple

Enter sentence:
Hi friends or chai pilo

compound

Name: Jayesh Bharat Devre **Roll**: TE-A-36

Assignment No. 08

Problem Statement: Write a Java program (using OOP features) to implement following scheduling algorithms:
FCFS , SJF (Preemptive), Priority (Non - Preemptive) and Round Robin (Preemptive)

---

1. FCFS Program:

```
package com.company;
// Java program for implementation of FCFS
// scheduling

import java.text.ParseException;

class FCFS {

    // Function to find the waiting time for all
    // processes
    static void findWaitingTime(int processes[], int n,
                                int bt[], int wt[]) {
        // waiting time for first process is 0
        wt[0] = 0;

        // calculating waiting time
        for (int i = 1; i < n; i++) {
            wt[i] = bt[i - 1] + wt[i - 1];
        }
    }

    // Function to calculate turn around time
    static void findTurnAroundTime(int processes[], int n,
                                   int bt[], int wt[], int tat[]) {
        // calculating turnaround time by adding
        // bt[i] + wt[i]
        for (int i = 0; i < n; i++) {
            tat[i] = bt[i] + wt[i];
        }
    }

    //Function to calculate average time
    static void findavgTime(int processes[], int n, int bt[]) {
        int wt[] = new int[n], tat[] = new int[n];
        int total_wt = 0, total_tat = 0;

        //Function to find waiting time of all processes
        findWaitingTime(processes, n, bt, wt);

        //Function to find turn around time for all processes
        findTurnAroundTime(processes, n, bt, wt, tat);

        //Display processes along with all details
        System.out.printf("Processes Burst time Waiting"
                +" time Turn around time\n");

        // Calculate total waiting time and total turn
        // around time
        for (int i = 0; i < n; i++) {
            total_wt = total_wt + wt[i];
            total_tat = total_tat + tat[i];
            System.out.printf(" %d ", (i + 1));
            System.out.printf("     %d ", bt[i]);
            System.out.printf("     %d", wt[i]);
            System.out.printf("     %d\n", tat[i]);
        }
        float s = (float)total_wt /(float) n;
        int t = total_tat / n;
        System.out.printf("Average waiting time = %f", s);
        System.out.printf("\n");
        System.out.printf("Average turn around time = %d ", t);
    }
```

```
    // Driver code
    public static void main(String[] args) throws ParseException {
        //process id's
        int processes[] = {1, 2, 3};
        int n = processes.length;

        //Burst time of all processes
        int burst_time[] = {10, 5, 8};

        findavgTime(processes, n, burst_time);

    }
}
```

Output:
"C:\Program Files\Java\jdk-15.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.3.2\lib\idea_rt.jar=60410:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.3.2\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\Jayesh\IdeaProjects\spos\out\production\spos com.company.FCFS
Processes Burst time Waiting time Turn around time
1 10 0 10
2 5 10 15
3 8 15 23
Average waiting time = 8.333333
Average turn around time = 16
Process finished with exit code 0

2.Shortest Job First Program:

```
package com.company;
import java.util.*;

class SJF {
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println ("enter no of process:");
        int n = sc.nextInt();
        int pid[] = new int[n];
        int at[] = new int[n]; // at means arrival time
        int bt[] = new int[n]; // bt means burst time
        int ct[] = new int[n]; // ct means complete time
        int ta[] = new int[n]; // ta means turn around time
        int wt[] = new int[n];  //wt means waiting time
        int f[] = new int[n];  // f means it is flag it checks process is completed or not
        int st=0, tot=0;
        float avgwt=0, avgta=0;

        for(int i=0;i<n;i++)
        {
            System.out.println ("enter process " + (i+1) + " arrival time:");
            at[i] = sc.nextInt();
            System.out.println ("enter process " + (i+1) + " brust time:");
            bt[i] = sc.nextInt();
            pid[i] = i+1;
            f[i] = 0;
        }

        boolean a = true;
        while(true)
        {
            int c=n, min=999;
            if (tot == n) // total no of process = completed process loop will be
terminated
                break;

            for (int i=0; i<n; i++)
            {
```

```
                /*
                 * If i'th process arrival time <= system time and its flag=0 and burst<min
                 * That process will be executed first
                 */
                if ((at[i] <= st) && (f[i] == 0) && (bt[i]<min))
                {
                    min=bt[i];
                    c=i;
                }
            }

            /* If c==n means c value can not updated because no process arrival time<
system time so we increase the system time */
            if (c==n)
                st++;
            else
            {
                ct[c]=st+bt[c];
                st+=bt[c];
                ta[c]=ct[c]-at[c];
                wt[c]=ta[c]-bt[c];
                f[c]=1;
                tot++;
            }
        }

        System.out.println("\npid  arrival brust  complete turn waiting");
        for(int i=0;i<n;i++)
        {
            avgwt+= wt[i];
            avgta+= ta[i];

System.out.println(pid[i]+"\t"+at[i]+"\t"+bt[i]+"\t"+ct[i]+"\t"+ta[i]+"\t"+wt[i]);
        }
        System.out.println ("\naverage tat is "+ (float)(avgta/n));
        System.out.println ("average wt is "+ (float)(avgwt/n));
        sc.close();
    }
}
```

OUTPUT:

pid arrival brust complete turn waiting

1	**0**	**3**	**6**	**6**	**3**
2	**0**	**1**	**1**	**1**	**0**
3	**0**	**2**	**3**	**3**	**1**

average tat is 3.3333333
average wt is 1.3333334

3.Priority Program:

```
package com.company;
import java.util.Scanner;
class NonPreemptivePriorityCPUSchedulingAlgorithm
{

    int burstTime[];
```

```
    int priority[];
    int arrivalTime[];
    String[] processId;
    int numberOfProcess;

    void getProcessData(Scanner input)
    {
        System.out.print("Enter the number of Process for Scheduling          : ");
        int inputNumberOfProcess = input.nextInt();
        numberOfProcess = inputNumberOfProcess;
        burstTime = new int[numberOfProcess];
        priority = new int[numberOfProcess];
        arrivalTime = new int[numberOfProcess];
        processId = new String[numberOfProcess];
        String st = "P";
        for (int i = 0; i < numberOfProcess; i++)
        {
            processId[i] = st.concat(Integer.toString(i));
            System.out.print("Enter the burst time   for Process - " + (i) + " : ");
            burstTime[i] = input.nextInt();
            System.out.print("Enter the arrival time for Process - " + (i) + " : ");
            arrivalTime[i] = input.nextInt();
            System.out.print("Enter the priority     for Process - " + (i) + " : ");
            priority[i] = input.nextInt();
        }
    }

    void sortAccordingArrivalTimeAndPriority(int[] at, int[] bt, int[] prt, String[] pid)
    {

        int temp;
        String stemp;
        for (int i = 0; i < numberOfProcess; i++)
        {

            for (int j = 0; j < numberOfProcess - i - 1; j++)
            {
                if (at[j] > at[j + 1])
                {
                    //swapping arrival time
                    temp = at[j];
                    at[j] = at[j + 1];
                    at[j + 1] = temp;

                    //swapping burst time
                    temp = bt[j];
                    bt[j] = bt[j + 1];
                    bt[j + 1] = temp;

                    //swapping priority
                    temp = prt[j];
                    prt[j] = prt[j + 1];
                    prt[j + 1] = temp;

                    //swapping process identity
                    stemp = pid[j];
                    pid[j] = pid[j + 1];
                    pid[j + 1] = stemp;

                }
                //sorting according to priority when arrival timings are same
                if (at[j] == at[j + 1])
                {
                    if (prt[j] > prt[j + 1])
                    {
                        //swapping arrival time
                        temp = at[j];
                        at[j] = at[j + 1];
                        at[j + 1] = temp;

                        //swapping burst time
                        temp = bt[j];
                        bt[j] = bt[j + 1];
                        bt[j + 1] = temp;

                        //swapping priority
```

```
                        temp = prt[j];
                        prt[j] = prt[j + 1];
                        prt[j + 1] = temp;

                        //swapping process identity
                        stemp = pid[j];
                        pid[j] = pid[j + 1];
                        pid[j + 1] = stemp;

                    }
                }
            }

        }
    }

    void priorityNonPreemptiveAlgorithm()
    {
        int finishTime[] = new int[numberOfProcess];
        int bt[] = burstTime.clone();
        int at[] = arrivalTime.clone();
        int prt[] = priority.clone();
        String pid[] = processId.clone();
        int waitingTime[] = new int[numberOfProcess];
        int turnAroundTime[] = new int[numberOfProcess];

        sortAccordingArrivalTimeAndPriority(at, bt, prt, pid);

        //calculating waiting & turn-around time for each process
        finishTime[0] = at[0] + bt[0];
        turnAroundTime[0] = finishTime[0] - at[0];
        waitingTime[0] = turnAroundTime[0] - bt[0];

        for (int i = 1; i < numberOfProcess; i++)
        {
            finishTime[i] = bt[i] + finishTime[i - 1];
            turnAroundTime[i] = finishTime[i] - at[i];
            waitingTime[i] = turnAroundTime[i] - bt[i];
        }
        float sum = 0;
        for (int n : waitingTime)
        {
            sum += n;
        }
        float averageWaitingTime = sum / numberOfProcess;

        sum = 0;
        for (int n : turnAroundTime)
        {
            sum += n;
        }
        float averageTurnAroundTime = sum / numberOfProcess;

        //print on console the order of processes along with their finish time & turn
around time
        System.out.println("Priority Scheduling Algorithm : ");
        System.out.format("%20s%20s%20s%20s%20s%20s%20s\n", "ProcessId", "BurstTime",
"ArrivalTime", "Priority", "FinishTime", "WaitingTime", "TurnAroundTime");
        for (int i = 0; i < numberOfProcess; i++) {
            System.out.format("%20s%20d%20d%20d%20d%20d%20d\n", pid[i], bt[i], at[i],
prt[i], finishTime[i], waitingTime[i], turnAroundTime[i]);
        }

        System.out.format("%100s%20f%20f\n", "Average", averageWaitingTime,
averageTurnAroundTime);
    }

    public static void main(String[] args)
    {
        Scanner input = new Scanner(System.in);
        NonPreemptivePriorityCPUSchedulingAlgorithm obj = new
NonPreemptivePriorityCPUSchedulingAlgorithm();
        obj.getProcessData(input);
        obj.priorityNonPreemptiveAlgorithm();
    }
}
```

OUTPUT:

```
Enter the number of Process for Scheduling          : 3
Enter the burst time   for Process - 0 : 4
Enter the arrival time for Process - 0 : 0
Enter the priority     for Process - 0 : 1
Enter the burst time   for Process - 1 : 3
Enter the arrival time for Process - 1 : 0
Enter the priority     for Process - 1 : 2
Enter the burst time   for Process - 2 : 7
Enter the arrival time for Process - 2 : 6
Enter the priority     for Process - 2 : 1
Priority Scheduling Algorithm :
```

ProcessId	BurstTime	ArrivalTime	Priority	FinishTime	WaitingTime	TurnAroundTime
P0	4	0	1	4	0	4
P1	3	0	2	7	4	7
P2	7	6	1	14	1	8
				Average	1.666667	6.333333

```
Process finished with exit code 0
```

4.Round Robin Program:

```java
package com.company;
// Java program for implementation of RR scheduling

class GFG
{
    // Method to find the waiting time for all
    // processes
    static void findWaitingTime(int processes[], int n,
                                int bt[], int wt[], int quantum)
    {
        // Make a copy of burst times bt[] to store remaining
        // burst times.
        int rem_bt[] = new int[n];
        for (int i = 0 ; i < n ; i++)
            rem_bt[i] = bt[i];

        int t = 0; // Current time

        // Keep traversing processes in round robin manner
        // until all of them are not done.
        while(true)
        {
            boolean done = true;

            // Traverse all processes one by one repeatedly
            for (int i = 0 ; i < n; i++)
            {
                // If burst time of a process is greater than 0
                // then only need to process further
                if (rem_bt[i] > 0)
                {
                    done = false; // There is a pending process

                    if (rem_bt[i] > quantum)
                    {
                        // Increase the value of t i.e. shows
                        // how much time a process has been processed
                        t += quantum;

                        // Decrease the burst_time of current process
                        // by quantum
                        rem_bt[i] -= quantum;
                    }

                    // If burst time is smaller than or equal to
                    // quantum. Last cycle for this process
                    else
                    {
```

```
                    // Increase the value of t i.e. shows
                    // how much time a process has been processed
                    t = t + rem_bt[i];

                    // Waiting time is current time minus time
                    // used by this process
                    wt[i] = t - bt[i];

                    // As the process gets fully executed
                    // make its remaining burst time = 0
                    rem_bt[i] = 0;
                }
            }
        }

        // If all processes are done
        if (done == true)
            break;
    }
}

// Method to calculate turn around time
static void findTurnAroundTime(int processes[], int n,
                               int bt[], int wt[], int tat[])
{
    // calculating turnaround time by adding
    // bt[i] + wt[i]
    for (int i = 0; i < n ; i++)
        tat[i] = bt[i] + wt[i];
}

// Method to calculate average time
static void findavgTime(int processes[], int n, int bt[],
                        int quantum)
{
    int wt[] = new int[n], tat[] = new int[n];
    int total_wt = 0, total_tat = 0;

    // Function to find waiting time of all processes
    findWaitingTime(processes, n, bt, wt, quantum);

    // Function to find turn around time for all processes
    findTurnAroundTime(processes, n, bt, wt, tat);

    // Display processes along with all details
    System.out.println("Processes " + " Burst time " +
            " Waiting time " + " Turn around time");

    // Calculate total waiting time and total turn
    // around time
    for (int i=0; i<n; i++)
    {
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
        System.out.println(" " + (i+1) + "\t\t" + bt[i] +"\t " +
                wt[i] +"\t\t " + tat[i]);
    }

    System.out.println("Average waiting time = " +
            (float)total_wt / (float)n);
    System.out.println("Average turn around time = " +
            (float)total_tat / (float)n);
}

// Driver Method
public static void main(String[] args)
{
    // process id's
    int processes[] = { 1, 2, 3};
    int n = processes.length;

    // Burst time of all processes
    int burst_time[] = {10, 5, 8};

    // Time quantum
    int quantum = 2;
```

```
        findavgTime(processes, n, burst_time, quantum);
    }
}
```

OUTPUT:

Processes	Burst time	Waiting time	Turn around time
1	10	13	23
2	5	10	15
3	8	13	21

Average waiting time = 12.0
Average turn around time = 19.666666

Process finished with exit code 0

Name: Jayesh Bharat Devre **Roll**: TE-A-36

Assignment No. 09

Problem Statement: Write a Java program to implement Banker's Algorithm

1. Banker's Algorithm Program**:**

```
package com.company;
//Java Program for Bankers Algorithm
 class Bankersalgo
{
    int n = 5; // Number of processes
    int m = 3; // Number of resources
    int need[][] = new int[n][m];
    int [][]max;
    int [][]alloc;
    int []avail;
    int safeSequence[] = new int[n];

    void initializeValues()
    {
        // P0, P1, P2, P3, P4 are the Process names here
        // Allocation Matrix
        alloc = new int[][] { { 0, 1, 0 }, //P0
                { 2, 0, 0 }, //P1
                { 3, 0, 2 }, //P2
                { 2, 1, 1 }, //P3
                { 0, 0, 2 } }; //P4

        // MAX Matrix
        max = new int[][] { { 7, 5, 3 }, //P0
                { 3, 2, 2 }, //P1
                { 9, 0, 2 }, //P2
                { 2, 2, 2 }, //P3
                { 4, 3, 3 } }; //P4

        // Available Resources
        avail = new int[] { 3, 3, 2 };
    }

    void isSafe()
    {
        int count=0;

        //visited array to find the already allocated process
        boolean visited[] = new boolean[n];
        for (int i = 0;i < n; i++)
        {
            visited[i] = false;
        }

        //work array to store the copy of available resources
        int work[] = new int[m];
        for (int i = 0;i < m; i++)
        {
            work[i] = avail[i];
        }

        while (count<n)
        {
            boolean flag = false;
            for (int i = 0;i < n; i++)
            {
                if (visited[i] == false)
                {
                    int j;
                    for (j = 0;j < m; j++)
                    {
                        if (need[i][j] > work[j])
                            break;
                    }
                    if (j == m)
```

```
                    {
                        safeSequence[count++]=i;
                        visited[i]=true;
                        flag=true;

                        for (j = 0;j < m; j++)
                        {
                            work[j] = work[j]+alloc[i][j];
                        }
                    }
                }
            }
            if (flag == false)
            {
                break;
            }
        }
        if (count < n)
        {
            System.out.println("The System is UnSafe!");
        }
        else
        {
            //System.out.println("The given System is Safe");
            System.out.println("Following is the SAFE Sequence");
            for (int i = 0;i < n; i++)
            {
                System.out.print("P" + safeSequence[i]);
                if (i != n-1)
                    System.out.print(" -> ");
            }
        }
    }

    void calculateNeed()
    {
        for (int i = 0;i < n; i++)
        {
            for (int j = 0;j < m; j++)
            {
                need[i][j] = max[i][j]-alloc[i][j];
            }
        }
    }

    public static void main(String[] args)
    {
        int i, j, k;
        Bankersalgo bg = new Bankersalgo();

        bg.initializeValues();
        //Calculate the Need Matrix
        bg.calculateNeed();

        // Check whether system is in safe state or not
        bg.isSafe();
    }
}
```

OUTPUT:

Following is the SAFE Sequence

P1 -> P3 -> P4 -> P0 -> P2

Process finished with exit code 0

Name: Jayesh Bharat Devre **Roll**: TE-A-36

Assignment No. 10 [UNIX System Calls]

Problem Statement: To write a program to implement UNIX system calls like for process Management.

---

1. Code**:**

Problem Statement : Write a C program to create a child process using fork system call. Display Status of running processes used in child process(EXEC) & terminate child process before completion of parent task(wait).

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>


int main()
{
        pid_t pid , ppid , p_status ;
        int status ;
        printf("parent process created \n");
        pid = fork();
        if(pid ==0)
        {
                printf("child created succesfull\n");
                printf("child process id : %d \n", pid);
                sleep(10);
                printf("child after sleep \n");
                execlp("/bin/ps","ps",NULL);

                printf("child terminating\n");
                exit(0);
        }
        else
        {
                printf("parent still executing");
                p_status = wait(&status);
                printf("status : %d \n",status);
                printf("p_status :%d \n",p_status);
                sleep(10);
                printf("parent after sleep\n");
                ppid = getppid();
                printf("parent process id : %d\n",ppid);
                printf("parent terminating\n");
                exit(0);
        }
```

```
        return 0;
}
```

Name: Jayesh Bharat Devre **Roll**: TE-A-36

Assignment No. 12

Problem Statement: To write a java program (using OOP feature) to implement LRU & Optimal algorithm for Page Replacement.

---

1. LRU (Last Recently Used) Program**:**

```
package com.company;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Arrays;

class LRU
{


    public static void main(String[] args)throws Exception { int hit=0;
        int miss=0;

        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

        System.out.println("Enter total no of frames"); int
noFrames=Integer.parseInt(br.readLine());

        int[] frames=new int[noFrames]; int[] lruTime=new int[noFrames];

        System.out.println("Enter total no of pages"); int totalPages =
Integer.parseInt(br.readLine());

        for(int i=0;i<totalPages;i++){
            System.out.println("Enter page value"); int page=
Integer.parseInt(br.readLine());
            int searchIndex=isPresent(frames, page ); if(searchIndex!=-1){
// page fonud
                hit++; lruTime[searchIndex]=i;
                System.out.println("Page Hit");
            }
            else{
                System.out.println("Page Miss");
                miss++;
                // page not found
                int emptyindex=isEmpty(frames);
                if(emptyindex!=-1){
                // if frame is empty
                       frames[emptyindex]=page;
                        lruTime[emptyindex]=i;
                        //user lru algo to find replace location
                }
                else{

                    int minLocationIndex=lru(lruTime);
                    System.out.println("Replace "+ frames[minLocationIndex]);
                    frames[minLocationIndex]=page;
                    lruTime[minLocationIndex]=i;

                }

            }

        }


        System.out.println("Total page hit" + hit); System.out.println("Total Page miss
" + miss); System.out.println(Arrays.toString(frames));

    }
```

```
    public static int lru(int[] lruTime){
        int min = 9999; int index = -1;
        for(int i=0;i<lruTime.length;i++){

            if(min>lruTime[i]){
                min=lruTime[i]; index=i;
            }

        }

        return index;
    }

    public static int isEmpty(int[] frames){

        for(int i=0;i<frames.length;i++)
        {
            if(frames[i]==0){
                return i;
            }
        }

        return -1;
    }

    public static int isPresent(int[] frames, int search){

        for(int i=0;i<frames.length;i++){ if(frames[i]==search)
            return i;
        }

        return -1;
    }


}
```

OUTPUT:
Enter total no of frames
3
Enter total no of pages
8
Enter page value
1
Page Miss
Enter page value
0
Page Hit
Enter page value
2
Page Miss
Enter page value
0
Page Hit
Enter page value
3
Page Miss
Enter page value
1
Page Hit
Enter page value
2
Page Hit
Enter page value
0
Page Miss
Replace 3

Total page hit4
Total Page miss 4
[1, 2, 0]

2.Optimal Page Replacement:

```
package com.company;
import java.io.BufferedReader; import java.io.IOException; import
java.io.InputStreamReader;
 class OptimalReplacement {
    public static void main(String[] args) throws IOException
    {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        int frames, pointer = 0, hit = 0, fault = 0,ref_len; boolean isFull = false;
        int buffer[]; int reference[];
        int mem_layout[][];

        System.out.println("Please enter the number of Frames: "); frames =
Integer.parseInt(br.readLine());

        System.out.println("Please enter the length of the Reference string:"); ref_len =
Integer.parseInt(br.readLine());

        reference = new int[ref_len]; mem_layout = new int[ref_len][frames]; buffer = new
int[frames];
        for(int j = 0; j < frames; j++) buffer[j] = -1;

        System.out.println("Please enter the reference string: "); for(int i = 0; i <
ref_len; i++)
    {
        reference[i] = Integer.parseInt(br.readLine());
    }
        System.out.println();
        for(int i = 0; i < ref_len; i++)
        {
            int search = -1;
            for(int j = 0; j < frames; j++)
            {
                if(buffer[j] == reference[i])
                {
                    search = j; hit++; break;
                }
            }
            if(search == -1)
            {
                if(isFull)
                {
                    int index[] = new int[frames];
                    boolean index_flag[] = new boolean[frames];

                    for(int j = i + 1; j < ref_len; j++)
                    {
                        for(int k = 0; k < frames; k++)
                        {
                            if((reference[j] == buffer[k]) && (index_flag[k] == false))
                            {
                                index[k] = j; index_flag[k] = true; break;
                            }
                        }
                    }
                    int max = index[0]; pointer = 0;
                    if(max == 0)
                        max = 200;
                    for(int j = 0; j < frames; j++)
                    {
                        if(index[j] == 0)
                            index[j] = 200; if(index[j] > max)
                    {
                        max = index[j]; pointer = j;
                    }
                    }
                }
```

```
                buffer[pointer] = reference[i]; fault++;
                if(!isFull)
                {
                    pointer++;
                    if(pointer == frames)
                    {
                        pointer = 0; isFull = true;
                    }
                }
            }
            for(int j = 0; j < frames; j++) mem_layout[i][j] = buffer[j];
        }

        for(int i = 0; i < frames; i++)
        {
            for(int j = 0; j < ref_len; j++) System.out.printf("%3d ",mem_layout[j][i]);
System.out.println();
        }

        System.out.println("The number of Hits: " + hit); System.out.println("Hit Ratio: "
+ (float)((float)hit/ref_len));

        System.out.println("The number of Faults: " + fault);
    }

}
```

OUTPUT:

```
Please enter the number of Frames:
3
Please enter the length of the Reference string:
8
Please enter the reference string:
1
0
2
0
3
1
2
0

 1  1  1  1  1  1  1  0
-1  0  0  0  3  3  3  3
-1 -1  2  2  2  2  2  2
The number of Hits: 3
Hit Ratio: 0.375
The number of Faults: 5
```