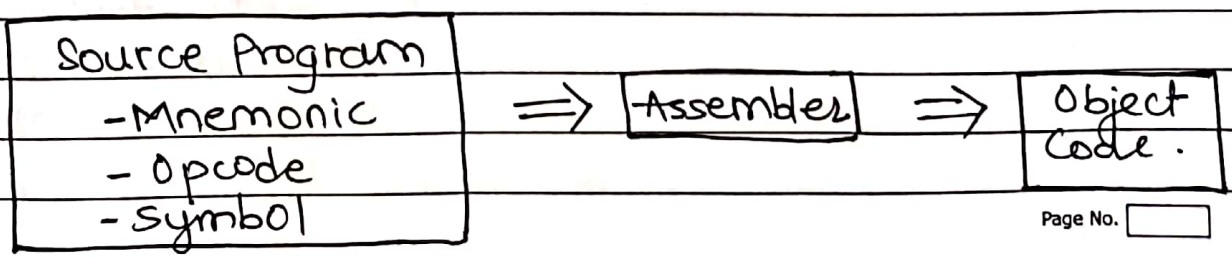
Name-Nirant Deore
TE Computer,
Div-A
Roll No-31

Saathi

Date __/__/__ SPOS Assignment No-1

- **Aim:-** To implement pass-1 assembler.
- **Problem statement:-** Design suitable data structures and implement pass-1 of a two-pass assembler for pseudo machine in Java using object oriented feature implementation of- few instructions for 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 each low-level machine.
- **Assembler:-** Assembly language is converted into executable machine by a utility program, referred to as assembler. The conversion process is referred to as assembly.

An assembler is a translator that translates an assembler program into a machine language program. Basically, assembler goes through program one line at a time and generates machine code for that instruction. Then assembler proceeds to the next instruction. In this way, the entire machine code program is created.



• Assembler Directives:- It directives are pseudo instructions. They will not be translated into machine instructions. They only provide instruct" direction | information to the assembly.
Basic assembler directives are -
i) START - Specify name & starting address for program.
ii) END - Indicate end of source program.
iii) EQU - Replace a number by a symbol.

• Main Data Structures:-
i) Operation Code Table (OPTAB)
ii) Location Counter (LOCCTR)
iii) Symbol Table (SYMTAB)

• One-Pass Assembler: A one-pass assembler passes over the source file exactly once, in the same pass collecting the labels, resolving future references and doing the actual references

• Data Structure for Assembler:-
i) Op-code table.
ii) Looked up for the translation of mnemonic code

Hashing is usually used once prepared, the table is not changed, efficient look up in desired

- Algorithm for Pass I Assembler:-

```
Begin
  if starting address is given
  LOCCTR = starting address;
  else
  LOCCTR = 0;
  while OPCODE != END do        ,or EOE
  begin
  read a line from the code
  if there is a label
  if there label is in SYMTB, ~~ere~~error
  else insert (label, LOCCTR) into SYMTAB
  search OPTAB for the opcode
  if found
      LOCCTR += N (N → 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
```

```
LTORG
  ADD  CREG='6'
STOP
A  DS 1
B  DS 1
END.
```

- Expected o/p => Symbol table:-

A	208
LOOP	208
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 feature.

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;
```

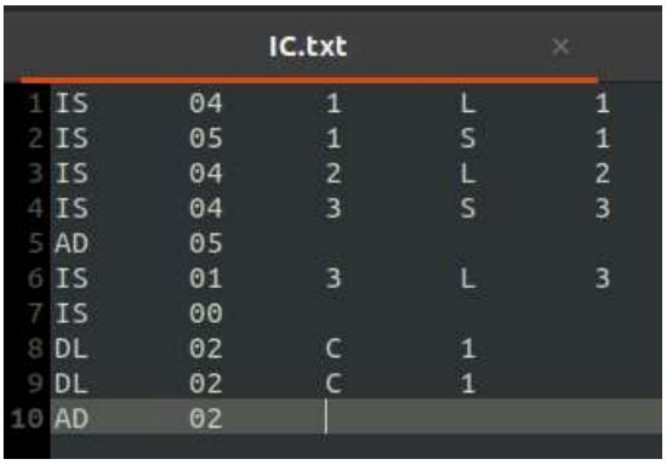
```
            int pooltabptr = 1;

            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";
```

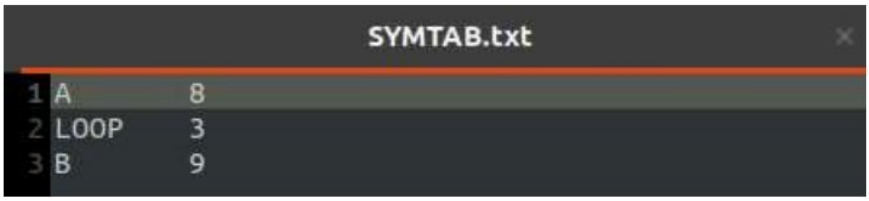
```
            FileWriter fw2 = new FileWriter(f2);
            BufferedWriter bw2 = new BufferedWriter(fw2);
            for (Map.Entry m : littab.entrySet()) {
                bw2.write(m.getKey() + "\t" + m.getValue() + "\n");
                System.out.println(m.getKey() + " " + m.getValue());
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

            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:

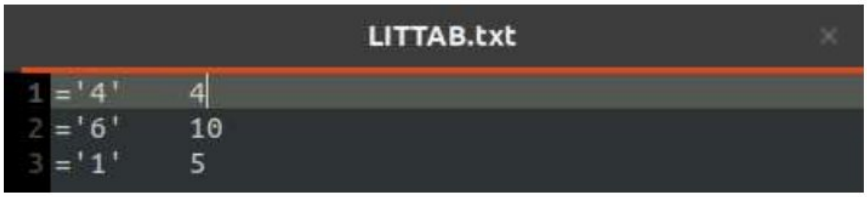
IC.txt



SYMTAB.txt



LITTAB.txt



Name- Nirant Deore
TE Computer
Div-A
Roll No-31

Saathi

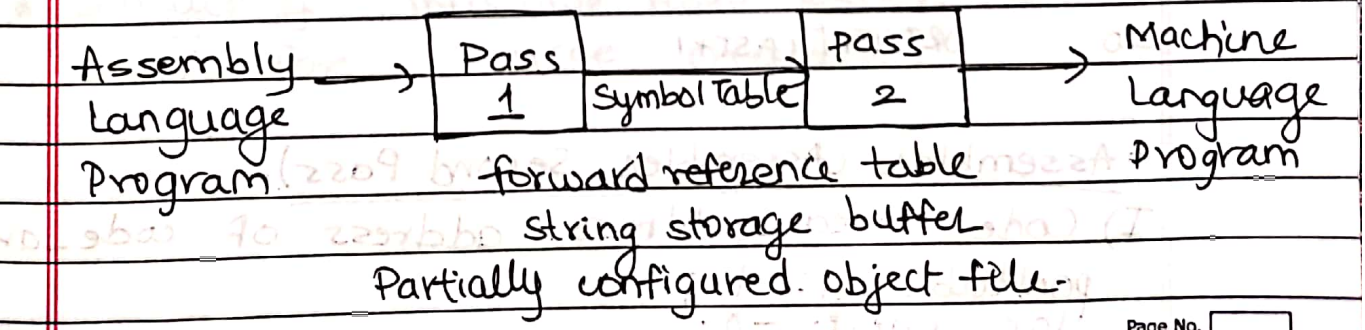
Date __/__/__ SPOS Assignment NO-2

- **Aim:-** To design data structure for Pass-2 assembler.

- **Problem Statement:-** Implement pass II of two pass assembler for pseudo-machine in Java using object oriented feature. The output of assignment I should be input for this 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 and 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 & literals are defined.
  Pass 2 - Object program is generated.

• **Data Structures:-**
i) Location Counter (LC)
ii) Op. Code translation table
iii) Symbol table.
iv) String storage buffer.
v) 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		LRORG		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				
25				219) +00	0	001
25			='1'			
20		ORIGIN	LAST+1			

- Assembler (Assembler Second Pass) :-

I) Code area_address := address of code_area
pooltab_ptr := 1 ;
loc_cntr := 0 ;

II) While next statement is not an END statement
a) clear machine - code - buffer;
b) IF an LTORG statement
i) Process literals in LITTAB [POOLTAB | pooltabptr1] LITTAB [POOLTAB [pooltab_ptr+1]] - 1 similal to processing of constants.
c) If a START or ORIGIN statement then
i) loc_cntr := value specified in operand file;
ii) size := 0;
d) If a declaration statement.
i) If a DC statement then.
Assemble the constant in machine_code, buffer.
ii) size := size of memory area required by DC/DS;
e) If an imperative statement.
i) Get operand address from SYMTAB or LITTAB.
ii) Assemble instruction in machine_code_buffer;
iii) size := size of instruction;
f) if size ≠ 0 then.
i) Move machine_code_begin contents to the address code_area_address + loc_cntr;
ii) loc_cntr := loc_cntr + size;
III) Processing of END statement
a) Perform steps 2(b) and 2(f)
b) write_code_area into O/P file.

- Conclusion:- Thus, we have generated machine code for source program.

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.

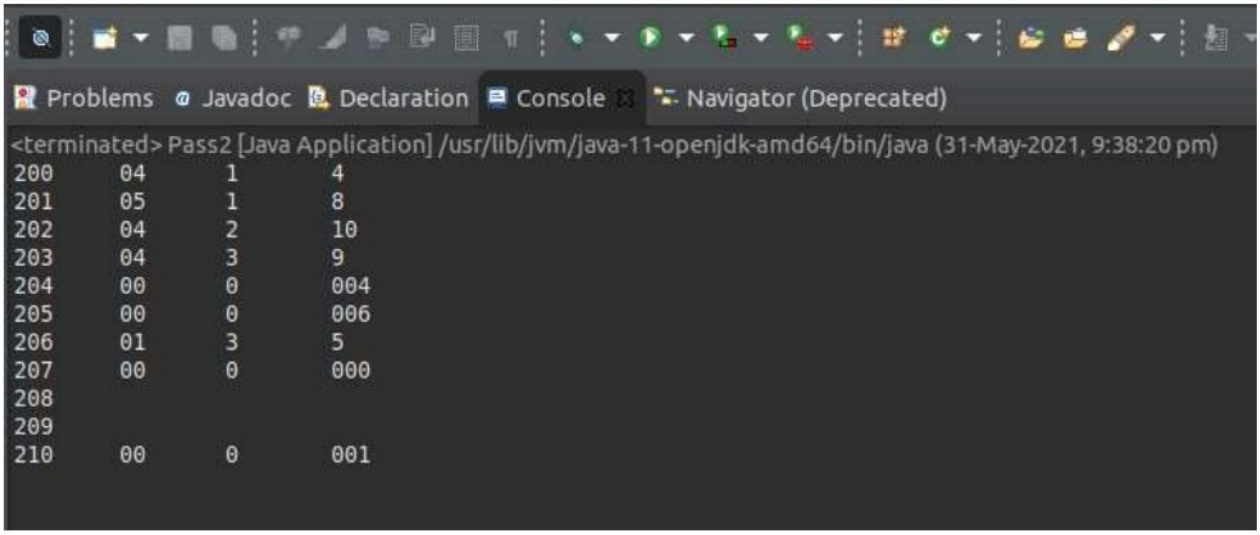
```
package com.company;

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 f1=new FileReader(f2);
            BufferedReader littab_file=new BufferedReader(f1);
            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) {
```

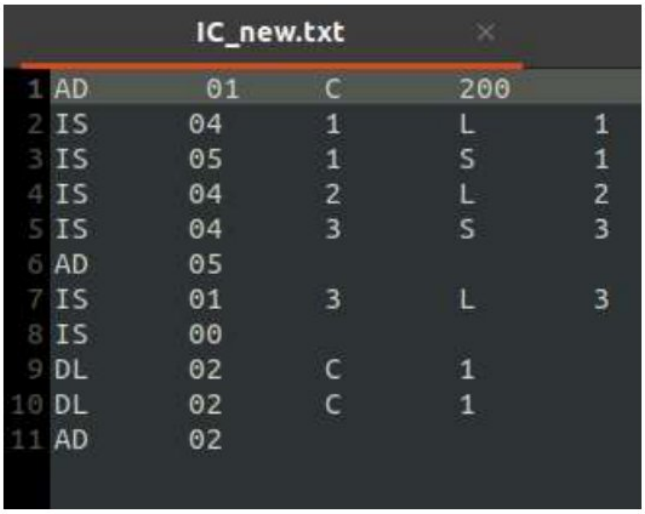
```
                        littab_file.readLine();
                    }
                    while (temp1 < temp2) {

                        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:

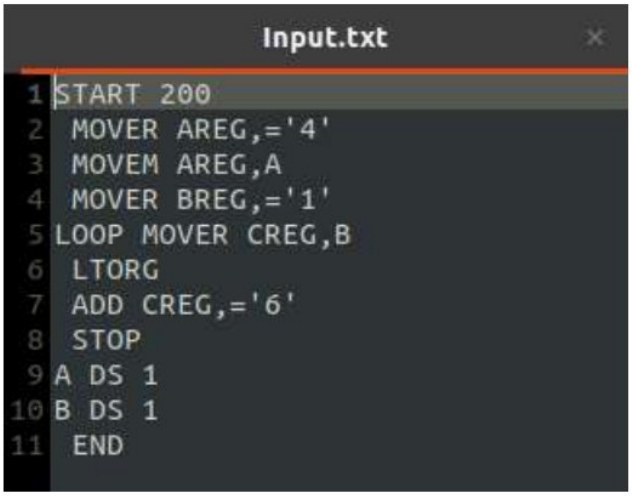
PASS- 2 OUTPUT:



IC_New.txt

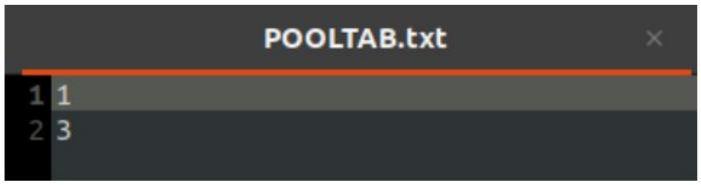


Input.txt

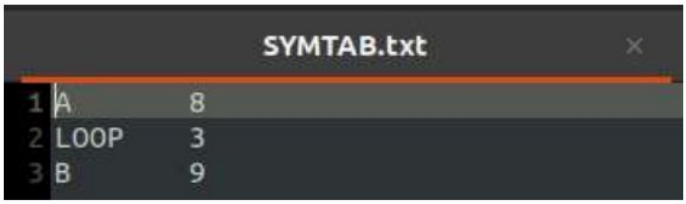


LITTAB.txt

POOLTAB.txt



SYMTAB.txt



Name - Nirant Deore
TE Computer
Div - A
Roll NO - 31

Date __/__/__

SPOS
Assignment NO - 3

- **Aim:-** To design data structures for microprocessor.
- **Problem statement:-** Design suitable data structure & implement pass-I of a two-pass-macro processor using OOP features in Java.
- **Theory:-** i) **Macro Processor:-** It is a program that reads a files & scans them for certain keywords when a keyword is found, it is replaced by some text. The keyword / text combination in called a macro.

ii) **Basic tasks perfored by macro processor:-**
a) Recognize macro definition.
b) Save the definition.
c) Recognize call.
d) Expanded calls & substitute arguments.

iii) **Macro definition parts:-**
a) Macro prototype statement.
b) Model statement
c) Preprocessor statement.

iv) **Data structure for macro definition processing:-**
a) Macro name table (MNT)
b) Parameter Name table (PNTAB)
c) Keywords parameter default table (KPDTAB)
d) Macro Definition table (MDT).

- **Algorithm:-**

```
begin {macro processor}
EXPANDING = FALSE
while OPCODE ≠ 'END' do
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 {PROCESS LINE}

procedure EXPAND
    begin
        EXPANDING := TRUE
    get first line of macro definition from DEFTAB
set up arguments from macro instructions in
ARGTAB. write macro invocation to expanded file
as comment while not end of macro definition
do.
begin
    GETLINE
    PROCESSLINE
    end {while}
```

EXPANDING := FALSE
end {EXPAND}
procedure GETLINE
begin.
if EXPANDING then
begin get next line of macro-definition from DEFTAB;
substitute arguments from ARGTAB for positional notation;
end {if}
else
read next line from input file.
end {GETLINE}

- Solved Example :-

Source			Expanded Source		
STRG	MACRO		.		
	STA	DATA1	.		
	STB	DATA2	.		
	STX	DATA3	STA	DATA1	
	MEND		STB	DATA2	
:			STX	DATA3	
STRG			:		
			STA	DATA1	
STRG			STB	DATA2	
			STX	DATA3	

procedure GETLINE
begin
if EXPANDING then
begin get next line of macro definition from

Page No.

DEFTAB;
substitute arguments from ARGTAB for positional notation;
end {if}
else
read next line from input file.
end {GETLINE}

- Solved Example:-

Source		
STRG	MACRO	
	STA	DATA1
	STB	DATA2
	STX	DATA3
	MEND	
.		
STRG		
.		
STRG		

Expanded Source		
.		
.		
.		
	STA	DATA1
	STB	DATA2
	STX	DATA3
.		
.		
	STA	DATA1
	STB	DATA2
	STX	DATA3

Source		
STRG	MACRO	&a1, &a2, &a3
	STA	&a1
	STB	&a2
	STX	&a3
	MEND	
.		
STRG	DATA1, DATA2, DATA3	
.		
STRG	DATA4, DATA5, DATA6	

Expanded Source		
.		
.		
	STA	DATA1
	STB	DATA2
	STX	DATA3
.		
.		
	STA	DATA4
	STB	DATA5
	STX	DATA6

- Conclusion:- Thus, pass I of macro processor is implemented and MNT, MDT & ALA file is generated.

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

```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-Nirant Deore
TE Computer
Div-A
Roll NO-31

Date __/__/__ SPOS Assignment NO-4

- **Aim:-** Design of a MACRO PASS-2

- **Problem Statement:-** Write a Java program for pass II of a two pass macro-processor, The output of assignment is (MNT, MDT and File without any macro definitions) should be input for this assignment.

- **Theory:-** i) Macro Processor:- It is a program that reads a File and scans them for certain keyword when a keyboard is found, it is replaced by same text. The keyword /text combination is called a Macro.

ii) Basic takes performed by Macro Processor:-
a) Recognize macro definition
b) Save the definition
c) Recognize call.
d) Expanded calls & substitute arguments
e)

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

- Steps:- i) Read the input data received from PassI
ii) Examine each operation code for finding respect-ive entity in the MNT.
iii) IF name of micro is encountered then
a) A pointer is set to MNT entry where name of macro is found. This pointer is called macro

definition table pointer (MOTP)

b) Prepare argument list array containing a table of dummy arguments.

c) Increase the value of MOTP by value one.

d) Read next line from MDT.

e) Substitute the values from arguments list of the macro for dummy arguments.

f) IF mind pseudo code is found then next source of input data is ~~used~~ read.

g) Else expand data input.

iv) When macro name isn't found then create expanded data file.

v) If end pseudo code is encountered then feed the expanding source file to assemblel.

vi) Else read next source of data input.

- Conclusion:- Thus, pass II of macroprocessor is implemented and ALA file is generated.

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-Nirant Deore
TE Computer
Div-A
Roll No-31

Saathi

Date __/__/__ SPOS Assignment NO-5

- Aim:- Design ~~for~~ 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, LEX160

- Theory:- Lexical analysis is a tool for generating scanners. Scanner are programs that recognize lexical ~~program~~ patterns in text is matched regular expression may have an associated action. Lex turns the users expressions & actions into 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 in Lex:- It can be divide in 3 steps:-

i) Specify the pattern associated actions in a form that lex can understand.
ii) Run lex over the file to generate C code for the scanner.
iii) Compile & link the C code to procedure the executable scanner.

- **Lex program in 3 sections:-**

i) Global C & lex declaration.
ii) Patterns
iii) Supplement C functions.
The section are delimited by '%%'.

A character class define a single character. Two operators supposed in a character class are hyphen ("-") and circumflex ("^")

```
...definitions
%%
...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 rules sections. If we don't specify any rules then the default action is to match everything & copy is to output.

- **Conclusion:-** Thus, we have studied lexical analyzer & implemented application for it to generate tokens.

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 - Nirant Deore
TE Computer
Div - A
Roll No - 31

Date __/__/__ SPOS

Assignment No - 6

- **Aim:-** To design lex program to count no. of words lines & character of given input file.

- **Problem Statement:-** Write a program using lex specifications to implement lexical analysers phase of compiler to count no. of words, lines & characters of given input file.

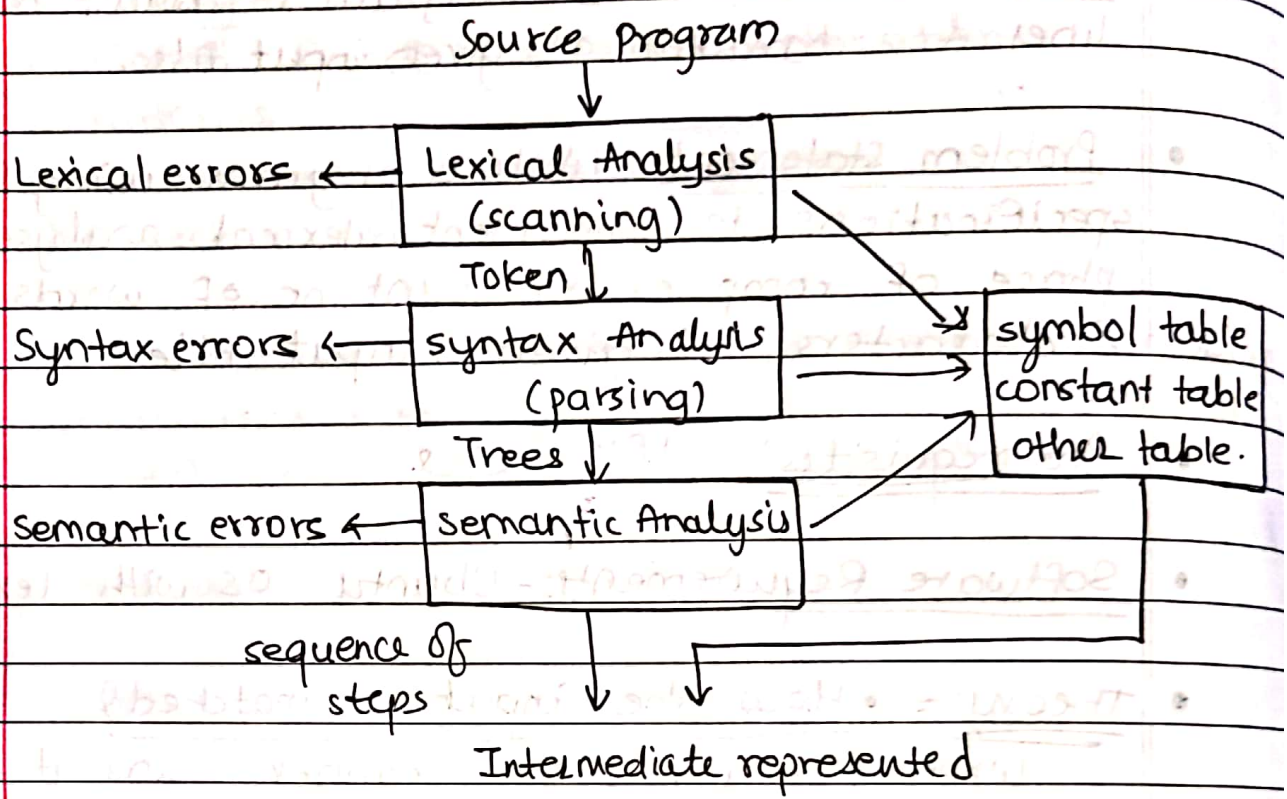
- **Pre-requisites:-** LEX Basics.

- **Software Requirements:-** Ubuntu OS with Lextool (flex)

- **Theory:-**

  • **How the input is matched?**

  When the generated scanner, runs it analyses its 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 choosen. Once the match is made available in yytext & its length in yyleng. If no. match is found then default rule is executed. the next character in the input is considered matched & copied to standard output.



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

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- Nirant Deore
TE Computer
DW-A
Roll No-81

Saathi

SPOS

Date __/__/__ Assignment NO-07

- **Aim:-** Design and Lex & Yacc program in 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 & syntax of variable declaration in Java.

- **Pre-requisites:-** 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 OS. developed by Stephen C. Johnson. It is a look ahead left-to-right parsor generator, the part of a compiler that tries to make syntatic sense of source code Based on analytic grammar written in a notation similar job is to analyze the structure of the input stream, & operate of the 'big picture'.

•**Structure of a Yacc file:-**

```
... definitions ...
%%
... rules ...
%%
... code ...
```

Definitions as with lex, all code between %{ & %} is copied to beginning of resulting C file. Rules as

Page No.

with lex, a number of combinations of pattern and action. Input to Yacc is divided into three sections. The definition section of consist of token declaration & c code bracketed by "%{ and %}". The BNF grammar is placed in rules section. IF can be used to progress context free lang.

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 & 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 yy.c y.tab.c -ll
./a.out-.

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

Lexical Analyser for Yacc:- The user must supply a lexical analyser to read input stream & communicate tokens. If there is a value associated with that token, it should be assigned to -ra.yylval.

The resultant potion of lexical analyzer might look like -

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

say name of token in 'DIGIT'

- Comparing sentence types:- Sentences gives structure to that language. They come in four types - simple, compound, complex & compound-complex.

- Application:- ~~You~~ Yacc are used to generate parsers, which in an integral part of compiler.

- Conclusion:- Hence, we have studied lexical analyser syntax & implemented ~~&~~ lex & Yacc application for syntax analyser to validate given inflex expression.

Name-Nirant Deore
TE Computer
Div-A
Roll No-31.

Date __/__/__ SPOS Assignment 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) is a standard parser generator for the UNIX o.s. An open source program yacc generates code for the parse in the C programming language. The acronym is usually rendered in lowercase but is occasionally seen as YACC or Yacc. This original version of yacc was written Stephen Johnson at America telephone & telegraph (AT&T). Version of Yacc have since been written for use with Ada. Java & several other less well-known programming languages.

Yacc file format:-

```
%{
  C declarations
%}
yacc declarations
%%
```

Grammar rules:-

%. %

Additional C code (/* user subroutines */)

- Algorithms:-

i) Include header files.
ii) Declare rules
Return Datatypes
Return Comma
Return ;
Return ID
iii) End

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

Name-Nirant Deore
TE Computer
Div-A
Roll No-31

Saathi

Date __/__/__ SPOS Assignment No-9

- **Aim:-** To implement Yacc specification with simple & compound sentences.

- **Problem statement:-** Write a program using Yacc specifications to implement syntax analysis phase of compile to recognize simple & compound sentences.

- **Theory:-** • **Syntax Analyzer:-** Syntax analysis or parsing is the second phase i.e. after lexical analysis. It checks the syntactical structure of given input- i.e. whether the given input is correct syntax or not. It does so by building the data structure, called a parse tree or syntax tree.

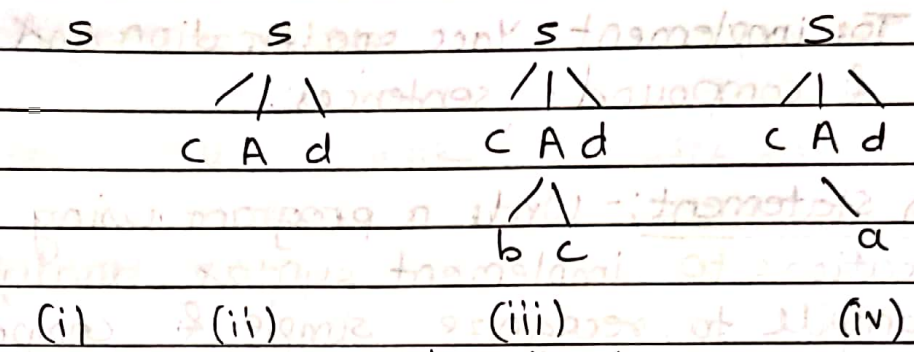
e.g. Suppose, production rules for the grammar of a language are -

$S \rightarrow cAd$

$A \rightarrow bc / a$

& the input string is "cad"

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



- **Algorithm:-**

Begin
include header file
define coompound = 0
define rules
define yywrap()
take input
yylex();
check if (compound == 1)
print ('sentence is compound')
else
print ('sentence is simple')
return
End.

- **Conclusion:-** Thus, we have implemented syntax analysis phase of compiler to recognize simple & compound statement given in input file.

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 - Nirant Deore
TE Computer
Div - A
Roll No - 31

SPOS

Date __/__/__ Assignment NO-10

- **Aim:-** Implement job scheduling algorithm i) FCFS ii) SIF iii) Priority iv) RR

- **Problem statement:-** Write a Java program (using OOP features) to implement following scheduling algorithm FCFS, SJF, Priority, RR.

- **Theory:-** i) **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 ~~que~~ queue. When a process request the CPU it is added at the tail of the job queue. The CPU is allocated to the process at the head of queue.

- **Algorithm:-** i) Input the process along with their burst time (bt)
ii) Find waiting time (wt) for all processes.
iii) As first process that comes need not to wait so waiting time for process will be 0 i.e. (wt) = 0.
iv) Find waiting for all other process i.e. for process i → wt [i] = bt [i-1] + wt [i-1]
v) Find turnaround time = waiting_time + burst_time for all processes.
vi) Find average waiting time = $\frac{\text{total\_waiting\_time}}{\text{no-of-processes.}}$
vii) Similarly, find average turnaround time = total_turn_around_time / no of processes.

ii) **Shortcut job first:-** This algorithm associates with it the length of next CPU burst. When the

Page No.

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 known as CPU burst of a job.

- Algorithm:- i) Sort all the processes in increased order according to burst time.
ii) Then simply, apply FCFS.

iii) Priority Scheduling:- Priority scheduling is a non-primitive algorithm. It is most common scheduling algorithm in batch systems. Process with same priority are executed on FCFS basics. Priority can be decided based on memory requirement, time requirements or any other resource requirements.

- Algorithm:- i) First input the process with their burst time & priority.
ii) Start the process, burst time & priority & according to priority.
iii) Now simply apply FCFS algorithm.

iv) Round Robin Scheduling:- RR is a CPU scheduling algorithm where each process is assigned a fixed time slot in a queue cyclic way. It is primitive as process are assigned CPU only for a fixed slice of time at most. Each process is provided a fix time to execute is called a quantum. Once a process is executed for given time period

is ~~pre.~~ & other process executes for a given period of time.

- Conclusion:- Thus, we have implemented job scheduling algorithm FCFS, RR, SJF & priority.

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:

```java
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:

```
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 - Nirant Deore
TE Computer
Div - A
Roll NO-31

Date __/__/__ SPOS Assignment NO-11

- **Aim:-** Bankers algorithm for deadlock detection & avoidance.

- **Problem statement:-** Write a Java 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 make an "s-state" check to 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 numbers of resources types-
Available:
- It is a 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 2D array of size 'n * m' 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 2d array of size 'n*m' that defines the numbers of resources of each type currently allocated to each process.
- Allocation [i,j] = k means a process Pi is currently allocated 'k' instances of resource type Rj.
- Need:- - It is a 2-d array of size 'n*m' that indicates the remaining source need of each process
- Need [i,j] = Max [i,j] - Allocation [i,j]

Safety Algorithm:- i) Let work & finish be vectors of length 'm' & 'n' respectively.
Initialize work = Available.
finish [i] = False ; for i = 1,2,3,4...n.
ii) Finish can i such that both.
a) finish [i] = false.
b) Need i <= work
• if No such i exists goto step (iv)
iii) work = work + Allocation [i]
finish [i] = true
goto step (ii)
iv) Finish [i] = true for all i
then system is in safe state.

- Conclusion:- Thus, we have studied implemented bankers algorithm for deadlock avoidance.

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 - Nirant Deore
TE Computer
Div - A
Roll No - 31

Saathi

Date __/__/__ SPOS

Assignment 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:-** When a program in user mode requires access to RAM or a hardware resource, it must ask the kernel to provide access to that resource. This is done via something called a system call.
When program makes a system call, the mode is switched from user mode to kernel mode. This is called a content switch.

- **Kernel Mode:-** - when a CPU is in kernel mode, the code being executed can access any memory address, and any hardware resource. In user mode, if any program crushes only that particular program is halted.
That means system call will be in safe state even if a program in user mode crashes. Hence, most program is an os runs in user mode.

- **Unix system calls:-** PS command - This command is used to provide information about the currently running process, including their process identification number (PID)

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

- Join command :- It is a command line utility for joining lines of two files on a command field

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

- Wait() command :- A call to wait() blocks the calling process until one of its child processes exists or a signal is received. After child process terminate parent continues its execution after wait system call instructions.

- Conclusion :- Thus, the process system call program is implement & studied various 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 - Nizant Deore
TE computer
Div-A
Roll NO-31

- **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:-**
  - **Android OS:-** Android is a mobile OS developed by Google, based on modified version of linux kernel & other open source software & designed primarily for touch screen mobile devices such as smartphone & tablets. These applications are more comfor-table and advanced for users.
  - **Tizen OS:-** Tizen is a mobile OS developed by samsung that runs on a wide range of samsung devices, including smartphones, tablets, in vehicle information (IVI) devices. As of 2017 Tizen is second largest smartwatch based on platform behind watchOS & ahead of Android wear.

Process scheduling in Android and Tizen OS:-

- **Normal scheduling:-** Android is based on linux & uses the linux kernel's scheduling mechanisms for determining scheduling policies. The linux time sliced scheduling policy combines static & dynamic properties.

- Real time scheduling:- The standard linux used kernel provides two real time scheduling policies, SCHED-FIFO & SCHED-RR. The main real time policy is SCHED-FIFO-It implements FIFO algorithm Non-real time task use the SCHED_NORMAL scheduling policy.

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

- Priority based Pre-emptive Task scheduling for Android O.S.:-
  This 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 & Tizen O.S.

Name - Nirant Deore
TE Computer
Div - A
Roll No - 31

Date __/__/__ SPOS Assignment NO-14

- **Aim:-** Implementing page replacement algorithm. i) LRU ii) optimal.

- **Problem Statement:-** To write a Java program (using OOP) to implement LRU & optimal algo. for page replacement.

- **Prerequisite:-** i) Explain concept of virtual memory. ii) Define page replacement algorithm : LRU & optimal iii) Explain address translation in paging system. iv) Explain Belady's Anomaly.

- **Theory:-** i) **LRU Page Replacement:-** The main differenc bet^n FIFO & 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 the time recent past as an approximation of future then we will replace the page that has not been used for longest period of time. This approach is called as least-recently used algorithm Now, consider reference string 7, 0, 1, 2, 0, 3, 4, 2, 3, 0, 3 with three memory frames or blocks. The first three reference cases pages fault that fill empty frames.

- **Algorithm:-** 1. Start traversing the pages.
i) 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.

Page No.

request are processed.
b) Simultaneously maintain the recent occured index of each page in a map called indexes
c) Increment page fault.
ii) else
If current page is present in set, do nothing
else
a) find the page in set that was level 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) Increment page faults.
d) Update index of current page.
2) Return Page faults.

ii) Optimal Replacement :- The algorithm has lowest page fault rate of all algorithm. This algorithm states that : Replace the page which will not be used for longest period of time.
Often called Balady's Min Bask idea : Replace the page that willnot be offered for the longest time.
Impossible to implement.

Consider the following reference string:-

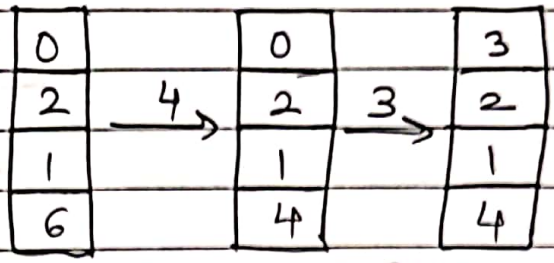
0, 2, 1, 6, 4, 0, 1, 0

x x x x

3, 1, 2, 1

x x

compulsory misses



Fault Rate = 6/12 = 0.50

- Algorithm:-

i) Start the process.
ii) Declare the size.
iii) Get number of pages to be inserted.
iv) Get the value.
v) Compare counter label & stack.
vi) Select optimal page by counter value.
vii) Stack them according the selection.
viii) Print pages with fault pages.
ix) Stop the process.

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

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
```