Mayur Gorane
TE(A) 48

Experiment No-1

Aim : To implement pass-1 Assembler

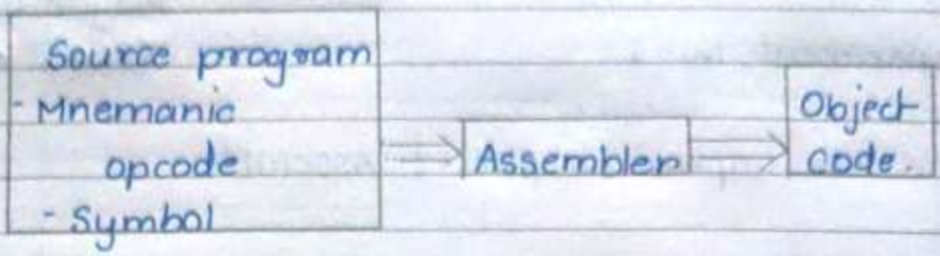
Problem Statement : Design suitable data structures and implement pass-I of a two-pass assembler for pseudo machine in Java using object oriented feature implementation of few instuclion from each category & few assembler directions.

Theory :

Assembly Language : Its a low level language for a computer or other programable device. Each assembly language is specific to particular computer architecture assembly language a memoric 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 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 & generates machine code for that instruction. Then assembler proceed to next instuction. In this way, entire machine code program is created.



Assembler directives : It directives are pseudo instructions. They will not be translated into machine instructions They will not only provide instructions / direction / information to assembly.

Basic assembler directives are :

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

Main data structures :

1. Operation code table (OPTAB)
2. Location counter (LOCCTR)
3. Symbol Table (SYMTAB)

One-pass assembler :

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

Data Strusture for assembler :

1. Op-code table.
2. Looked up for translation of mnemowic code.

Hashing is usually used once prepared, table is not changed, efficient loop up is desired.

Algorithm for pass-I assembler.

```
Begin
    if starting address is given
    LOCCTR = starting address,
    else
    LOCCTR=0;
    while OPCODE ! = END do , on EOF
     begin
     read a line from code
     if there is a label
     if these label is in SYMTAB, error
     else insert (label, LOCCTR) into SYMTAB
     search OPTAP for 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 DS1
 B DS1
 END
```

Expected o/p : Symbol Table

A	208
LOOP	203
B	209

Intermediate key:

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

Conclusion : Thus we have implemented PASS-1 assembler using object oriented features.

Assignment No. 01 [Pass 1 Assembler]

Problem Satement: Design suitable data structures and implement pass-I of a twopass 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:

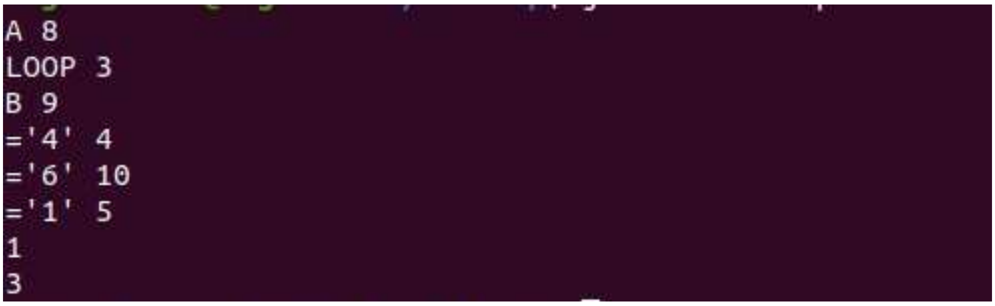
```
import
java.io.BufferedReader;
import java.io.*; import
java.io.IOException;
import java.util.*;

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

                BufferedReader br = null;
                FileReader fr = null;

                FileWriter fw = null;
                BufferedWriter bw = null;

                try {

                        String inputfilename = "/home/mayur-r/Desktop/Input.txt";
                        fr = new FileReader(inputfilename); br = new
                        BufferedReader(fr);

                        String OUTPUTFILENAME = "/home/mayur-r/Desktop/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++;

                                            }
```

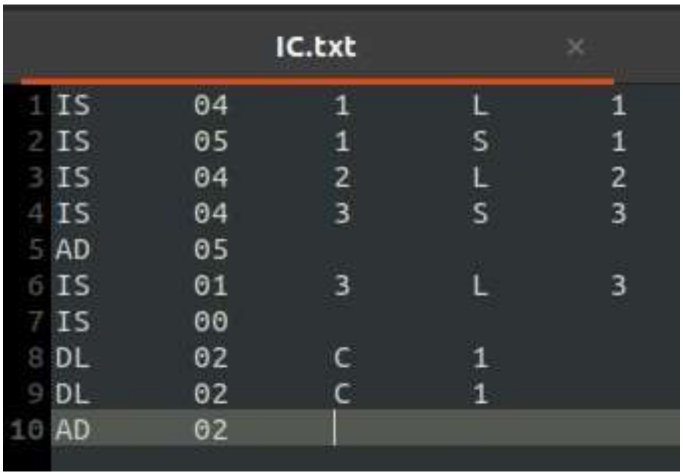
```
                    }

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

                    if (mind_the_LC == 0)
                            locptr++;

            }

            String f1 = "/home/mayur-r/Desktop/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 = "/home/mayur-r/Desktop/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 = "/home/mayur-r/Desktop/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:



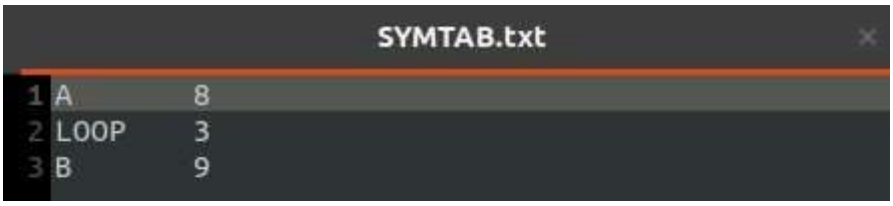
```
A 8
LOOP 3
B 9
='4' 4
='6' 10
='1' 5
1
3
```

IC.txt



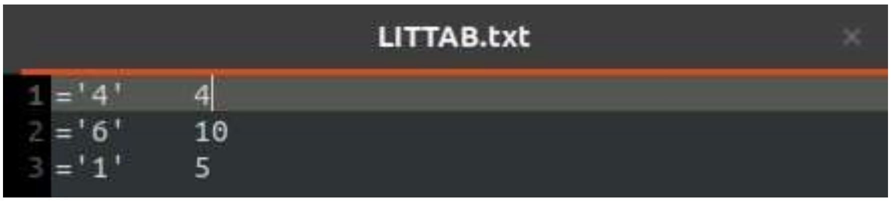
```
IS      04      1       L       1
IS      05      1       S       1
IS      04      2       L       2
IS      04      3       S       3
AD      05
IS      01      3       L       3
IS      00
DL      02      C       1
DL      02      C       1
AD      02
```

SYMTAB.txt



```
A       8
LOOP    3
B       9
```

LITTAB.txt



```
='4'    4
='6'    10
='1'    5
```

POOLTAB.txt

Mayur Gorane

Experiment no: 2 TE(A) 48

Aim: To design data steucture for pass-2 assembler

problem statement: Implement pass-II of 2 pass assembler for pseudo-machine in Java using object oriented features The output of assignment I should be input for this assignment

Theory:
Two pass assembler: Two pass assembler perform two passes over the source program. In first pass it reads entire source program all labels are collected & placed in symbol table. In the second pass, insteuctions are again, read and 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 Steuctures:
1. Location counter (LC)
2. Op code teanslation table
3. Symbol table
4. steing storage buffer.
5. Forward refexnce table.

Assembly language program → Pass 1 $\xrightarrow[\text{table}]{\text{symbol}}$ Pass 2 → Machine language program.

Forward reference table
Steing storage buffer
Partially configured object file.

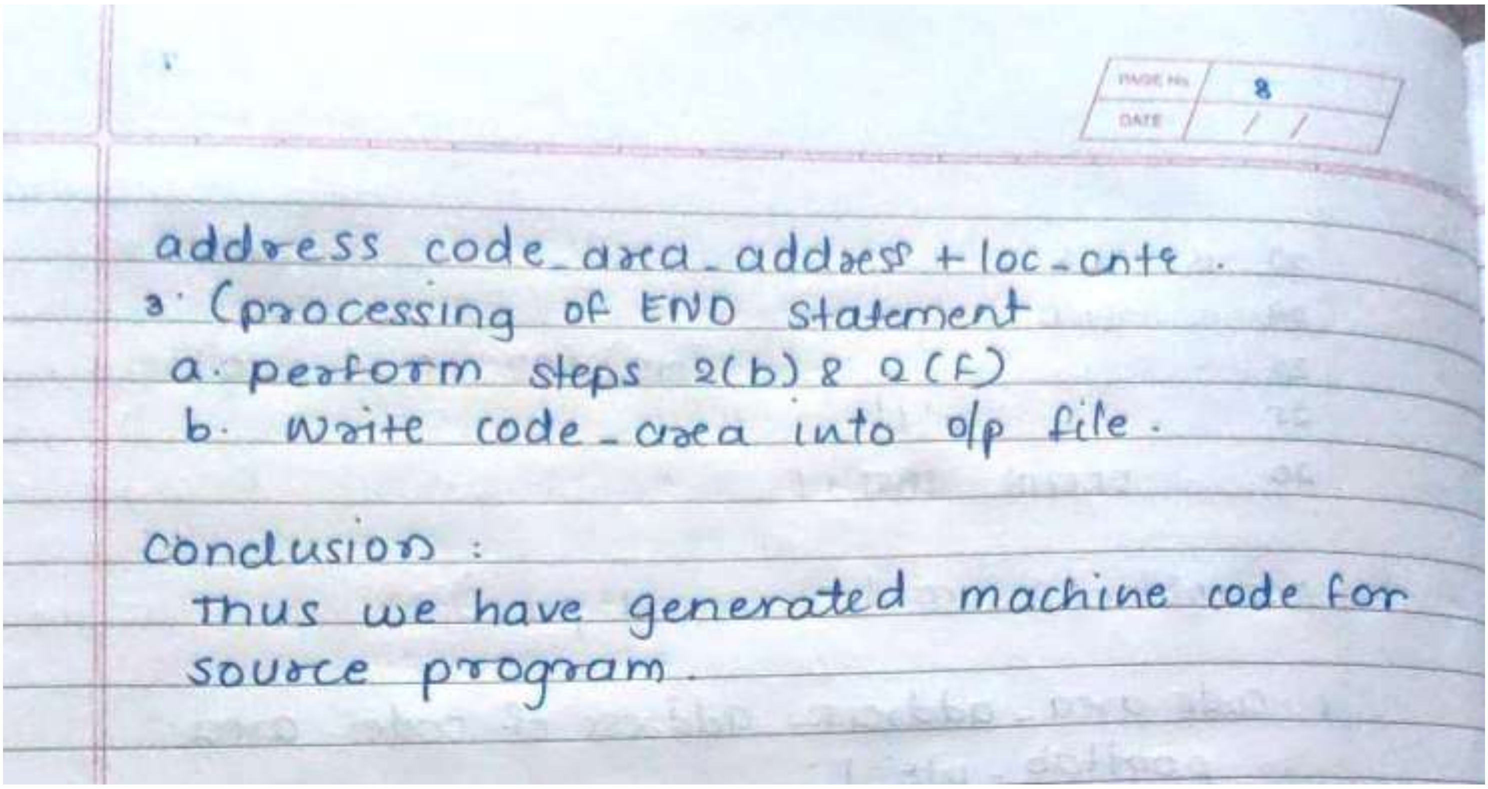
Solved example:

```
1    START       200            200) +04   1   211
2    MOVER       AREG='5'       201) +05   1   217
3    MOVEM       AREG, A        202) +04   1   218
4 LOOP MOVER     AREG, A        203) +05   3   218
5    MOVER       CREG, B        204) +01   3   212
6    ADD         CREG,='1'
7    ...                        210) +07   6   214
12   BC          ANY, NEXT
13   LTORG                      211) +00   0   005
                 ='5'           212) +00   0   001
                 ='1'

14                              214) +02   1   219
15 NEXT SUB      AREG='1'       215) +07   1   202
16        BC     LT, BACK.      216) +00   0   000
17 LAST   STOP
18        ORIGIN LOOP+2         204) +03   3   218
19        MULT   CREG.B
20        ORIGIN LAST+1         217)
21 A      DS     1
22.BACK   EQU    LOOP           218)
```

```
23  B    DS   1
24       END
25                       219) +00   0     001
25             ='1'
20       ORIGIN  LAST +1.
```

Assembler (Assembler second pass)

```
1. Code area_addrers:= addsess of code_area
     pooltab_pts:=1;
     loc_cnts:= 0;
2. while next statement is not an END Statement
   a) clear machine_code_buffer;
   b) If an LTORG stat-ement
      1. Process literals in LITTAB [POOLTAB[pooltab_ptr]]
      LITTAB [POOLTAB [pooltab_pts +1]]-1 similar to
         processing of constants
   c) If a START or ORIGIN Statement then
      1. loc-cnts:= value specified in operand file;
      2. size := 0;
   d) If a declaration statement
      1. If a DC statement then
      Assemble constant in machine_code, buffer
      ii. size := size of memory area required by DC/DS;
   e) If an imperative statement
      1. Get operand address from SYMTAB or LITTAB
      ii. Assemble instruction in machine_code_buffer;
      iii. Size := size of instruction;
   f) If size ≠ 0 then.
      1. Move machine_code_buffer contents to buffer
```



address code_area_address + loc_cntr .

3· (processing of END statement

a. perform steps 2(b) & 2(F)

b. write code_area into o/p file.

Conclusion :

Thus we have generated machine code for source program.

Mayur Gorane
TE(A) 48

Experiment no : 3

Aim: To design data steucture for microprocessor

Problem Statement : Design suitable data steucture and implement pass I of a two pass macro processor using oop fealures in Java.

Theory :

1. Macro processor.

Its a program that reads a files & scans them for certain keywoods when a keyword is found its replaced by some text the keyword / text combination is called macro.

2. Basic tasks performed by macro processor :

a. Recognize macro definition
b. Save definition
c. Recognize call
d. Expanded calls & substitute arguments.

3. Macro definition part :

a. Macro prototype statement
b. Model statement
c. Preprocessor statement.

4. Data steucture 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 {PROCESSLINE}
Procedure EXPAND
  begin
      EXPANDING : = TRUE
```

get first line of macro definition from DEFTAB set up arguments from macro instructions in ARGTAS. 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 EXPADING then
    begin got next line of macro definition from DEFTAB,
    positional notation ,
    end {if}
    else
     read next file from input file .
    end {GETLINE}
```

Solved example :

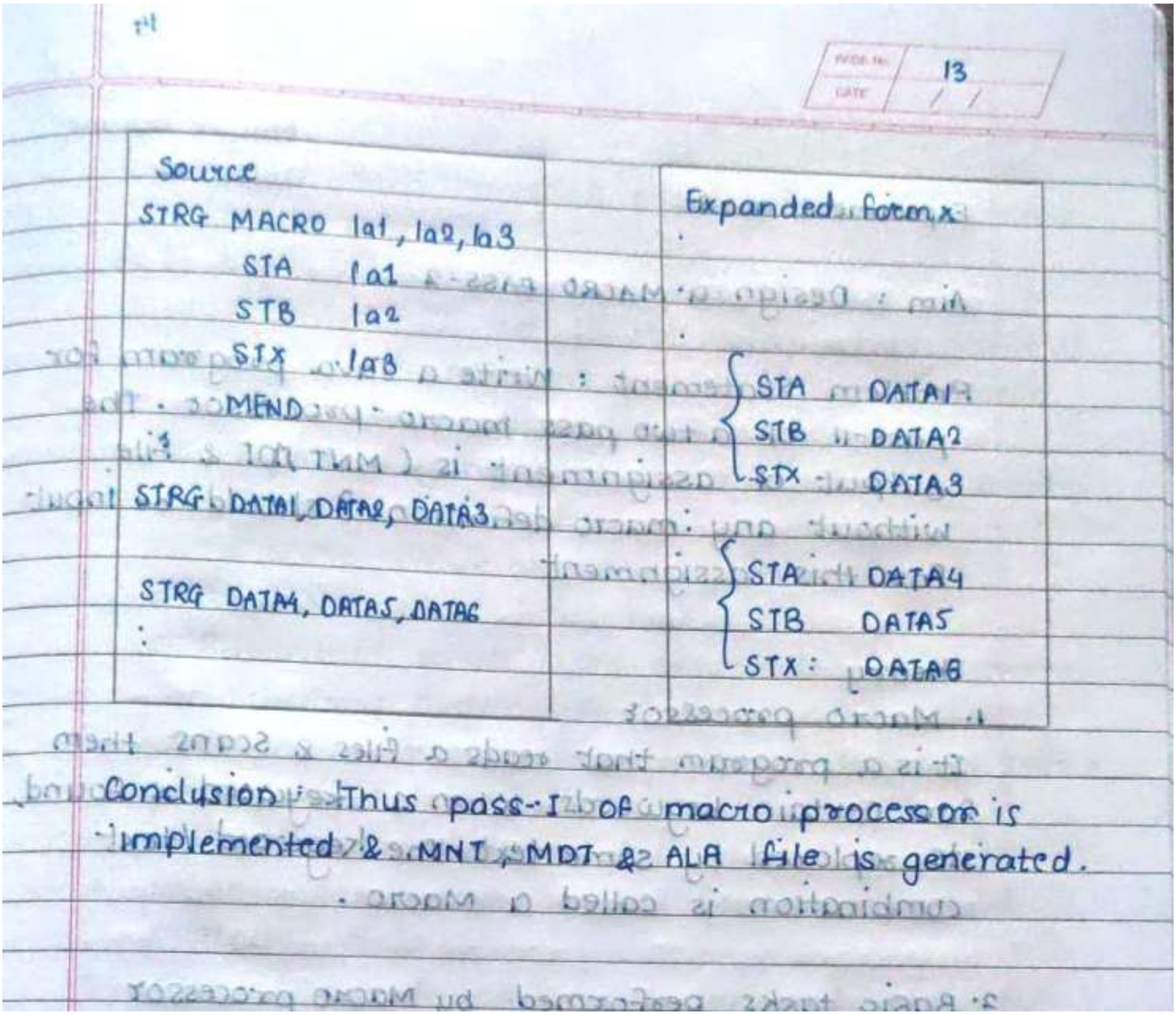
Source			Expanded Source	
STRG	MACRO			
	STA	DATA1		
	STB	DATA2	.	
	STX	DATA3	STA	DATA1
	MEND	:	STB	DATA2
.			STX	DATA3
STRG			:	
			STA	DATA1
			STB	DATA2
STRG			STX	DATA3.

```
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
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
```



PAGE No. 13
DATE / /

Source		Expanded form :
STRG MACRO la1, la2, la3		.
STA la1		
STB la2		.
STX la3		STA DATA1
MEND		STB DATA2
		STX DATA3
STRG DATA1, DATA2, DATA3		
		STA DATA4
STRG DATA4, DATA5, DATA6		STB DATA5
:		STX DATA6

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

Name: Mayur Vijay Gorane **DivRoll**: TE-A-48

Assignment No. 03 [PASS-1 Macroprocessor]

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

1. Pass 1 Macro Code:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashMap;

public class macroPass1 {
      public static void main(String[] Args) throws IOException{
            BufferedReader b1 = new BufferedReader(new 
FileReader("input.txt"));
            FileWriter f1 = new FileWriter("intermediate.txt");
            FileWriter f2 = new FileWriter("mnt.txt");
            FileWriter f3 = new FileWriter("mdt.txt");
            FileWriter f4 = new FileWriter("kpdt.txt");
            HashMap<String,Integer> pntab=new HashMap<String,Integer>();
            String s;
            int paramNo=1,mdtp=1,flag=0,pp=0,kp=0,kpdtp=0;
            while((s=b1.readLine())!=null){
                  String word[]=s.split("\\s");       //separate by space
                  if(word[0].compareToIgnoreCase("MACRO")==0){
                        flag=1;
                        if(word.length<=2){

      f2.write(word[1]+"\t"+pp+"\t"+kp+"\t"+mdtp+"\t"+(kp==0?kpdtp:(kpdtp+1
))+"\n");
                              continue;
                        }
                        String params[]=word[2].split(",");
                        for(int i=0;i<params.length;i++){
                              if(params[i].contains("=")){
                                    kp++;
                                    String 
keywordParam[]=params[i].split("=");

      pntab.put(keywordParam[0].substring(1,keywordParam[0].length()),param
No++);
                                    if(keywordParam.length==2)

      f4.write(keywordParam[0].substring(1,keywordParam[0].length())+"\t"+k
eywordParam[1]+"\n");
                                    else

      f4.write(keywordParam[0].substring(1,keywordParam[0].length())+"\t"+"
-"+"\n");

                              }
                              else{

      pntab.put(params[i].substring(1,params[i].length()),paramNo++);
                                    pp++;
                              }
                        }

      f2.write(word[1]+"\t"+pp+"\t"+kp+"\t"+mdtp+"\t"+(kp==0?kpdtp:(kpdtp+1
))+"\n");
```

```
                    kpdtp+=kp;
                }
                else if(word[0].compareToIgnoreCase("MEND")==0){
                    f3.write(s+'\n');
                    flag=pp=kp=0;
                    mdtp++;
                    paramNo=1;
                    pntab.clear();
                }
                else if(flag==1){
                    for(int i=0;i<s.length();i++){
                        if(s.charAt(i)=='&'){
                            i++;
                            String temp="";
                            while(!(s.charAt(i)=='
'||s.charAt(i)==',')){
                                temp+=s.charAt(i++);
                                if(i==s.length())
                                    break;

                            }
                            i--;
                            f3.write("#"+pntab.get(temp));
                        }
                        else
                            f3.write(s.charAt(i));
                    }
                    f3.write("\n");
                    mdtp++;
                }
                else{
                    f1.write(s+'\n');
                }
            }
            b1.close();
            f1.close();
            f2.close();
            f3.close();
            f4.close();
    }
}
```

/*
OUTPUT:

```
mayur-r@Mayur-HP:~/SPOSL$ javac macroPass1.java
mayur-r@Mayur-HP:~/SPOSL$  java macroPass1
mayur-r@Mayur-HP:~/SPOSL$  cat intermediate.txt
M1 10,20,&b=CREG
M2 100,200,&u=AREG,&v=BREG

mayur-r@Mayur-HP:~/SPOSL$  cat mnt.txt
M1      2       2       1       1
M2      2       2       7       3
M3      2       0       13      4

mayur-r@Mayur-HP:~/SPOSL$  cat mdt.txt
MOVE #3,#1
ADD #3,='1'
MOVER #3,#2
M2 69,169
ADD #3,='5'
MEND
MOVER #3,#1
MOVER #4,#2
M3 73,173
ADD #3,='15'
ADD #4,='10'
MEND
```

```
ADD #1,#2
MEND

mayur-r@Mayur-HP:~/SPOSL$  cat kpdt.txt
a               AREG
b               -
u               CREG
v               DREG

*/
```

Mayur Gorane
TE(A) 48

Experiment no-4

Aim : Design 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 & file without any macro definitions) should be input for this assignment

Theory :

1. Macro processor
It is a program that reads a files & scans them for certain keywords. When a keyword is found, its replaced by some text. The keyword / text combination is called a Macro.

2. Basic tasks performed by Macro processor
a. Recognize macrodefinition
b. Save the definition
c. Recognize call
d. Expanded calls & substitute arguments.

Pass2 Macro calls & expansions :
Pass 2 algorithm examines the operation code of every input line to check whether it exists in MNT or not.

Steps :
1. Read the input data recieved from Pass-I.

2. Examine each operalion code for finding respective entity in the MNT.
3. 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 (MDTR)
   b: Prepare argument list array containing a table of dummy arguments.
   c. Increase value of MDTR by value one.
   d. Read Next line from MDT.
   e. Substitute values from arguments list of macro of dummy arguments
   f. If mena pseudo code is found then next source of i/p data is record.
   g. Else expand data input
4. When macro name is not found then create expanded data file.
5. If end pseudo code is encountered then feed expanding source file to assembler for pocessing
6. Else read next Source of data input

Conclusion :

Thus pass II of macro processor is implemented & ALA file is generated.

Assignment No. 04 [PASS-2 Macroprocessor]

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:

```
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,initializ
edParams[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:

OUTPUT:
mayur-r@Mayur-HP:~/SPOSL$ javac macroPass2.java
mayur-r@Mayur-HP:~/SPOSL$ java macroPass2
mayur-r@Mayur-HP:~/SPOSL$ cat Pass2.txt

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
```

Mayur Gorano
TE(A) 48

Experiment no : 5

Aim : Design Lex program to generate token of given input file

Problem statement : Write a program using Lex specification to implement lexical analysis phase of compiler to generate tokens of subset of Java program

Pre-requisite :- LEX 110, LEX 120, LEX 130, LEX 140, LEX 160, 250

Theory : Lexical analyzer is a tool for generating scanners. Scanners are programs that recognize lexical patterns in text. A method regular expression may have an associated action. Lex turns users expressions and actions into host generals purpose language, 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 divided in 3 steps :

1. Specify the pattern : associated actions in a format lex can understand
2. Run lex over this file to generate c code for scanner.
3. Compile & link c code to procedure executable scanner.

Lex program is divided in 3 sections :
1. Global C and lex declaration.
2. Patterns
3. Supplement C functions.

The sections are delimited by %%.

A character class define a single character. Two operators supported in a character class are hyper ("_") and circumflex ("^")

```
.. definition
%%
.. rules

%%
.. subroutines
```

Input to Lex is divided in 3 sections with %% dividing section %%. Input is copied to output one character at a time. The first %% is always required as there always must be rules section. If we dont specify any rules then default action is to match everything and copy is to output. Defaults for input & output are stdin & stdout.

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

Assignment No. 05 [LEX Program]

Problem Satement: 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:

```
%{
   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;

}
```

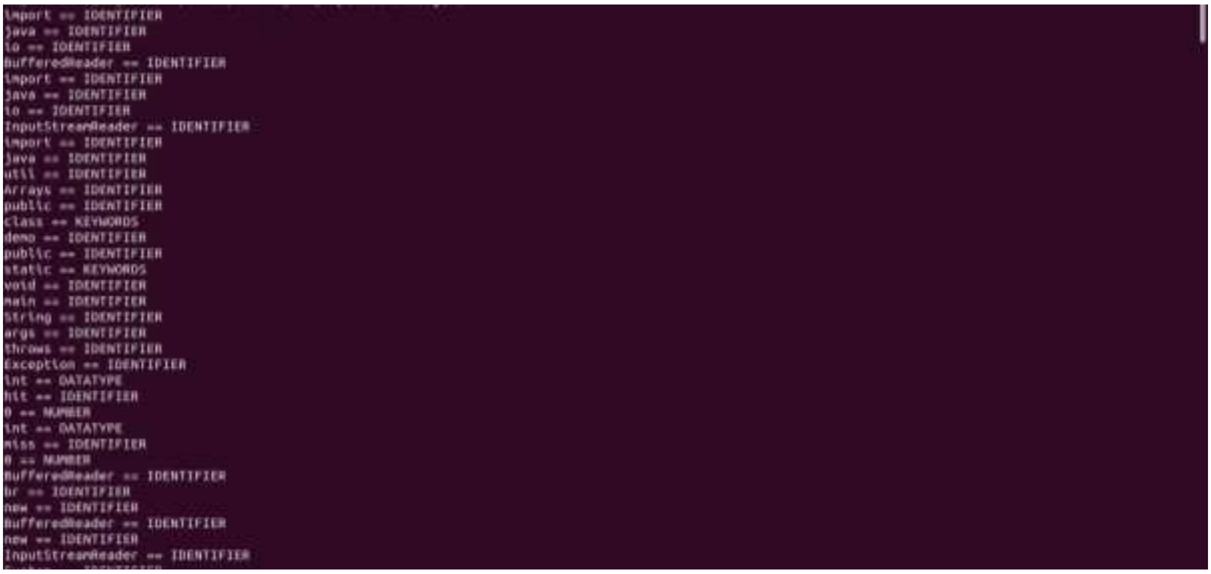
2. Demo.java Code:

```
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");

                                        }
                                        e
                                        l
```

```
                                            s
                                            e
                                            {

                                                  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;
                                                  }
                                                  e
                                                  l
                                                  s
                                                  e
                                                  {

//user lru algo to find replace location

                                                      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;
                    }

                    retu
          rn -1; }


}
```

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
```

OUTPUT:

mayur-r@Mayur-HP:~/SPOSL/LexProgram$ lex b2.l mayur-r@Mayur-HP:~/SPOSL/LexProgram$ gcc lex.yy.c mayur-r@Mayur-HP:~/SPOSL/LexProgram$ ./a.out demo.java

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

mayur-r@Mayur-HP:~/SPOSL/LexProgram$
```

Mayur Gorane
TE(A) 48

Experiment no : 6

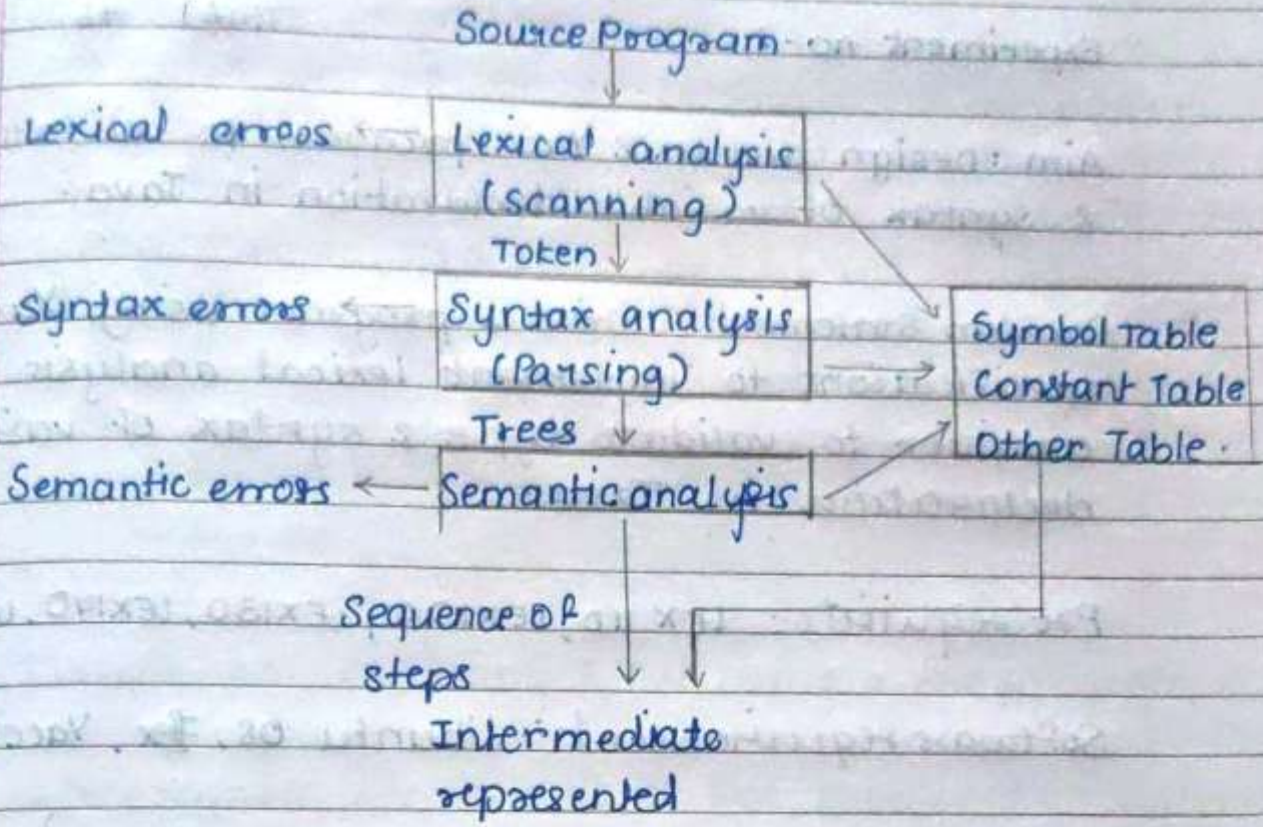
Aim : Design Lex program to countro of words, lines & characters of given input files

Problem Statement : write a program using Lex specifications to implement lexical analyses phase of compiler to couxt no of words, lines and characters of given input file :

Pre requisite : LEX Basics

Software requirements : Ubuntu OS with Lextool (flex)

Theory : How the input is method ?
When generated scanner runs, it analyzes its input looking for steings which match any of it patterns If it finds more than one match it takes one matching most text. If its finds 2 or more matches of same length rule listed frost in the flex input file is chooren. Once match is determined text corresponding to match is made available in yytext and its length is yyleng. If no match is found, then default rate is executed the next character in input is considered matched & copied to std o/p.



Conclusion :

Thus we have studied lexical analysis & implemented an application for lexical analyzer to count total no. of words, characters & lines, etc.

Assignment No. 06 [LEX Program]

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

1. Code b3.l:

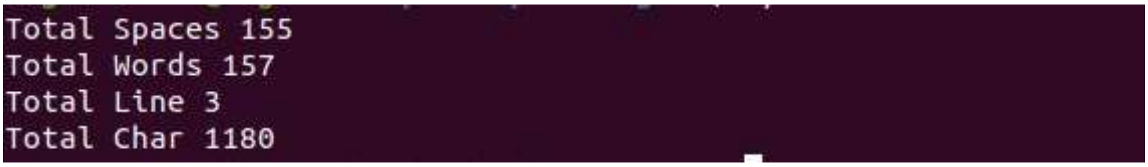
```
%{

int
no_line=0;
int
no_space=0;
int
no_char=0;
int
no_words=0;
#include<stri
ng.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
```

Mayur Gorane
TE(A) 48

Experiment no-7

Aim : Design and lex & Yacc program to validate type & 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 requisite: LEX 110, LEX120, LEX130, LEX140, LEX160, 25

Software requirement: Ubuntu OS, Flex, Yacc(lex&Yac

Theory : Yacc (Yet another compiler -compiler) is a a computer program for UNIX operating system developed by Stephen C. Johnson. It is a look ahea left to right parsor generator part of a compiler that tries to make syntatic sense of source code Based on analytic sense of source code similar job is to analyze structure of input stream & operate of '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 with lex, a no of combinations of pattern & action. I/p to Yacc is divided into three sections The defintion section consist of token declaration and c code bracketed by "%{ and %}"
The BNF grammar is placed in rules sections. It can be used to express context free languages

```
Eg  E → E+E
    E → E*E
    E → id.
```

Translating compiling & Executing a Yacc program.

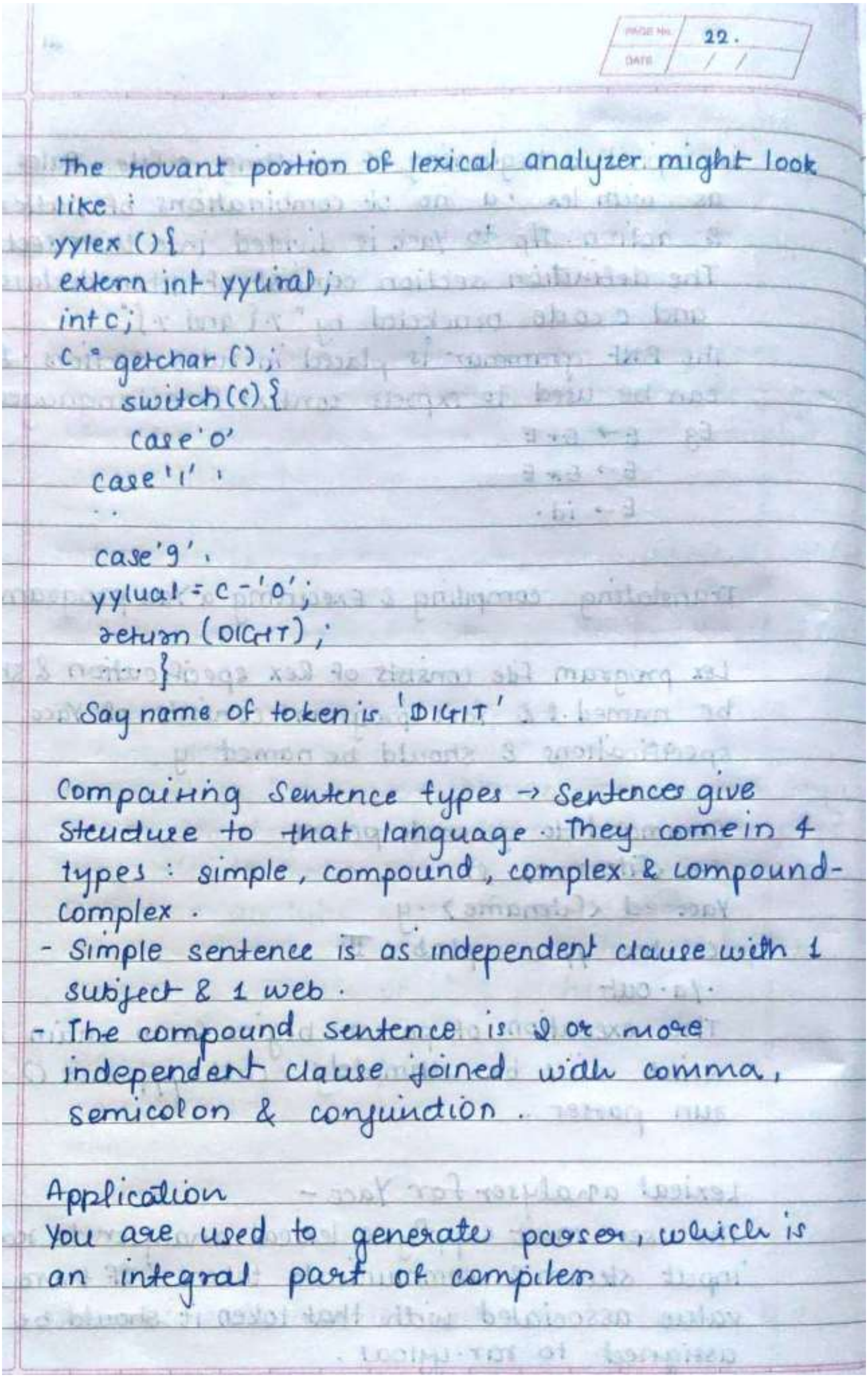
Lex program file consists of lex specification & should be named .l & Yacc program consists of Yacc specifications & should be named .y

Command to generate parser.

```
Lex <filename>.l
Yacc -d <filename>.y
cc lex yy c ytab c II
./a.out
```

The execution of parser begins from main fun^n which will be ultimately call yyparse() to run parser.

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



The rovant portion of lexical analyzer might look like :

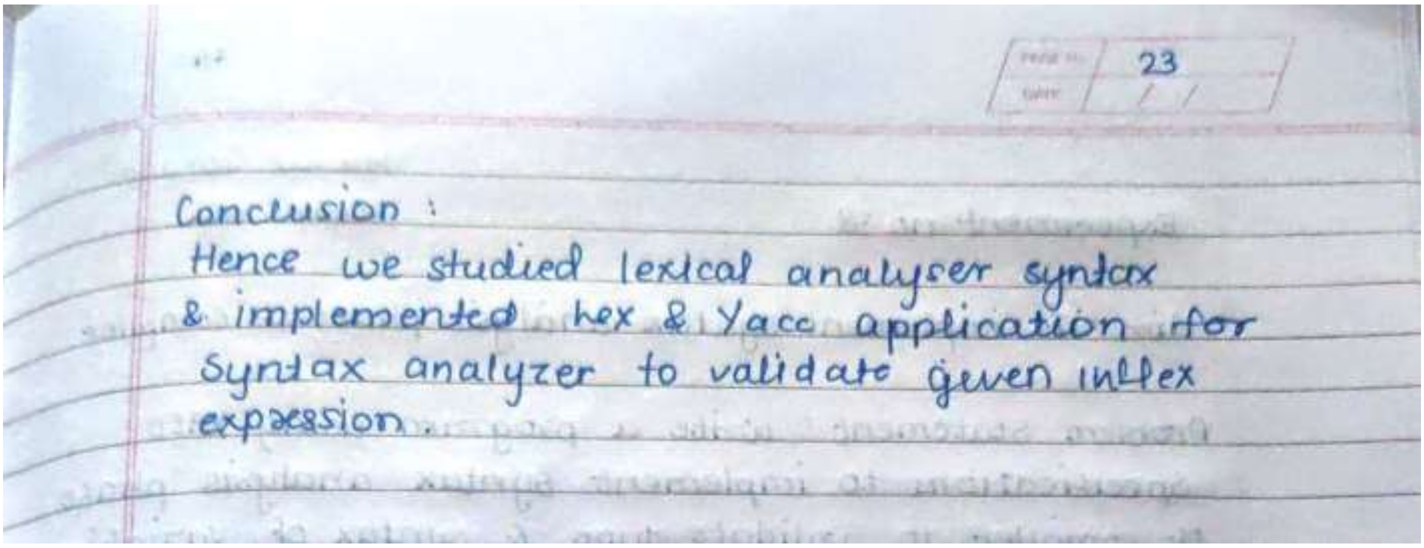
```
yylex (){
extern int yylval;
int c;
c = getchar ();
   switch(c){
     case'0'
   case'1' :
   ..

   case'9'.
   yylval = c-'0';
   return (DIGIT);
   }
   Say name of token is 'DIGIT'.
```

Compairing Sentence types → Sentences give Structure to that language. They come in 4 types : simple, compound, complex & compound-complex.

- Simple sentence is as independent clause with 1 subject & 1 web.
- The compound sentence is 2 or more independent clause joined with comma, semicolon & conjunction.

Application

You are used to generate parser, which is an integral part of compiler.



23

Conclusion :

Hence we studied lexical analyser syntax & implemented Lex & Yacc application for Syntax analyzer to validate given infix expression.

Name: Mayur Vijay Gorane **DivRoll**: TE-A-48

Assignment No. 07 [LEX Program]

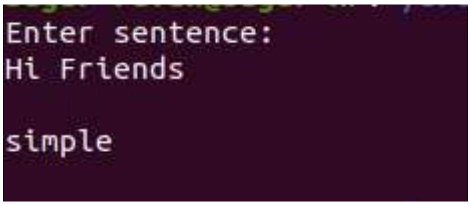
Problem Satement: Write a program using YACC specifications to implement syntax analysis phase of compiler to recognize simple and compound sentences given in input file

---

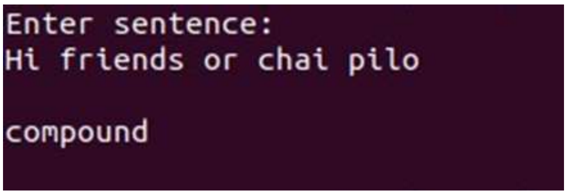
1. Code b5.l:

```
%{
   #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");


}
el
se
{

       printf("simple\n\n");

   }
retur
n 0;

}
```

OUTPUT



```
Enter sentence:
Hi Friends

simple
```



```
Enter sentence:
Hi friends or chai pilo

compound
```

Mayur Gorane
TE(A) 48

Experiment no : 8

Aim : To implement syntax analysis phase of compiler.

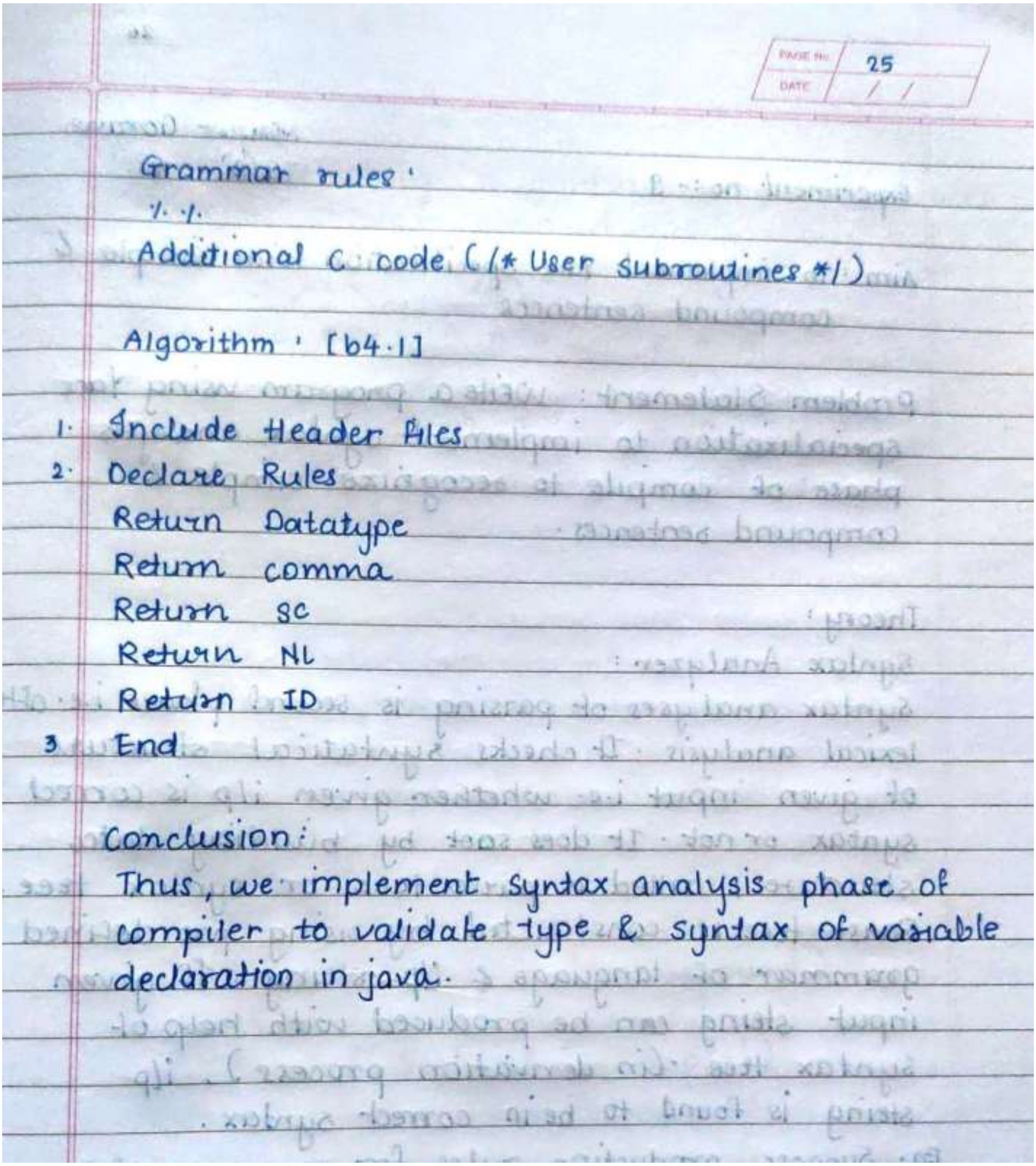
Problem Statement : write a program using YACC specifications to implement syntax analysis phase of compiler to validate type & syntax of variable declaration in Jawa.

Theory :

YACC (Yet another compiler - compiler)
Its a standard parser generator for UNIX OS. An open source program yacc generate code for parser in C programming language. The acronym is usually rendered in lowercase but is occassionally seen as YACC or Yacc. This original version of Yacc was written by stephen Johnson at American 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 declaration
%}
   yacc declarations
%%
```



Grammar rules :

%%

Additional C code (/* User subroutines */)

Algorithm : [b4.1]

```
1. Include Header Files
2. Declare Rules
   Return Datatype
   Return comma
   Return sc
   Return NL
   Return ID
3. End
```

Conclusion:

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

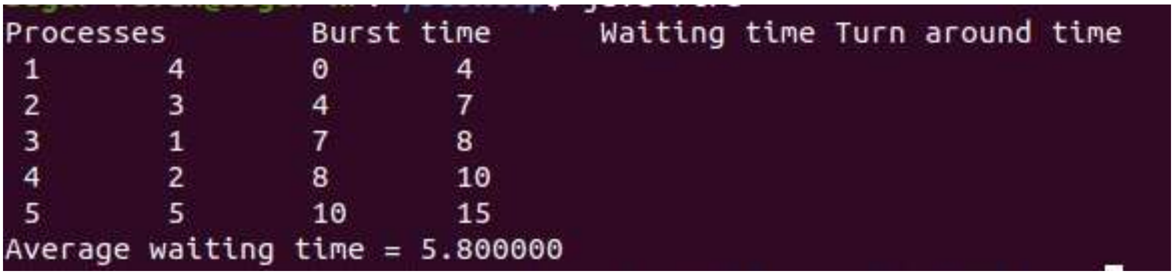
Assignment No. 08

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

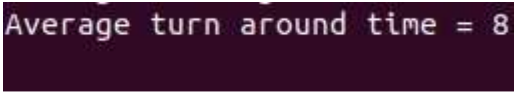
1. FCFS Program:

```
// Java 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 \t Burst time \t 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,
                3,4,5}; int n =
                processes.length;

                //Burst time of all processes int

                burst_time[] = {4,3,1,2,5};

                findavgTime(processes, n,

                burst_time);

        }

}
```

FCFS OUTPUT:



```
Processes        Burst time      Waiting time Turn around time
 1       4       0       4
 2       3       4       7
 3       1       7       8
 4       2       8       10
 5       5       10      15
Average waiting time = 5.800000
```



```
Average turn around time = 8
```

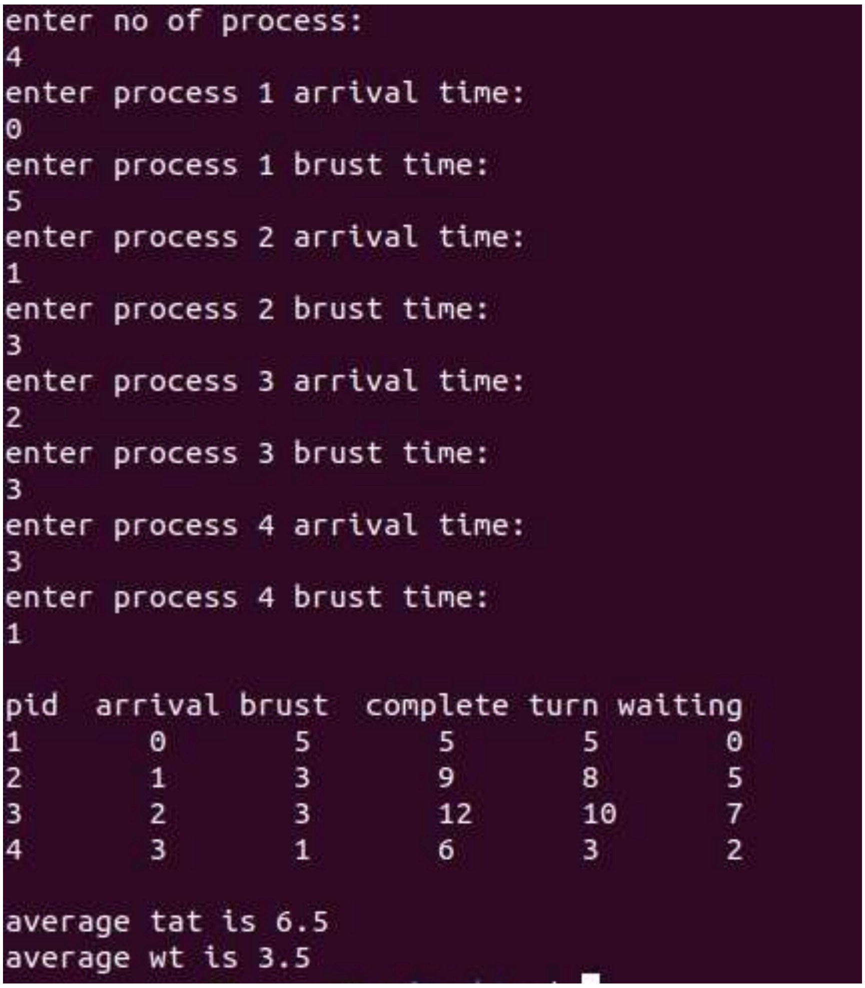
2. Shrtest Job First Program:

```
import java.util.*;


public 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.o
            ut.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();
    }
}
```

SJF OUTPUT:



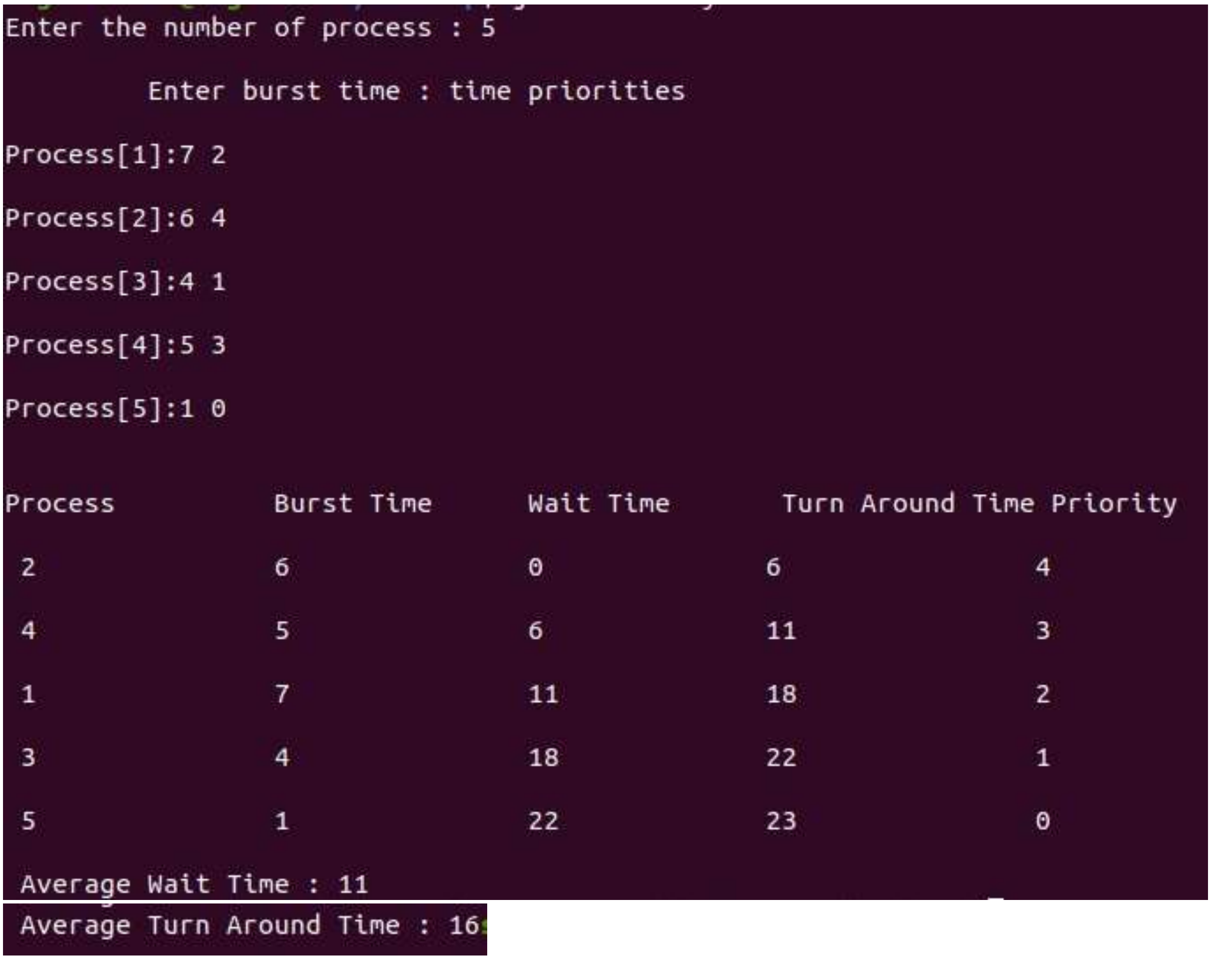
```
enter no of process:
4
enter process 1 arrival time:
0
enter process 1 brust time:
5
enter process 2 arrival time:
1
enter process 2 brust time:
3
enter process 3 arrival time:
2
enter process 3 brust time:
3
enter process 4 arrival time:
3
enter process 4 brust time:
1

pid  arrival brust  complete turn waiting
1       0       5       5       5       0
2       1       3       9       8       5
3       2       3       12      10      7
4       3       1       6       3       2

average tat is 6.5
average wt is 3.5
```

3. Priority Program:

```
import java.util.Scanner;
public class Priority {

 public static void main(String args[])
{  Scanner s = new
Scanner(System.in);  int
x,n,p[],pp[],bt[],w[],t[],awt,atat,i;

 p = new int[10];
pp = new int[10];
bt = new int[10];  w
```

```
= new int[10];  t =
new int[10];  //n is
number of process

 //p is process

 //pp is process priority

 //bt is process burst time

 //w is wait time

 // t is turnaround time

 //awt is average waiting time

 //atat is average turnaround time

 System.out.print("Enter the number of process : ");

 n = s.nextInt();

 System.out.print("\n\t Enter burst time : time priorities \n");
for(i=0;i<n;i++)

 {

 System.out.print("\nProcess["+(i+1)+"]:");

 bt[i] = s.nextInt();
pp[i] = s.nextInt();
p[i]=i+1;

 }

//sorting on the basis of priority  for(i=0;i<n-
1;i++)

 {

 for(int j=i+1;j<n;j++)

 {
if(pp[i]<p
p[j])  {
x=pp[i];
pp[i]=pp[
j];
pp[j]=x;
x=bt[i];
bt[i]=bt[j
];
bt[j]=x;
x=p[i];
p[i]=p[j];
p[j]=x;  }

 }
}
w
[
0
]=
0;
a
w
t
=
0;

t[0]=bt[0];
atat=t[0];
for(i=1;i<n;i++
)

 {
w[i]=t[i-
1];
awt+=w[i]
;
t[i]=w[i]+
bt[i];
atat+=t[i];

 }

//Displaying the process

 System.out.print("\n\nProcess \t Burst Time \t Wait Time \t Turn Around Time Priority \n");
for(i=0;i<n;i++)

 System.out.print("\n "+p[i]+"\t\t "+bt[i]+"\t\t "+w[i]+"\t\t"+t[i]+"\t\t
"+pp[i]+"\n"); awt/=n; atat/=n;

 System.out.print("\n Average Wait Time : "+awt);

 System.out.print("\n Average Turn Around Time : "+atat);


}
}
```

Priority OUTPUT:



```
Enter the number of process : 5

        Enter burst time : time priorities

Process[1]:7 2

Process[2]:6 4

Process[3]:4 1

Process[4]:5 3

Process[5]:1 0

Process          Burst Time      Wait Time       Turn Around Time Priority

 2               6               0              6                4

 4               5               6              11               3

 1               7               11             18               2

 3               4               18             22               1

 5               1               22             23               0

 Average Wait Time : 11
 Average Turn Around Time : 16
```

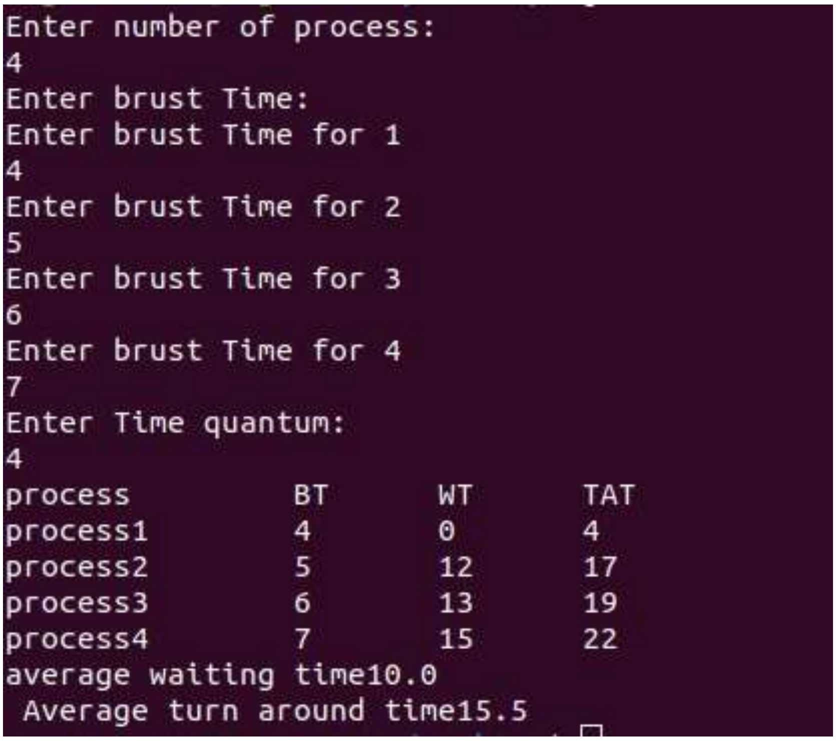
4. Round Robin Program:

```
import
java.io.*;
class
RoundR {

public static void main(String args[])throws IOException

{

DataInputStream in=new DataInputStream(System.in);
int i,j,k,q,sum=0;

System.out.println("Enter number of
process:"); int
n=Integer.parseInt(in.readLine()); int
bt[]=new int[n]; int wt[]=new int[n]; int
tat[]=new int[n]; int a[]=new int[n];

System.out.println("Enter brust Time:");
for(i=0;i<n;i++)

{

System.out.println("Enter brust Time for "+(i+1));

bt[i]=Integer.parseInt(in.readLine());

}

System.out.println("Enter Time
quantum:");
q=Integer.parseInt(in.readLine());
for(i=0;i<n;i++) a[i]=bt[i]; for(i=0;i<n;i++)
```

```
wt[i]=0;
do {
for(i=0;i<n
;i++)

{

if(bt[i
]>q) {

bt[i]-=q;
for(j=0;j<n;j++) {
if((j!=i)&&(bt[j]!
=0)) wt[j]+=q; }

}

else {
for(j=0;j<n;j++) {
if((j!=i)&&(bt[j]!
=0))

wt[j]+=
bt[i]; }

bt[i]=0;

} } sum=0;
for(k=0;k<n;
k++)
sum=sum+b
t[k];

}
while(sum!
=0);
for(i=0;i<n;
i++)
tat[i]=wt[i]
+a[i];

System.out.println("process\t\tBT\tWT\tTAT");
for(i=0;i<n;i++)

{

System.out.println("process"+(i+1)+"\t"+a[i]+"\t"+wt[i]+"\t"+tat[i]);

}

float
avg_wt=0;
float
avg_tat=0;

for(j=0;j<n;j++)

{

avg_wt+=wt[j];

}
```

```
for(j=0;j<n;j++)

{

avg_tat+=t
at[j]; }

System.out.println("average waiting time"+(avg_wt/n)+"\n Average turn around time"+(avg_tat/n));

}

}
```

Round Robin OUTPUT:



```
Enter number of process:
4
Enter brust Time:
Enter brust Time for 1
4
Enter brust Time for 2
5
Enter brust Time for 3
6
Enter brust Time for 4
7
Enter Time quantum:
4
process         BT      WT      TAT
process1        4       0       4
process2        5       12      17
process3        6       13      19
process4        7       15      22
average waiting time10.0
 Average turn around time15.5
```

Mayur Gorane
TE(A) 48

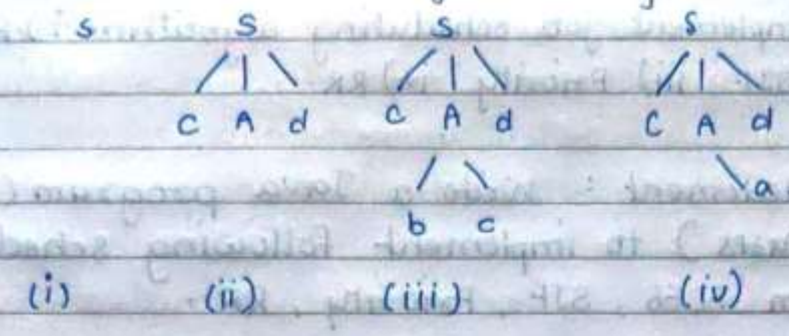
Experiment no :- 9

Aim : To implement Yacc specification with simple & compound sentences

Problem Statement : Write a program using Yacc specialization to implement syntax analysis phase of compile to recognize simple & compound sentences.

Theory :
Syntax Analyzer :
Syntax analyses of parsing is second phase i.e. after lexical analysis. It checks syntatical steucture of given input i.e. whether given i/p is correct syntax or not. It does soot by building data steuctuse, called a parse teee or syntax teee. Parse teee is constcucted by using pre defined grammar of language & i/p steing. If given input steing can be produced with help of syntax teee (in derivation process), i/p steing is found to be in correct syntax.
Eg. Suppose production rules for grammar of a language are
S → cAd
A → bc/a
And input steing is 'cad"

Now parser attempts to consteuct syntax tree from this grammar for given input steing. It uses given production rules & applies those as needed to generate steing · To generate steing 'cad' it uses rules as shown in given diagram ·



Algorithm

```
Begin
include header files
 define compound = 0
 define rules
 define yywrap()
 take input
  yylex();
  checkif ( compound = = 1)
       print (' Sentence is compound')
    else ·
       print (' Sentence is simple');
 Return
 End
```

Conclusion :
Thus we have implemented syntax analyses phase of compiler to recognize simple & compound statement given in i/p file

Assignment No. 09

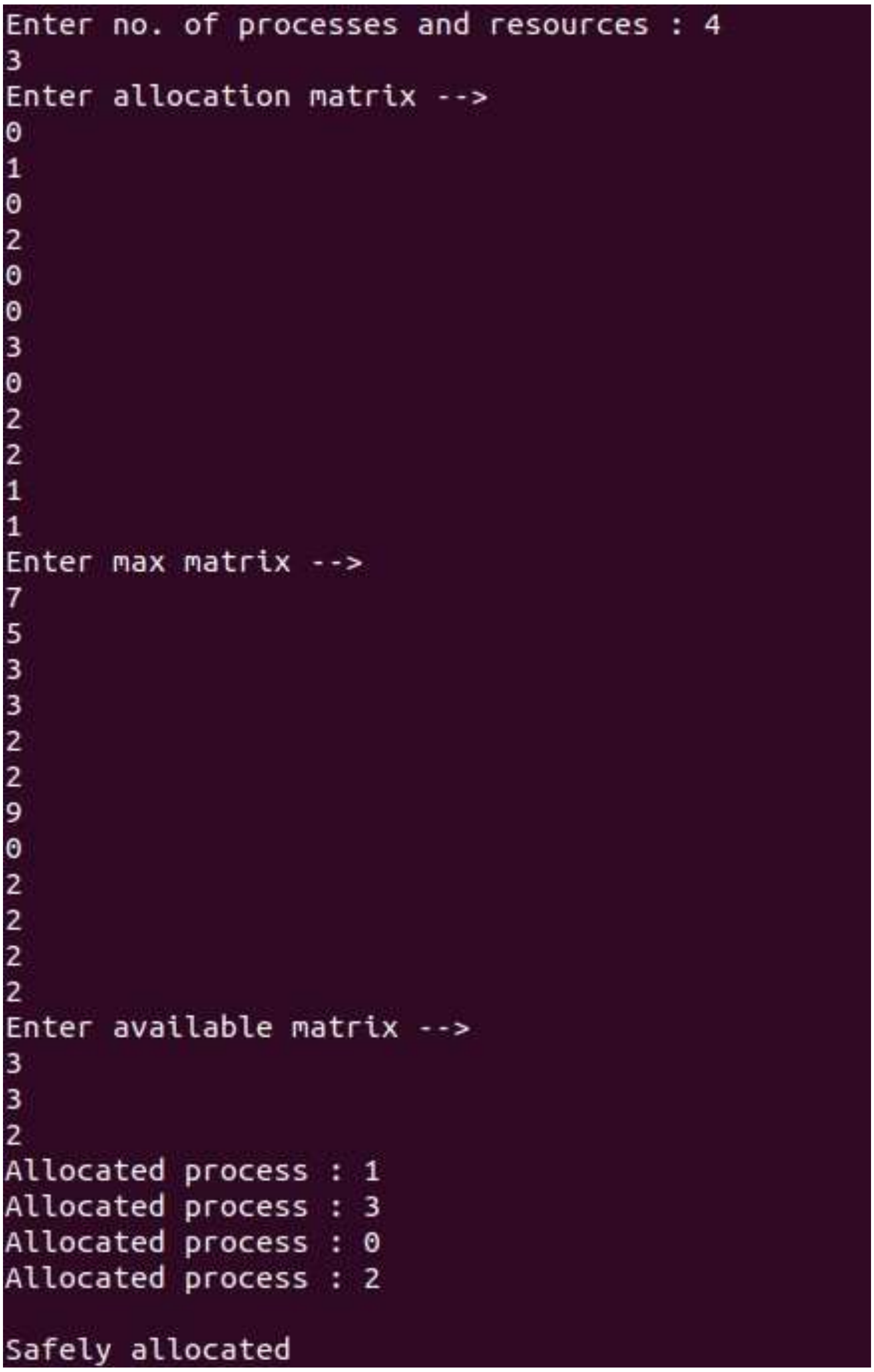
Problem Satement: Write a Java program to implement Banker’s Algorithm

1. Banker’s Algorithm Program:

```
import java.util.Scanner; public class Bankers{
private int
need[][],allocate[][],max[][],avail[][],np,nr;

 private void input(){
 Scanner sc=new Scanner(System.in);
 System.out.print("Enter no. of processes and resources :
");  np=sc.nextInt(); //no. of process  nr=sc.nextInt(); //no.
of resources  need=new int[np][nr]; //initializing arrays
max=new int[np][nr];  allocate=new int[np][nr];
avail=new int[1][nr];

 System.out.println("Enter allocation matrix --
>");  for(int i=0;i<np;i++)  for(int j=0;j<nr;j++)
 allocate[i][j]=sc.nextInt(); //allocation matrix

 System.out.println("Enter max matrix --
>");  for(int i=0;i<np;i++)  for(int
j=0;j<nr;j++)
 max[i][j]=sc.nextInt(); //max matrix

 System.out.println("Enter available matrix --
>");  for(int j=0;j<nr;j++)
avail[0][j]=sc.nextInt(); //available matrix

 sc.close();
 }

 private int[][] calc_need(){  for(int
i=0;i<np;i++)  for(int j=0;j<nr;j++)
//calculating need matrix
need[i][j]=max[i][j]-allocate[i][j];

 return need;  }  private
boolean check(int i){
 //checking if all resources for ith process can be
allocated  for(int j=0;j<nr;j++)  if(avail[0][j]<need[i][j])
return false;
```

```
 return true;  }  public void isSafe(){
input();  calc_need();  boolean
done[]=new boolean[np];  int j=0;
while(j<np){ //until all process allocated
boolean allocated=false;  for(int
i=0;i<np;i++)  if(!done[i] && check(i)){
//trying to allocate  for(int k=0;k<nr;k++)

 avail[0][k]=avail[0][k]-
need[i][k]+max[i][k];
System.out.println("Allocated process :
"+i);  allocated=done[i]=true;  j++;  }
if(!allocated) break; //if no allocation

 }  if(j==np) //if all processes are
allocated
System.out.println("\nSafely
allocated");  else

 System.out.println("All proceess cant be allocated safely");

 }


 public static void main(String[] args) {
new Bankers().isSafe();


}
}
```

OUTPUT:



```
Enter no. of processes and resources : 4
3
Enter allocation matrix -->
0
1
0
2
0
0
3
0
2
2
1
1
Enter max matrix -->
7
5
3
3
2
2
9
0
2
2
2
2
Enter available matrix -->
3
3
2
Allocated process : 1
Allocated process : 3
Allocated process : 0
Allocated process : 2

Safely allocated
```

Mayur Gorane
TE(A) 48

Experiment no : 10

Aim : Implement job scheduling algoeithm i) FCFS ii) SJF iii) Priority iv) RR.

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

Theory :

① First come first serve:
The process that request CPU first, is the one to which its allocated first. The algorithm is implemented using a job queue. When a process request CPU its added at tail of job queue The CPU is allocated to process at head of queue.

Algorithm :

1. Input process along with their, burst time (bt)
2. Find waiting time (wt) for all processes.
3. At first process that comes need not wait so waiting time for process 1 will be 0 i.e. wt [0] = 0
4. Find waiting for all other process i.e. for process i → wt [i] = bt [i-1] + wt [i-1]
5. Find turnaround time = waiting - time + burst - time for all processes.
6. Find average waiting time = total_waiting_time / no_of_processes
7. Similarity find average turnarounds time = total_turn_around_time / no of process.

② Shoetest Job First :

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

Algorithm :

1. Sort all processes is increased order according to burst time
2. Then simply, apply FCFS

③ Priority Scheduling :

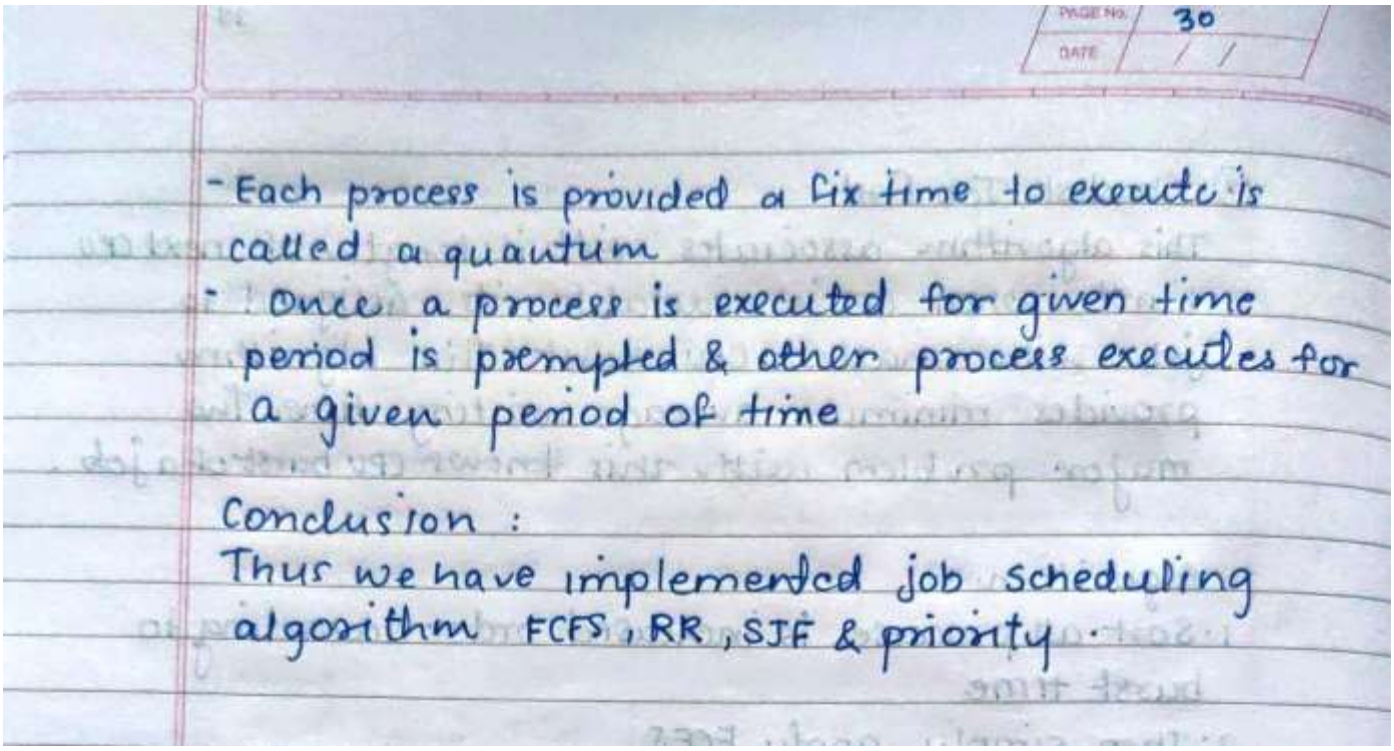
- Priority scheduling is a non primitive algorithm
- Its most common scheduling algorithm in batch systems.
- Process with same priority are executed on FCFS basiss
- Priority can be decided based on memory requirement time requirements or any other resource requirement

Algorithm

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

④ Round robin Scheduling :

- RR is CPU scheduling algorithm where each process is assigned fixed time slot in a cyclic way
- Its primitive as process are assigned CPU only for a fixed slice of time at most .



PAGE No. 30

DATE / /

- Each process is provided a fix time to execute is called a quantum
- Once a process is executed for given time period is prempted & other process executes for a given period of time

Conclusion :

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

Name: Mayur Vijay Gorane **DivRoll**: TE-A-48

____

Assignment No. 10 [UNIX System Calls]

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

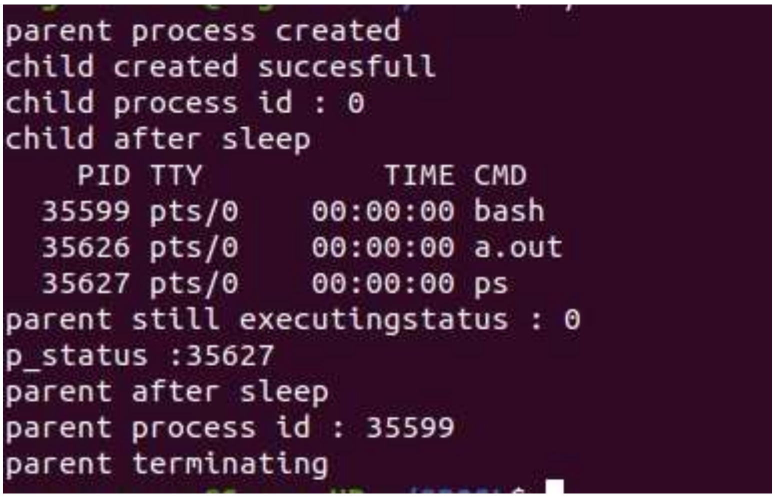
____

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;

}
```

OUTPUT:



```
parent process created
child created succesfull
child process id : 0
child after sleep
    PID TTY          TIME CMD
  35599 pts/0    00:00:00 bash
  35626 pts/0    00:00:00 a.out
  35627 pts/0    00:00:00 ps
parent still executingstatus : 0
p_status :35627
parent after sleep
parent process id : 35599
parent terminating
```

Mayur Gorane
TE(A) 48

Experiment no : 11

Aim : Bankers algorithm for deadlock detection & avoidance

Problem Statement : write a Java program to implement implement Banker's algorithm

Theory :
The banker's algorithm is a resource allocation & deadlock avoidance algorithm that texts 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 continued.

Following data stuctures are used to implement banker's algorithm
Let 'n' be no of processes in system & 'm' be no of resources types :

Available
- Its a 1-d array of size 'm' indicating no of available resources of each type.
- Available [j] = k means there are 'k' instances of resources type Rj.

Max :
- Its a 2-d array of size 'n * m' that defines maximum demand of each process in a system M[i, j] = k means process Pi may request at most 'k' instances of resource type Rj.

Allocation :

- Its a 2-d array of size 'n*m' that defines no. of resources of each type currently allocated to each process.
- Allocation [i,j]=k means P process Pi is currently allocated 'k' instances of resource type Rj

Need :

Its a 2-d array of size 'n * m' that indicate remaining source need of each process

- Need [i,j] = Max [i,j] - Allocation [i,j]

Safety Algorithm :

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

Conclusion :

Thus we have studied implemented banker's algorithm for deadlock detection & avoidance.

Mayur Gorane

Experiment no : 12 TE(A) 48

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 Kernel to provide access to that resource. This is done via something called system call.
- When program makes a system call, mode is switched from user mode to kernel mode. This called a context switch.

Kernel Mode :

- When CPU is in kernel mode, code being executed can access any memory address & any hardware resource. In user mode if any program crashes 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 in an OS runs in user mode.

Unix System Calls :

PS command : This command is used to provide info about currently running processes, including their process identification nos (PID)

Fork command : This command is used to provide info about currently running processes, including their one is child process & other parent process

Join command : Its a command line utility for joining lines of 2 files on a command field.

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

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

Conclusion :
Thus process system call program is implement & studied various system calls.

Assignment No. 12

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

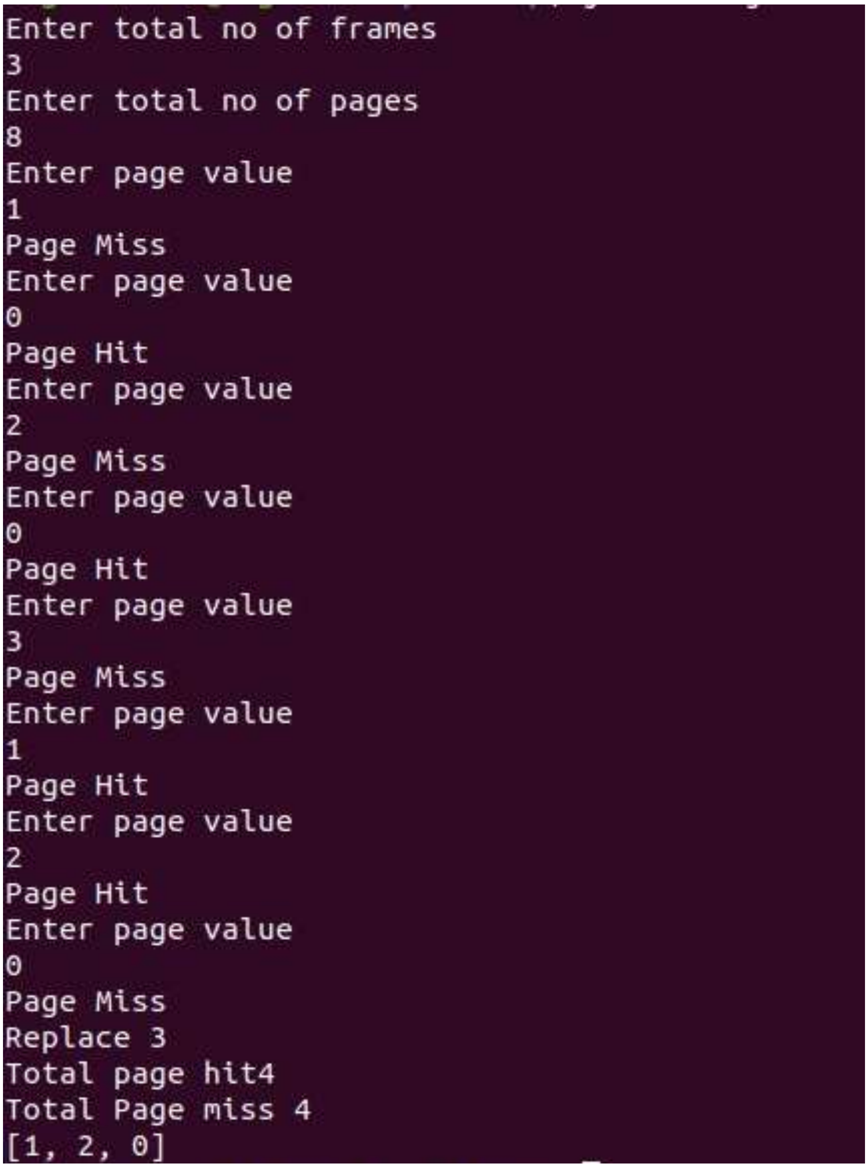
1. LRU (Last Recently Used) Program:

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

public 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");

                                  }
                                  e
                                  l
                                  s
                                  e
                                  {
                                        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;

                                        }
                                        e
                                        l
                                        s
                                        e
                                        {
            //user lru algo to find replace location 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;

                              }

                              retu
                        rn -1; }


     }
```

OUTPUT:



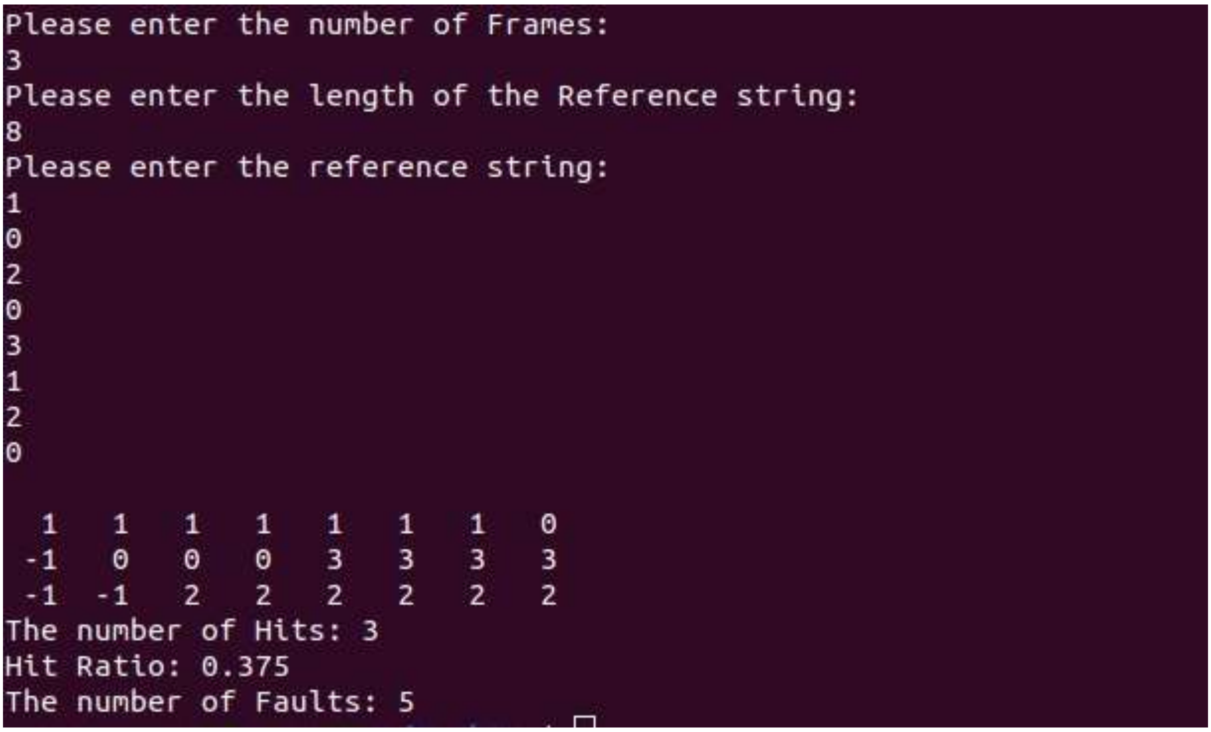
```
Enter total no of frames
3
Enter total no of pages
8
Enter page value
1
Page Miss
Enter page value
0
Page Hit
Enter page value
2
Page Miss
Enter page value
0
Page Hit
Enter page value
3
Page Miss
Enter page value
1
Page Hit
Enter page value
2
Page Hit
Enter page value
0
Page Miss
Replace 3
Total page hit4
Total Page miss 4
[1, 2, 0]
```

2. Optimal Replacement Program:

```
import java.io.BufferedReader;
import java.io.IOException;
import
java.io.InputStreamReader;
public 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])

 {
searc
h = j;
hit++
;
brea
k;  }
}

 if(search == -1)

 {
if(isF
ull)  {

 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
```

Mayur Gorane
TE(A). 48

Experiment no : 13

Aim : Study assignment on process scheduling algorithm in Android & Jizen

Problem Statement : Study assignment on scheduling algorithm in Android & Jizen

Software Requirement : Android SDK

Theory :

Android OS-

Android is a mobile OS developed by Google, based on a modified version of linux kernel & other open Source Software & designed primarily for touch screen, mobiles devices such as smartphones & tablets Those application are more comfortable & 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 2nd largest smartwatch based platform behind watches & ahead of Android wear.

Process scheduling algorithms in Android & Tizen OS

Normal Scheduling -

Android is based on Linux & uses Linux kernels scheduling mechanisms for determining scheduling policies. The linux's time sliced scheduling policy combines static & dynamic priorities

Real Time scheduling :

The standard linux kernel provides 2 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 SCHED-NORMAL scheduling policy

Thread Scheduling

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

Priority based Pre-Emptive Task scheduling for Andooid Operating System.

This scheduling is particularly used for mobile OS as CPU utilization is medium, turnaround & response time is high. Mobile phones are required to meet specific time deadlines for task to occur.

Conclusion :

Thus, we have studied concept of process scheduling of Android & Tizen Operating System.

Mayur Gorane
T.E(A) 48

Experiment no :14

Aim : Implementing page replacement algorithm 1) LRU 2) Optimal.

Problem Statement : To write a java program (using OOP) to implement LRU & optimal algorithm for page replacement.

Prerequisite -

1. Explain concept of virtual memory
2. Define page replacement algorith : LRU & Optimal
3. Explain address translation in paging system
4. Explain Bolady's Anomaly

Theory

1. LRU Page Replacement :-

The main difference between FIFO & optimal page replacement is that FIFO algorithm uses time which page was brought into memory & optimal algorithm uses time when a page is to be used. If we use recent past as an approximation of future then we will replace 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 3 memory frame or blocks. The first 3 reference cases pages fault that full empty frames .

Algorithm:

1. Start traversing pages

If set holds less pages than capacity

a. Insert page into set one by one until the size of set reaches capacity or all page request are processed

b. Simultaneously, maintain recent occured index of each page in a map called indexes.

c. Increment page fault.

II. Else

If current pg is present in set, do nothing

Else

a. Find page in set that was least recently used. we find it using index array. We basically need to replace the place with minimum index.

b. Replace the found page with current page.

c. Increment page faults

d. Update index of current page.

2. Return Page faults :

2. Optimal Replacement :

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

- Often called Baladys Min Bask idea : Replace page that will not be offered for longest time.

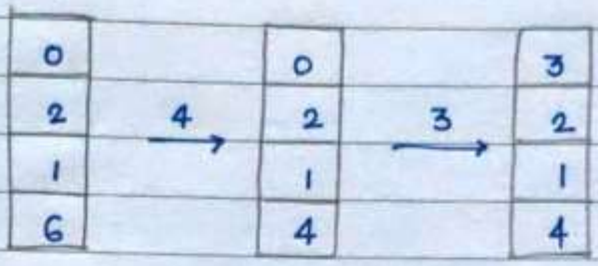
- Impossible to implement .

- consider following reference string

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

3 , 1 , 2 , 1

compulsory misses .



Fault Rate = 6112 = 0.50

Algorithm

1. Start the process
2. Declare the size
3. Get no of pages to be inserted
4. Get the value.
5. Compare counter label & stack
6. Select optimal page by counter value.
7. Stack them according solution .
8. Print pages with fault pages
9. Stop processer .

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