Assignment No 01

Page No.
Date

Aim : To implement Pass-I Assembler.

Problem Statement : Design Suitable data structures and implement Pass I of a two-Pass assembler for pseudo-machine in Java using objected oriented features. Implementation should consist of a few instructions from each category and few assembler directives.

Theory :

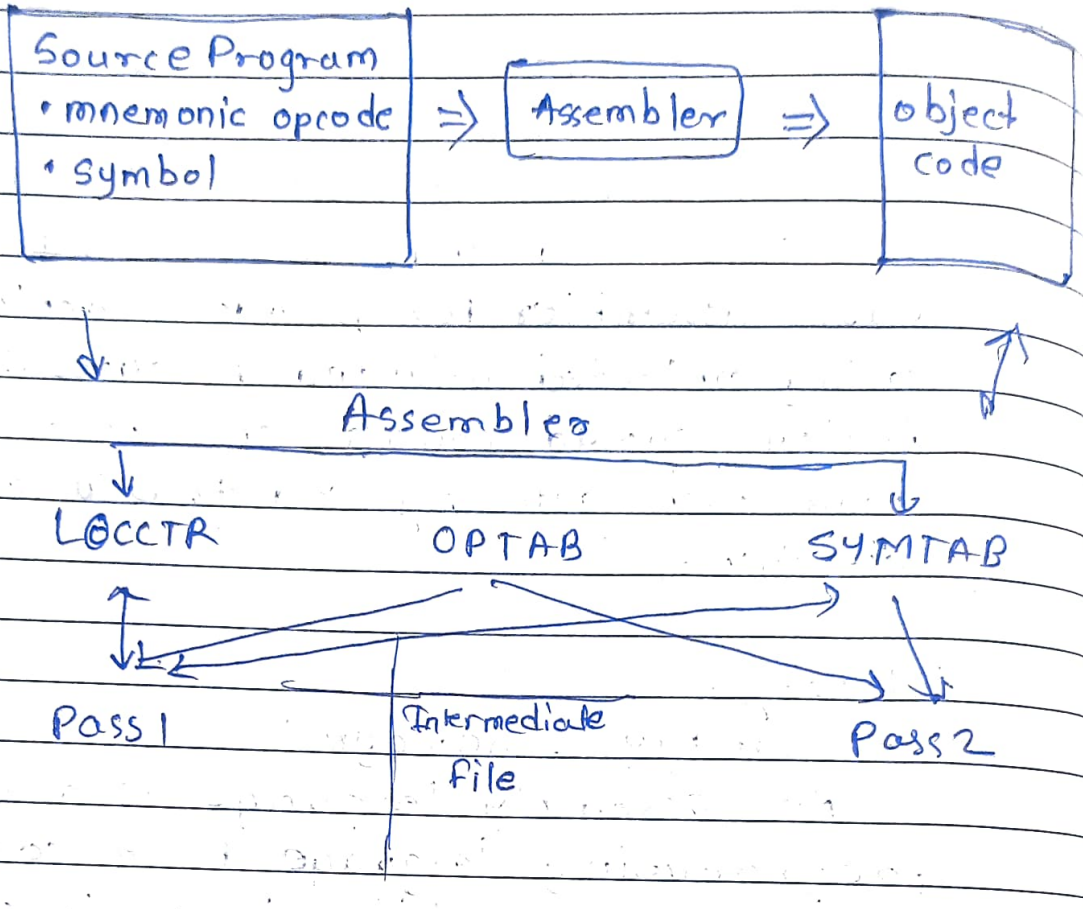
Assembly Language :

An assembly language is a low-level programming language for a computer, or other programmable device, in which there is a very string (generally one-to-one) correspondence between the language and the architectures machine code instructions. Each assembly language.

Assembler

Assembly language is converted into executable machine code by a utility program referred to as an assembler;

An assembler is a translator that translates an assembler program into a conventional machine language program.



Assembler Directives

- Assembler directives are pseudo instructions.
  - ⊙ They will not be translated into machi instructions
  - ⊙ They only provide instruction / direction / information to the assembler.
- Basic Assembler directives
  - START : Specify name and starting addres for program
  - END : Indidate the end of the source program
  - EQU : The EQU directive is used to replace a number by a symbol.

Three main data structures
- Operation Code Table (OPTAB)
- Location Counter (LOCCTR)
- Symbol Table (SYMTAB)

Instruction formats.
- Adressing Modes. Direct addressing. Register addressing. Register indirect addressing. Immediate addressing. Implicit addresing.
- Program relocation. It is desirable to load and run several programs and resources at time. The system must be able to load programs into memory wherever there is room. The exact starting address of the program is not known until load time. The assembler can identify (for the loader) those parts of the program that need no modification. An object program that contains this type of modification information necessary to perform modication called

Data structure for assemble.
Op-ode table
Looked up for the translation of mnemonic code
key - mnemonic code

Algorithum for pass 1 - Assemble.

Page No.
Date

```
begin
    read a line from the code
    if starting address is given
        LOCCTR = starting code address ;
    else
        LOCCTR = 0
while OPCODE , = End do ;; OR EOF
    begin
        read a line from the code
        if there is lable
            if this Label is in SYMTAB then
        else insert (Label , LOCCTR ) into
        SYMTAB search OPTAB for the
        OP code
        if Found.
    LOCCTR + = N ;; N is the length of insert
    else if this is an assembly directive
        update LOCCTR as directed.
    else error.
    write Line to intermediate file
    end.
    program size = LOCCTR - starting address
end.
```

Input :-

```
START 200
MOVER AREG ,='4'
MOVER AREG ,A
MOVER BREG ='1'
LOOP MOVER CREG , B
```

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

Expected output : Symbol table.

A	208
Loop	203
B	209

Intermediate code

```
AD  01  C  200
IS  04  1  2  1
IS  05  1  5  1
IS  04  2  2  2
IS  04  3  5  3
AD  05
IS  01  3  2  3
IS  00
DL  02  C  1
DL  02  C  1
AD  02
```

Conclusion :

Thus we have implemted PASS-1 Assemble using object oriented feature.

Name: Ujjwal Lade **Div Roll**: TE-B-06

Assignment No. 01 [Pass 1 Assembler]

Problem Satement: 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:

```
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 = /Desktop/Input.txt"; fr = new
			FileReader(inputfilename);
			br = new BufferedReader(fr);

			String OUTPUTFILENAME = " /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 = "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 = "/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 = "/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:

Java Pass1 Input.txt

A 8

LOOP 3

B 9

='4' 4

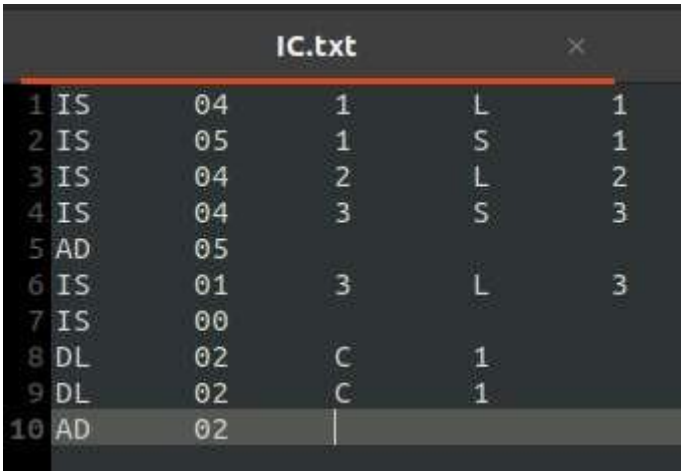
='6' 10

='1' 5

1

3

IC.txt



SYMTAB.txt

LITTAB.txt

POOLTAB.tx

Page No.
Date

Assignment NO. 2.

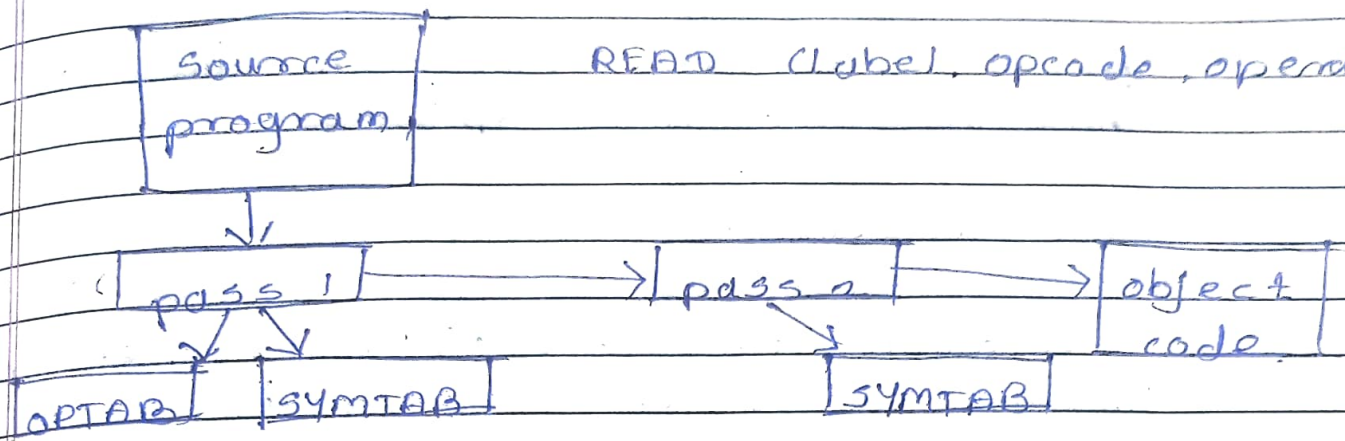
- Aim : To design data structure for Pass-2 assemble.

- Problem statement = To implement Pass II of two pass Assembles for pseudo-machine in java using object oriented features. The output of assignment-I (Intermediate file and symbol table) should be inputte for this assignments.

- Theory -

Two pass assembler

Two pass assembler performs two passes over the source program In the first pass. It reads the entire source programs. Looking only for leble definition. All the labels are collected assigned address and placed in the Symbol table in the programs. No instruction as assembled and at the end the symbol table should contain all the labels defines in the programs. To design address to labels. The assembels maintain a location counters.



Mnemonic & opcode mapping are referenced from here.

Labels and address mapping enteres here.

labels and address mapping are refference from here.

Difference between onepass and two pass Assembler

A one pass assembler passes over the source file exactly once in the same pass collecting the tables, resolving future references and doing the actual assembly. The difficult part is to resolve future label reference and assemble code in one pass. The one pass assembler prepare an intermediate file, which is used as input by the two pass assembler.

In two pass assembler performs two sequential scan over the source code.

pass 1 : symbol and literals are defined.

pass 2 : object program is generated.

Page No.
Date

parsing : moving in program lines to pull out op code and operands.

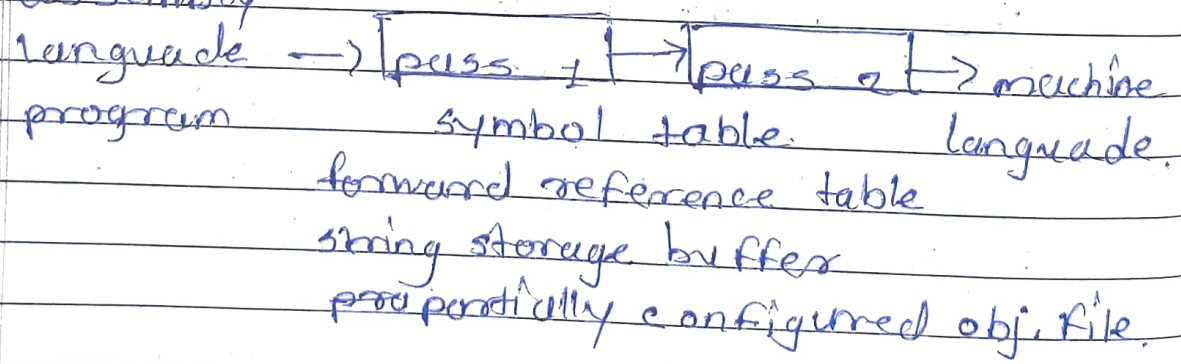
Data structure :

- Location counter (LC) : point to the next location where the code will be placed.

- opcode translation table : Contains symbolic instruction, their lengths and their op-codes.

- Symbol table (ST) : containg Labels and their values.

- starting storage buffer (SSB) : Contain ASCII characters for the strings.



Algorithum.
begin.

If starting addr is given
LOCCTR = starting addr.
else
LOCCTR = 0;
while OPCODE = End do ;; or EOF
begin
read a line from the code
if there is a label
if this label is in SYMTAB there error
else insert (Label, LOCCTR) into SYMTAB
search OP-TAB for the op-code
if found.
LOCCTR + N.
else if this an assembly directive.
update LOCCTR as directed
else error.
write line to intermediate file
end.
program size = LOCCTR - starting addr
end.

Input

IC				LOCCTR			
AD	01	C	200	200	04	1	204
IS	04	1	L	201	05	1	208
IS	05	1	S	202	04	2	210
IS	04	2	L	203	04	3	209
IS	04	2	L	204	00	0	206
IS	04	3	S	205	00	0	206
AD	05			206	01	3	205

Expected o/p.

IS 01 3 L 3 206 01 3 209
IS 00 207 00 0 000
DL 02 C 1 208
DL 02 C 1 209
AD 02 2k 00 0 001

LITTAB txt
='4' 204
='6' 210
='1' 205

SYMTAB txt
A 208
Loop : 205
B 209.

POOLTAB txt

1
3

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

Name: UJJWAL LADE **DivRoll**: TE-B-06

Assignment No. 02 [Pass 2 Assembler]

Problem Satement: 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.

---

1. Pass 2 Program:

```
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 ="/home/sagar-ravan/Desktop/IC_new.txt";
                           FileReader fw =new FileReader(f);
                           BufferedReader IC_file=new BufferedReader(fw);

                           //2.Read Symbol table file
                           String f1="/home/sagar-ravan/Desktop/SYMTAB.txt";
                           FileReader fs=new FileReader(f1);
                           BufferedReader symtab_file=new BufferedReader(fs);
                           symtab_file.mark(500);

                           //3.Read Literal table file
                           String f2="/home/sagar-ravan/Desktop/LITTAB.txt";
                           FileReader fl=new FileReader(f2);
                           BufferedReader littab_file=new BufferedReader(fl);
                           littab_file.mark(500);


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


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

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

                                littab[z][0]=str.split("\\s+")[0]; //first word
                                littab[z][1]=str.split("\\s+")[1]; //second word
```

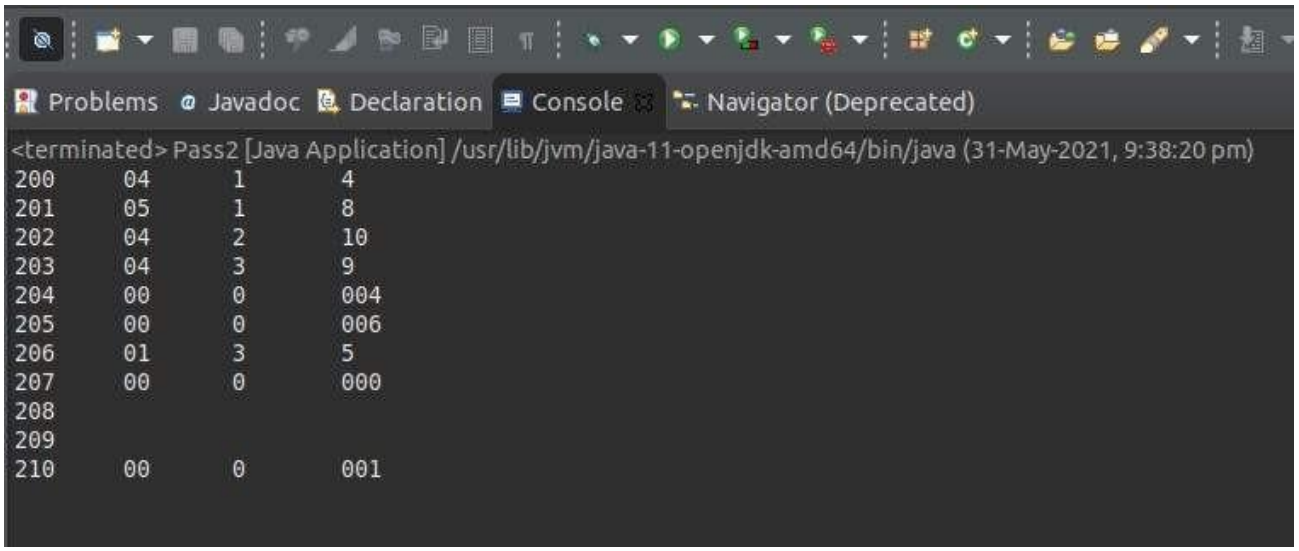
```
                        z++;
                }
                //6.Read SYMTAB.txt

                while ((str = symtab_file.readLine()) != null) {
                        symtab.put(str.split("\\s+")[0], str.split("\\s+")[1]);
                }
        //7.Read POOLTAB.txt
                String f3 = "/home/sagar-ravan/Desktop/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);
                                    }
                            }
                    }                               System.out.print("00\t0\t00" +

"END" stmt

            }
    }
```

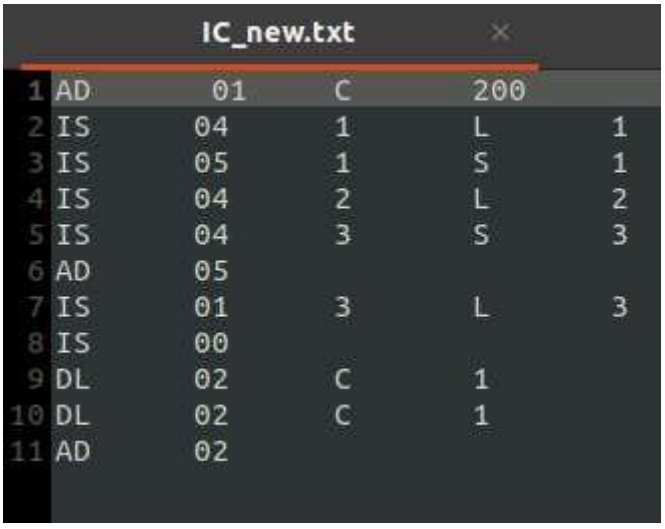
```
                        if(s0.equals("DL")&&s1.equals("01")){          //if it
is DC stmt

                                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:

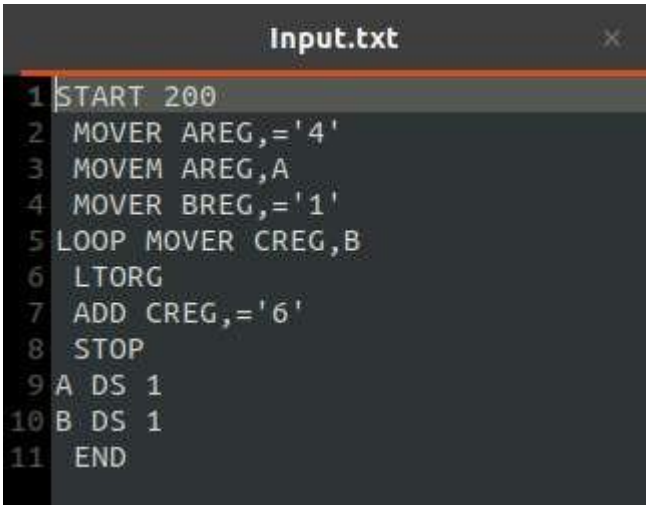
PASS- 2 OUTPUT:



IC_New.txt



Input.txt



LITTAB.txt

POOLTAB.txt

SYMTAB.txt

Assignment No: 03

Page No. | Date

Pass I - Macroprocessor.

Name: Ujjwal Late
Div: B : R.N. 06.
Class : TE (Comp)

Aim : To design Data Structure for macroprocessor.

Problem Statement: Design suitable data structure & implementes pass of two propews mono processor using cor feture in Jawa.

Theory -
① Macro processor:
A macro processor is a program that ridds a file and them for wtain knowords when akeywords is found it is srplaud by home fiek

② Basic lasts programed by macro processor ~~that radle a file~~.
@. Recognize macro definitions.
ⓑ gair the defination.
ⓒ Recognize call.
ⓓ Expainded call and Subtitle argumements

③ Macro defination part.
① Macro prototype Stalimex:
② model stalimer.
③ preprosusor stallary.

④ Macro call & Expansion:
The operation define by macro can be used by using a macro name the mnemonic tab 4 find its quan

⑤ Implement logic:

a. Definition processing.
b. macro Expansion.

⑥ Data structure: required for macro definition.

a. Macro Name table (MNT)
b. parametr name table (PNTAB)
c. keywords parametr dejault table
d. macro definition table (MDT)

⑦ ~~Data~~ Algorithm pseducode:

Bejore processing any definition initial KPD TAB - (P)tr, MDL PTR to MNT - PN to -1

Page No.
Date

Algorithum

```
begin {macroprocessor}
Expanding := FALSE
while OP-CODE <> 'END' do
begin.
  GETLINE
  end {while}
end {macroprocessor}
procedure PROCESS LINE
  begin.
    Search NAMTAB for OPCODE
  if found then
  EXPAND.
else if OPCODE = 'MACRO' then.
  DEFINE
else write source in-line do expended file.
end {processline}.
```

Input.

```
MACRO  INCR X & y & REG1
  ADD REG & y.
  MOVEM & REG & x
MEND
      START  100
      READ   N1
      READ   N2
      INCR   N1. N2
      STOP
      N1  DS1
      N2  DS2
END.
```

C:\ABC> Java Macro Java.
C:\ABC> Java macro.

```
MACRO  INCR  &x  &y  &REG1
       MOVER  &REG1  &x
       ADD    &REG1  &y
       MOVEM  &REG1  &x.
MEND.

START  100
READ   N1
READ   N2
INCR   N1  N2
STOP
N1     DS  1
N2     DS  2
END.
```

★★★★★★★★★★★★★★

MNT:

INDEX	MACRONAME	MDT NAME.
1	INCR	1

★★★★★★★★★★★★★★★★

ALA.

INDEX	ARGUMENT.
#1	&x
#2	&y
#3	&REG1

★★★★★★★★★★★★★★★★

#MDT.

```
MACRO   INCR    &x   &y   &REG
        MOVER   #3   #1
        ADD     #3   #2
        MOVEM   #3   #1
MEND
*********
```

- Conclusion:-

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

Name: UJJWAL LADE DivRoll: TE-B-06

Assignment No. 03 [PASS-1 Macroprocessor]

Problem Satement: 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()),par
amNo
+
                                                        if(keywordParam.length==2)
+);


          f4.write(keywordParam[0].substring(1,keywordParam[0].length())+"\
t"+keywordParam[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"+(k
p==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
:

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

sagar-ravan@Sagar-HP:~/SPOSL$ cat mnt.txt
```

M1	2	2	1	1
M2	2	2	7	3
M3	2	0	13	4

sagar-ravan@Sagar-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

sagar-ravan@Sagar-HP:~/SPOSL$ cat kpdt.txt
a               AREG
b               -
u               CREG
v               DREG

*/
```

Assignment No - 4

Page No. [ ]
Date [ ]

Pass 2 macroprocessor

Aim : Design of a Macro Pass-2

Problem statement : Write your program for pass - II of a two - pass macroprocessor. The output of assignment - B (MNT - MDT and file without any macro defination) should be input for this assigment

Theory

① macroprocessor:

macroprocessor is a program that reads a file (or files) and scans them for certain keywords when a keyword is found it is replaced by some text the keyword text combination is called a macro.

② Basic task performed by macroprocessor

a) Recognize macrodefination
b) save defination
c) Recognize call
d) Expanded call and substitute arguments

In two - pass macroprocessor you have two algorithm to implement first pass second pass Both the algorithm examine line by line over the input data available.

- Pass 1 : macro defination
- Pass 2 : macro calls and expansion

Page No.
Date

Pass I macro defination

Pass I algorithm examine each line of the inputs data for macro pseudo opcode following are the steps that are performed during pass I algorithm

1) Initialize MDTC and MNTC with value one. so that previous value of MDTC and MNTC is set to value one, so that previous value of MDTC and MNTC is set
2) Read the first i/p
3) if this data contains MACRO pseudo operation
   a) Read the next i/p
   b) Enter the name of macro and current value of MDTC in MNT
   c) Increase the counter value of MNT by value
   D) prepare that argument list array
   E) Enter the macro defination im MDT
   F) Read next i/p
   G) Substitute the index notation for dummy argument passed in macro
   H) Increase the counter of the MDT by value one
   I) If mend pseudo opcode in encounter

4) if macro pseudo opcode is not encounted in data input then
   A) A copy of input data is created
   B) if end pseudo opcode is found then go to pass 2
   C) Otherwise read next

Page No.
Date

- Pass-2 macro call and Expansion

pass two algorithm examines the operation code of every input line to check whether it exist in MNT or NOT

1) Read the i/p data received from pass I
2) Examine each operation code for finding respective entery in MNT
3) If Name of macro is encountered then
   A) A pointer is set to the MNT entry where name of the macro is found
   B) Prepare argument list array containing a table of dummy arguments.
   c) Increase the value of MDTP by value one
   d) Read Next line
   E) Substitude the values from the argument list of the macrofor
   f) IF mend pseudo opcode is found then next source of i/p data is Read
   G) Else expands data input
   4) when macro name is not found then create expanded data line
5) IF end pseudo opcode is encountered then Feed the expanded
6) Else read next

Algorithm :

Input

Input
MACRO
INCR1 & FIRST & SECOND = DATA 9
A 1, & FIRST

```
L 2, & SECOND
MEND MACRO
INCR 2 & ARG1 , & ARG2 = DATA5
L 3, & AGR1
ST 4, & ARG2
MEND
PRG2 START
USING *BASE
INCR1 DATA1
INCR2 DATA3, DATA4
FOUR DC F'4'
FIVE DC F'5'
BASE EQU 8
TEMP DS IF
DROP 8
END
```

Output

——— PASS 1 ———

```
ALA
[ & FIRST , & SECOND ]
[ & ARG1 , & ARG2 ]

MNT:
[INCR 1 , 0 ]
[INCR2 , 4]

MDT
INCR & FIRST & SECOND = DATA
1 L 9
A 1, #0
2 2, #1
```

```
MEN
    D    &ARG1,&ARG2:=DATA3
INCR
    2
    L        3,#0
    ST       4,#1
    D
```

Name: UJJWAL LADE **DivRoll**: TE-B-06

Assignment No. 04 [PASS-2 Macroprocessor]

Problem Satement: 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,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:

```
OUTPUT:
sagar-ravan@Sagar-HP:~/SPOSL$ javac macroPass2.java
sagar-ravan@Sagar-HP:~/SPOSL$ java macroPass2
sagar-ravan@Sagar-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

M
DT
--
M
OV
E
#3,
#1
ADD #3,='1'
MOVER #3,#2
ADD
#3,='
5'
MEN
D
MO
VER
#3,#1
MOVER #4,#2
ADD #3,='15'
AD
D
#4,
='1
0'
M
EN
D

Assignment No: 5

Page No.
Date

LEX Program

Aim: Design LEX program for 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 requisities: lex 110, lex 120, Lex 180 lex 0, Lex 160, 250

Objectives: 1) To understand Lex concept
2) To implement Lex program
3) To study about Lex and Java
4) To know important about lexical analyer.

Theory:

Lex stands for Lexical Analyzer Lex is tool for generating Scanner. Scanner are program that recognize lexical pattern in text these lexical pattern (or regular expression) are defined in a particular syntax A matched regular expression may have an associated action This action may also include returning a token when lex receives input in the form of a time and continue until a pattern is matched then lex perform the associated action (which may include returning a token). if on the lex and Cor Highly coupled

Name: UJJWAL LADE **DivRoll**: TE-B-06

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;
}
```

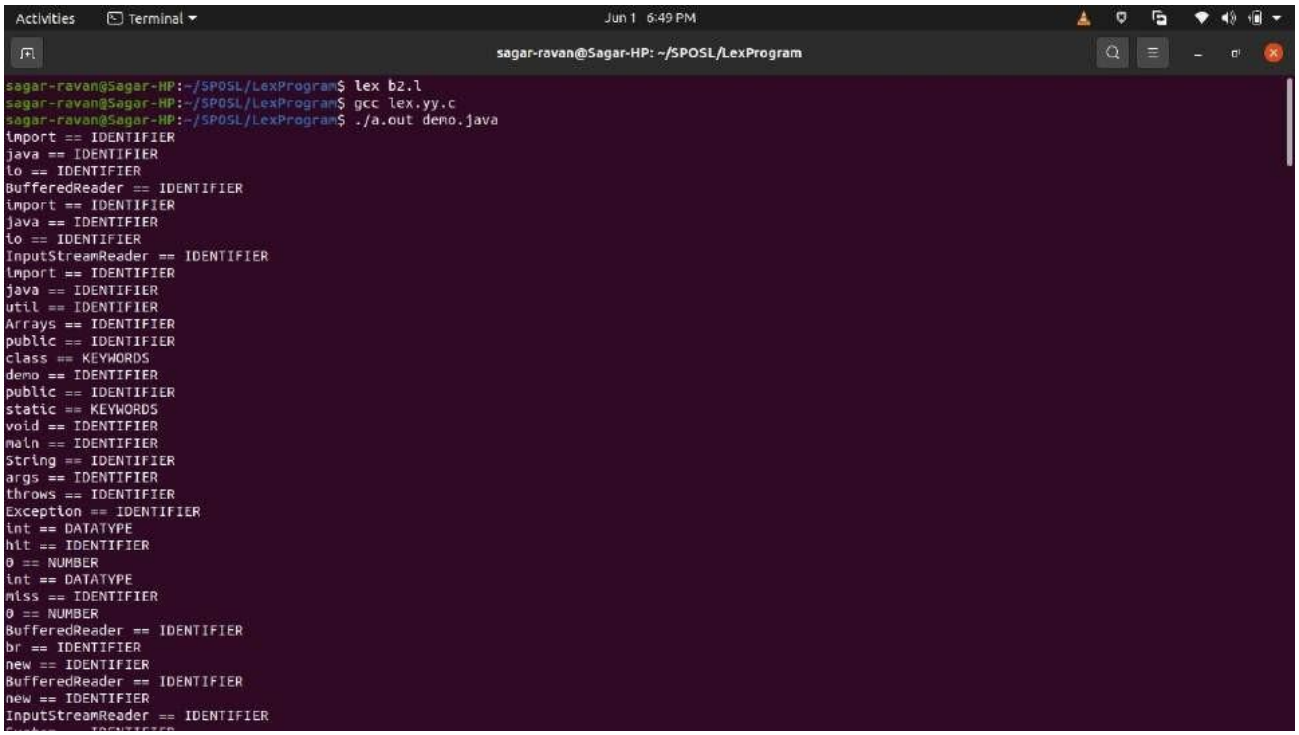
2. Demo.java Code:

```
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");
				}
				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;
					}
					else{
//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;
                }

                return -1;
            }


}
```

OUTPUT:



OUTPUT:

```
sagar-ravan@Sagar-HP:~/SPOSL/LexProgram$ lex b2.l
sagar-ravan@Sagar-HP:~/SPOSL/LexProgram$ gcc lex.yy.c
sagar-ravan@Sagar-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 ==
IDENTIFIE
R int ==
DATATYP
E i ==
IDENTIFIE
R
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 ==
DATATYP
E i ==
IDENTIFI

```
ER  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
sagar-ravan@Sagar-HP:~/SPOSL/LexProgram$
```

Assignment No : 06

- Aim: Design Lex program to count no. of words line and characters of given file.

- Problem statement : Write a program using Lex specification of compiler to count no. of word. line and character of given input file.

- pre requisites - LEX Basics.

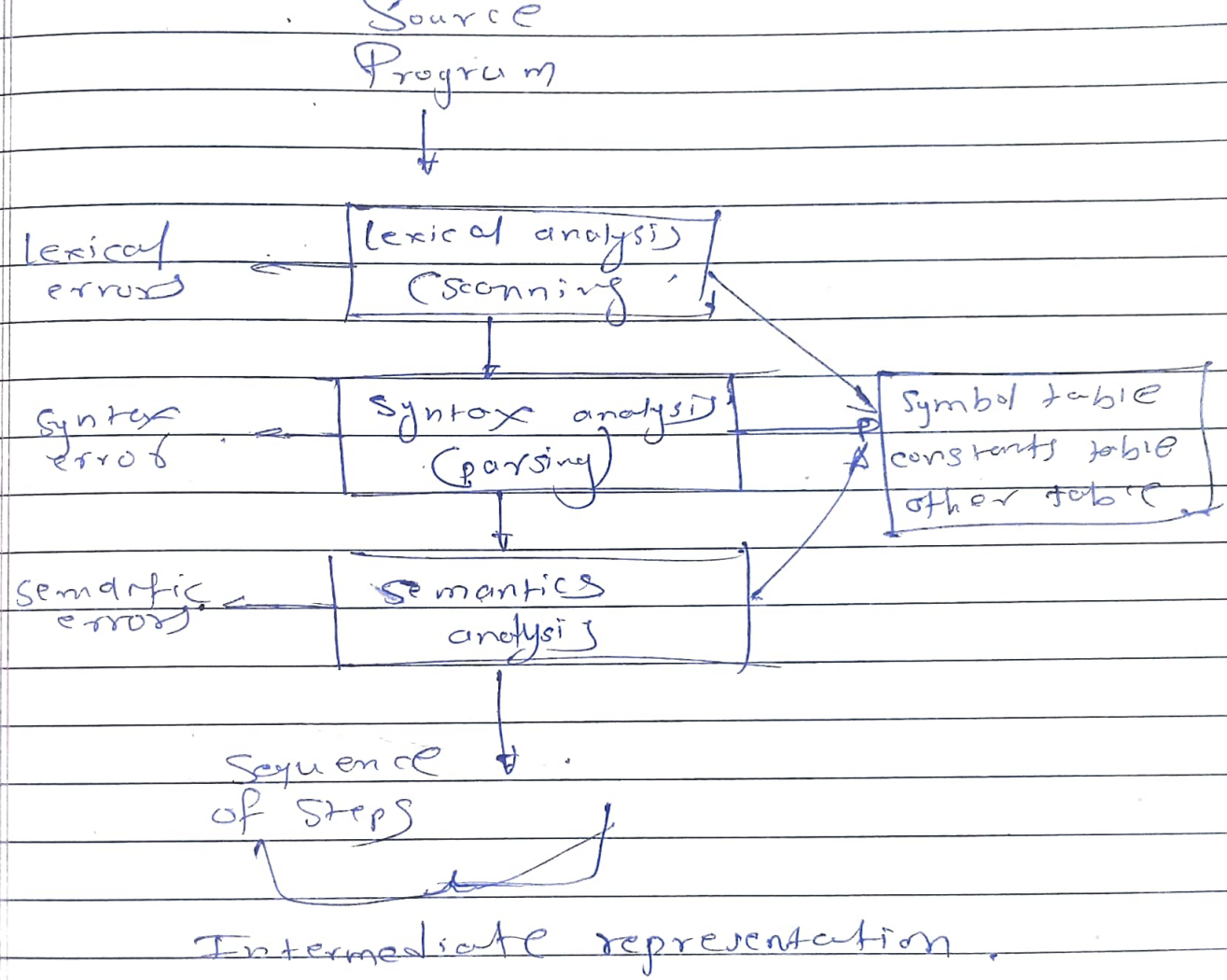
- Software Requirement.

Sr.No.	Facilities	Quantity
1	System	1
2	o/s	ubuntu kylin
3	s/w name	Lex tool.

- Objectives:
  i) To understand lex concepts.
  ii) To implement lex program for no. of count.
  iii) To study about lex and java.
  iv) To know important about Lexical analyzer

- How the i/p is entered:
  When the generated scanner is run, it analyzes its inputs looking for strings which match any of its patterns. if it finds more than one match it

it takes the one matching the most text it finds two more matches of the some length the rules listed first in the flex input file is choosen. once the match is determined the text and its length in the gobal integer yyleng.



Conclusion:

Thus we have studied lexial analyzer and implement an application for lexical analyzer to count total no. of words chard and line etc.

Name: UJJWAL LADE **DivRoll**: TE-B-06

Assignment No. 06 [LEX Program]

Problem Satement: 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:

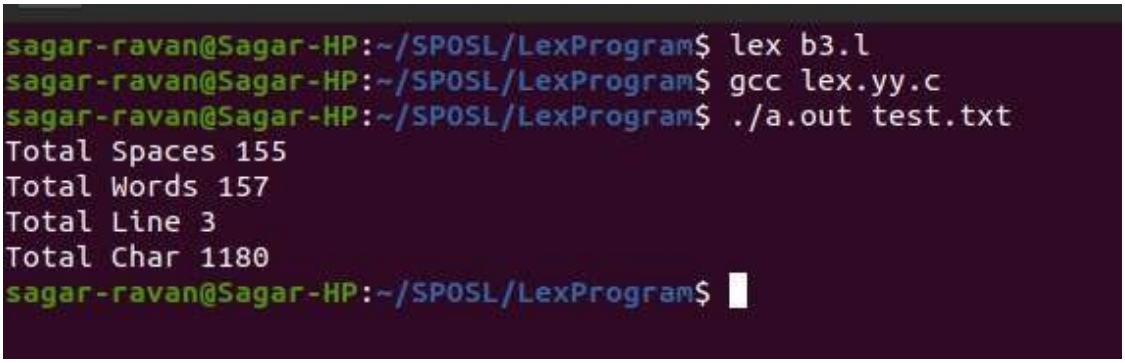
```
%{
int no_line=0;
int no_space=0;
int no_char=0;
int no_words=0;
#include<string.h>
%}

%%
([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:



Assignment No - 07

Name - Ujjwal lade
TE B 06

Aim:- Design Lex and yacc program to validate type and Syntax of variable declaration on java.

Problem Statement:- Write a program using yacc specification to emplemented lexical analysis phase of compiler to validate type and Syntax of variable declaration in java.

Pre-requisites:-

Lex 110, Lex120, Lex 130, Lex140, Lex160, 250

Software Requirment:-

SRNO	facilities required	Quantity
1.	System	1
2.	O/S	Ubonta kylin
3.	S/w name	Flt x YACC

Theory:-

YACC is a computer program for the UNIX operating System developed by step it is look ahead left to right pass of compila that this to make syntactin sense of source code, specific all LA R parser based on an analysis grammer written in notation similar to Backus. Nor form.

Lexical analysis for Yacc"

the uses must supply a lexical analysis to read the stream and communicate sokin to the parse the lexical analysis is an unlegs valued fonction called yylex. The relevant protion of the lexial analyze might look like

```
YYlex C; {
extern int yylex;
 int C;
  C= getchar (c)
  ... Soit in (c) {
   .... case '0';
   .
   ... case 9
   .
      YYlval E= (-'0'),
      return [DIGIT];
      ....;
```

APPLICATION

YAC is used to generate parses are an integral post of compilet

CONCLUSION

Thus, we have studied lexical analyze Synlex and emplement Lex and yac application for Syntax analyze to validate the given infix expression.

Name UJJWAL LADE **DivRoll**: TE-B-06

Assignment No. 07 [LEX Program]

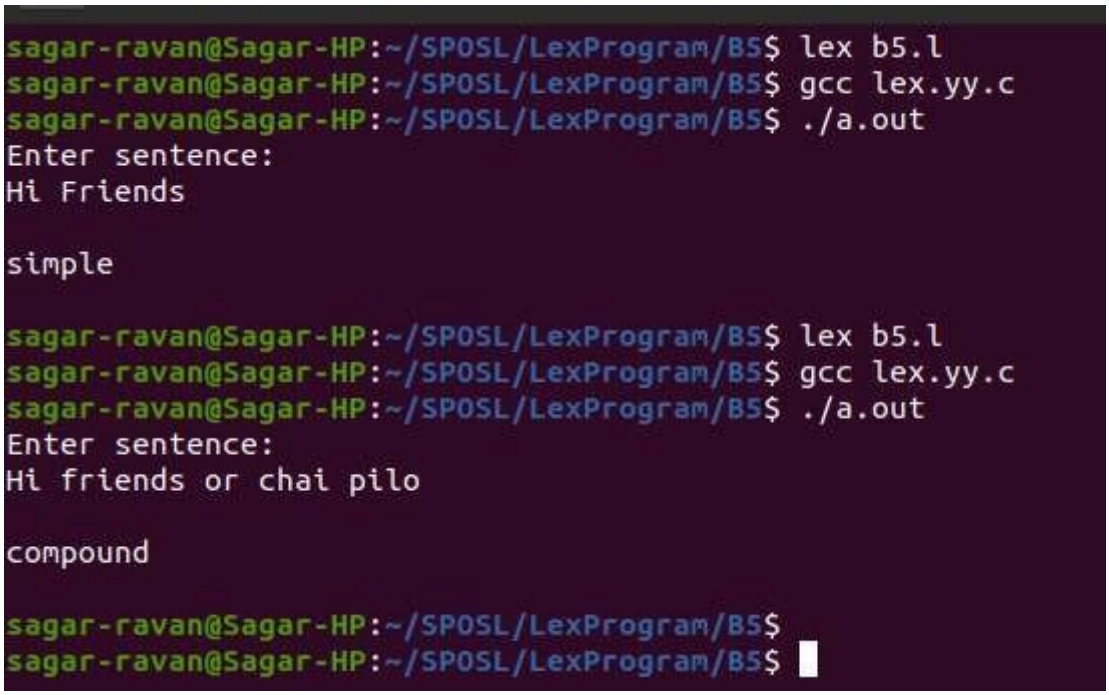
Problem Satement: Write a program using YACC specifications to implement syntax analysis phase of compiler to recognize simple and compound sentences given in input file

---

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");
   }
   else{
      printf("simple\n\n");
   }
   return 0;
}
```

OUTPUT


sagar-ravan@Sagar-HP:~/SPOSL/LexProgram/B5$ lex b5.l
sagar-ravan@Sagar-HP:~/SPOSL/LexProgram/B5$ gcc lex.yy.c
sagar-ravan@Sagar-HP:~/SPOSL/LexProgram/B5$ ./a.out
Enter sentence:
Hi Friends
simple
sagar-ravan@Sagar-HP:~/SPOSL/LexProgram/B5$ lex b5.l
sagar-ravan@Sagar-HP:~/SPOSL/LexProgram/B5$ gcc lex.yy.c
sagar-ravan@Sagar-HP:~/SPOSL/LexProgram/B5$ ./a.out
Enter sentence:
Hi friends or chai pilo
compound
sagar-ravan@Sagar-HP:~/SPOSL/LexProgram/B5$
sagar-ravan@Sagar-HP:~/SPOSL/LexProgram/B5$

Name- Vaibhavi Jade
TE B 06

Assignment No - 08

Aim :- Implement job Schedulling algorithm
i) FcFs
ii) Shortest Job First
iii) Priority.
iv) Round Robin.

Problem Statement :- Write a java program to implement following scheduling algorithm FcFs, SJF, Priority, Round Robin

Theory :-

① **FcFs [First come first serve] :-**
This is the simplest CPU scheduling algorithm the process that request the CPU first is the one to which is it allocated first
a. Implementation
b. Shortest Job first
c. Priority based scheduling
d. Round robin Scheduling

② **Shortest Job first :-**
This algorithm associatis with it the length of the next CPU Burst when the CPU is available it is assigned to that job with the smallest CPU Burst

③ **Priority Based scheduling**
Priority scheduling is non-preemptive algorithm and one of the most common is Soon

implementation.

④ Round Robin algorithm :-

Round robin is CPU scheduling algorithm where low process is assignid fixed time slot in cyclic way.

Conclusion:-

Thus, we have studied the emplemintation of sheduling algorithm fcfs, SJF, Priority, Round Robin

Name: UJJWAL LADE **DivRoll**: TE-B-06

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:

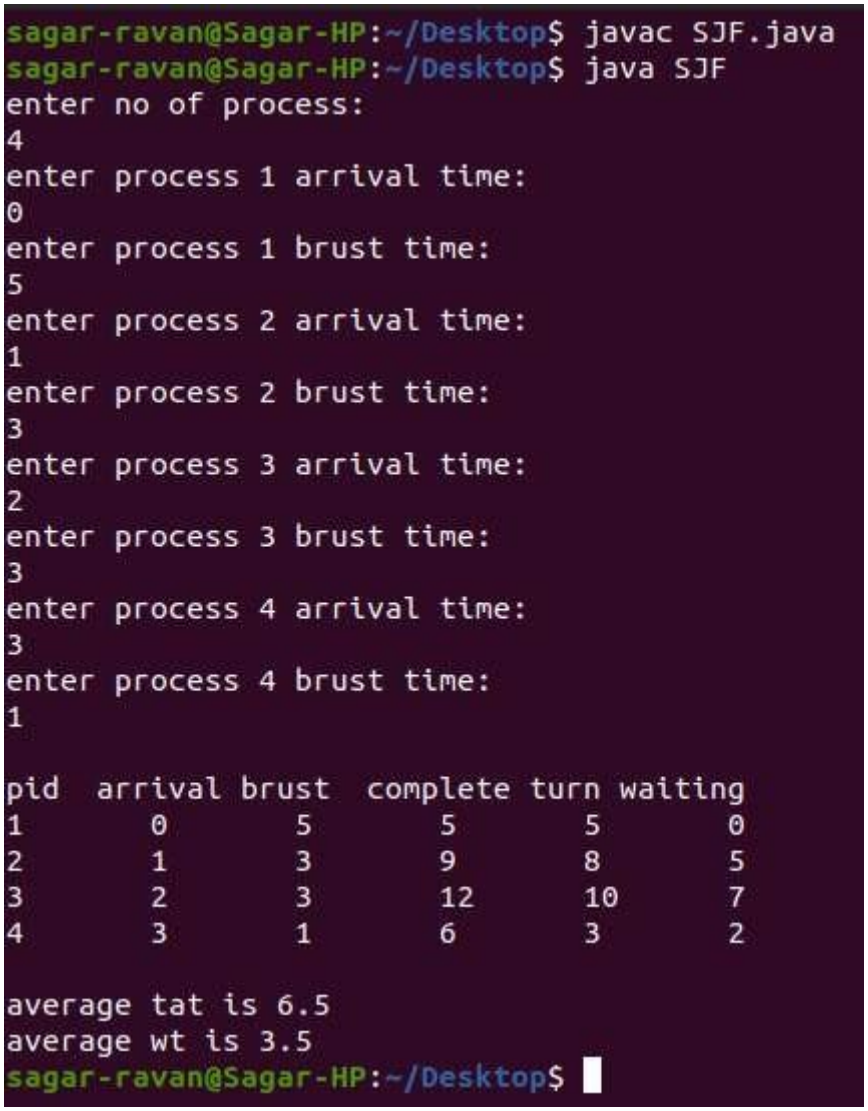
2. Shrtest Job First Program:

```
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.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();
        }
}
```

SJF OUTPUT:



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]<pp[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;
```

```
awt=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:

```
sagar-ravan@Sagar-HP:~/Desktop$ javac Priority.java
sagar-ravan@Sagar-HP:~/Desktop$ java Priority
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 : 16sagar-ravan@Sagar-HP:~/Desktop$
```

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+bt[k];
}
while(s
um!=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+=tat[j];
}
System.out.println("average waiting time"+(avg_wt/n)+"\n Average turn around
time"+(avg_tat/n));
}
}
```

Round Robin OUTPUT:

```
sagar-ravan@Sagar-HP:~/Desktop$ java RoundR
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
sagar-ravan@Sagar-HP:~/Desktop$
```

TE B' 06

Assignment No - 09

Aim :- Bankers algorithm for deadlock detection and avoidance.

Problem Statement :- Write java program to implementation Banker's algorithm

Operating System:-

Banker's algorithm:-

The banker algorithm is resource allocation and deadlock avoidance algorithm that test for safety by simulating the allocation for predetermined maximum possible amount of all resource then make an s-star check for possible activities

Following data structure are used to implement.

let 'n' be the number of process in the system is 'm' be the No. of resource type.

Available :-

It is a 1-array of size 'm' in dicating then of available resource of each type.

Max :-

It is 2nd array of size n+m that define the maximun demand of each process in a system.

Need :-

It is 2d array of size n*m that indicate the remaing resources need of each process.

Safety :-

The algorithm for finding out whose or not a system is in a stat can be described.

Conclusion :-

Thus, we have stoded a java program to implementation Bankers algorithm.

Name: UJJWAL LADE **DivRoll**: TE-B-06

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:

```
sagar-pavan@sagar-hp:~/Desktop$ java Bankers
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
```

Name - Ujjwal Jade
TE B 063

Assignment No - 10

Aim:- Implement UNIX System can like for process management.

Problem Statement:- To write a program to implement UNIX system calls like for process managment

Pre-requisitis :- 1. Explain concept of system call 2. Explain State diagram working of new process.

Software requirment :-
ubuntu kylin, ctuxbo '0x ycc

Objective:-
1. To understand UNIX System call
2. To understand concept of process managment
3. Implementation of some System call of as.

Theory:-

System call:

When a program is user mode require as to Ram or hardware resource. There is do via Something called a System call.

Generally System call are made by the user level process in the situation.

Kernal mode:-

When CPU is kernal mode the code is being executed can be aon any memory addres and ang hardware resource.

System call Basics:-

Since System calls or function we used need to include the proper header fetes.

Unix System calls:-

1. Ps Command.
2. Cask Command
3. Join Command.
4. Excel() command.
5. Wait ()

Conclusion:-

Thus, the process System Call program and studied various, System call.

Name: UJJWAL LADE **DivRoll**: TE-B-06

Assignment No. 10 [UNIX System Calls]

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

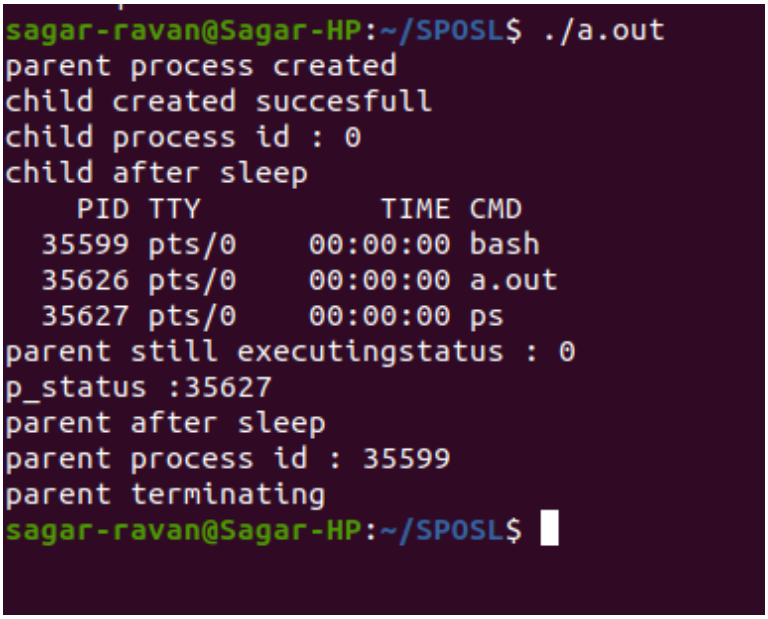
---

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:


sagar-ravan@Sagar-HP:~/SPOSL$ ./a.out
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
sagar-ravan@Sagar-HP:~/SPOSL$

Assignment No -12

Page Replacement algorithm

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

Problem statement :- To write a java program to implement LRU and optimal algorithm for page replacement.

Pre - requsitu :-

1. Explain the concept of virtual memory.
2. Define page replacement algorithm LRU optimal.
3. Explain add" translation in paging system.
4. Explain Belady's Anomly.

Theory :-

Whenever there is page refernc for which the page needed in memory that unit is called page fault or page futex or page fator situation.

There are seval algorithm to achieve
1. last recently used [LRU]
2. Optimal

1] LRU : page replacement

The main DB .IC IKO and optimal page replacement is that the FIFO algorithm uses the time when the page was brought into memory.

2] Optimal Page replacement :-

The algorithm has lowest page fault rate of all algorithm. Thus algorithm State that. Replace the page which will not be used for longest period time i.e future knowledge of reference string is required.

Conclusion :-

Thus we have studied to write a java program to implement LRU and optimal algorithm for page replacement.

Name: UJJWAL LADE **DivRoll**: TE-B-06

Assignment No. 12

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

---

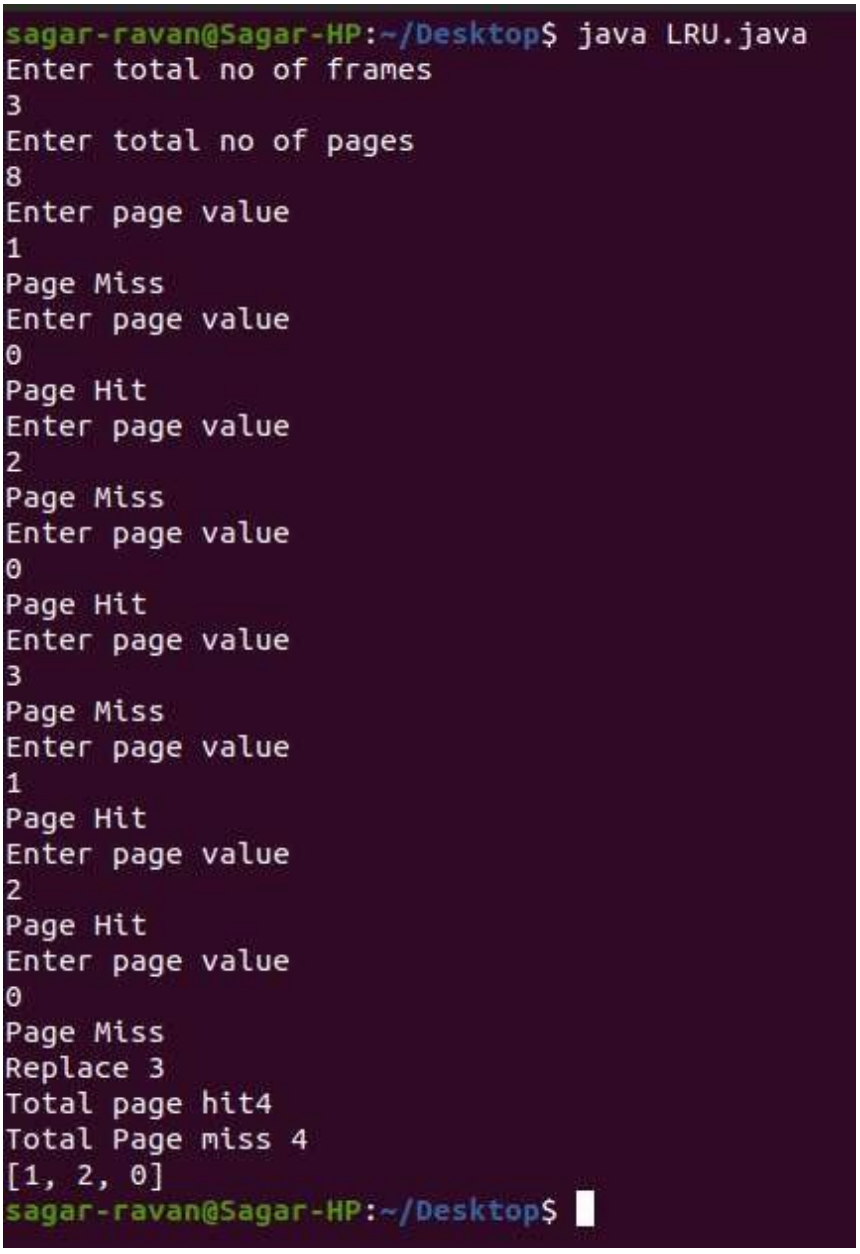
1. LRU (Last Recently Used) Program:

```
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");
				}
				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;
                                        }
                                        else{
//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;
                }

                return -1;
        }


}
```

OUTPUT:



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])
 {
 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: