PASS-1 ASSEMBLER :

MAIN PROGRAM:

```
import java.io.*;
class P1
{
public static void main(String ar[])throws IOException
{
BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
int i;
String a[][]={{"","START","101",""},
{"","MOVER","BREG","ONE"},
{"AGAIN","MULT","BREG","TERM"},
{"","MOVER","CREG","TERM"},
{"","ADD","CREG","N"},
{"","MOVEM","CREG","TERM"},
{"N","DS","2",""},
{"RESULT","DS","2",""},
{"ONE","DC","1",""},
{"TERM","DS","1",""},
{"","END","",""}};
int lc=Integer.parseInt(a[0][2]);
String st[][]=new String[5][2];
int cnt=0,l;
for (i=1;i<11;i++)
{
if (a[i][0]!="")
{
st [cnt][0]=a[i][0];
st[cnt][1]=Integer.toString(lc);
cnt++;
if(a[i][1]=="DS")
{
int d=Integer.parseInt(a[i][2]);
lc=lc+d;
}
else
{
lc++;
}
}
else
{
lc++;
}
}
System.out.print("***SYMBOL TABLE****\n");
System.out.println("_____________________");
```

```
for(i=0;i<5;i++)
{
for(cnt=0;cnt<2;cnt++)
{
System.out.print(st[i][cnt]+"\t");
}
System.out.println();
}
String
inst[]={"STOP","ADD","SUB","MULT","MOVER","MOVEM","COMP","BC
","DIV","READ","P
RINT"};
String reg[]={"NULL","AREG","BREG","CREG","DREG"};
int op[][]=new int[12][3];
int j,k,p=1,cnt1=0;
for(i=1;i<11;i++)
{
for(j=0;j<11;j++)
{
if(a[i][1].equalsIgnoreCase(inst[j]))
{
op[cnt1][0]=j;
}
else
if(a[i][1].equalsIgnoreCase("DS"))
{
p=Integer.parseInt(a[i][2]);
}
else if(a[i][1].equalsIgnoreCase("DC"))
{
op[cnt1][2]=Integer.parseInt(a[i][2]);
}
}
for(k=0;k<5;k++)
{
if(a[i][2].equalsIgnoreCase(reg[k]))
{
op[cnt1][1]=k;
}
}
for(l=0;l<5;l++)
{
if(a[i][3].equalsIgnoreCase(st[l][0]))
{
int mn=Integer.parseInt(st[l][1]);
op[cnt1][2]=mn;
}
}
cnt1=cnt1+p;
}
System.out.println("\n *****OUTPUT*****\n");
System.out.println("**********MOT TABLE**********");
int dlc=Integer.parseInt(a[0][2]);
for(i=0;i<12;i++)
{
System.out.print(dlc+++"\t");
for(j=0;j<3;j++)
{
System.out.print(" "+op[i][j]+" ");
}
System.out.println();
}
System.out.println("");
}
}
```

OUTPUT :

```
***SYMBOL TABLE****
_____________________
AGAIN 102
N 106
RESULT 108
ONE 110
TERM 111
*****OUTPUT*****
**********MOT TABLE**********
101 4 2 110
102 3 2 111
103 4 3 111
104 1 3 106
105 5 3 111
106 0 0 0
107 0 0 0
108 0 0 0
109 0 0 0
110 0 0 1
111 0 0 0
112 0 0 0
```

```
/*
Problem Statement: Design suitable data structures and implement pass-I of a two-pass assembler
for pseudo-
machine in Java using object oriented feature. Implementation should consist of a few
instructions from each category and few assembler directives.
*/
import java.io.*;
class SymTab
{
    public static void main(String args[])throws Exception
    {
        FileReader FP=new FileReader(args[0]);
        BufferedReader bufferedReader = new BufferedReader(FP);

        String line=null;
        int line_count=0,LC=0,symTabLine=0,opTabLine=0,litTabLine=0,poolTabLine=0;

         //Data Structures
         final int MAX=100;
         String SymbolTab[][]=new String[MAX][3];
         String OpTab[][]=new String[MAX][3];
         String LitTab[][]=new String[MAX][2];
         int PoolTab[]=new int[MAX];
         int litTabAddress=0;
/*---------------------------------------------------------------------------------------------------*/


System.out.println("_______________________________________________________");
          while((line = bufferedReader.readLine()) != null)
           {
             String[] tokens = line.split("\t");
            if(line_count==0)
            {
                LC=Integer.parseInt(tokens[2]);
                //set LC to operand of START
                for(int i=0;i<tokens.length;i++)               //for printing the input
program
                    System.out.print(tokens[i]+"\t");
                System.out.println("");
            }
            else
            {
                 for(int i=0;i<tokens.length;i++) //for printing the input program
                    System.out.print(tokens[i]+"\t");
                 System.out.println("");
                if(!tokens[0].equals(""))
                {
                    //Inserting into Symbol Table
                    SymbolTab[symTabLine][0]=tokens[0];
                    SymbolTab[symTabLine][1]=Integer.toString(LC);
                    SymbolTab[symTabLine][2]=Integer.toString(1);
                    symTabLine++;
                }
                else
if(tokens[1].equalsIgnoreCase("DS")||tokens[1].equalsIgnoreCase("DC"))
                {
                    //Entry into symbol table for declarative statements
                    SymbolTab[symTabLine][0]=tokens[0];
                    SymbolTab[symTabLine][1]=Integer.toString(LC);
                    SymbolTab[symTabLine][2]=Integer.toString(1);
                    symTabLine++;
                }

                if(tokens.length==3 && tokens[2].charAt(0)=='=')
                {
                    //Entry of literals into literal table
                    LitTab[litTabLine][0]=tokens[2];
                    LitTab[litTabLine][1]=Integer.toString(LC);
                    litTabLine++;
                }

                else if(tokens[1]!=null)
                {
                        //Entry of Mnemonic in opcode table
                    OpTab[opTabLine][0]=tokens[1];


    if(tokens[1].equalsIgnoreCase("START")||tokens[1].equalsIgnoreCase("END")||tokens[1].eq
ualsIgnoreCase("ORIGIN")||tokens[1].equalsIgnoreCase("EQU")||tokens[1].equalsIgnoreCase("LTOR
G"))        //if Assembler Directive
                    {
                        OpTab[opTabLine][1]="AD";
                        OpTab[opTabLine][2]="R11";

                    }
                    else
if(tokens[1].equalsIgnoreCase("DS")||tokens[1].equalsIgnoreCase("DC"))
                    {
                        OpTab[opTabLine][1]="DL";
                        OpTab[opTabLine][2]="R7";

                    }
                    else
                    {
                        OpTab[opTabLine][1]="IS";
```

```
                                        OpTab[opTabLine][2]="(04,1)";
                                }
                        opTabLine++;
                        }
            }
            line_count++;
            LC++;
          }


System.out.println("________________________________________________________");

                //print symbol table
                System.out.println("\n\n            SYMBOL TABLE              ");
                System.out.println("--------------------------");
                System.out.println("SYMBOL\tADDRESS\tLENGTH");
                System.out.println("--------------------------");
                for(int i=0;i<symTabLine;i++)

System.out.println(SymbolTab[i][0]+"\t"+SymbolTab[i][1]+"\t"+SymbolTab[i][2]);
                System.out.println("--------------------------");


                //print opcode table
                System.out.println("\n\n            OPCODE TABLE              ");
                System.out.println("----------------------------");
                System.out.println("MNEMONIC\tCLASS\tINFO");
                System.out.println("----------------------------");
                for(int i=0;i<opTabLine;i++)

System.out.println(OpTab[i][0]+"\t\t"+OpTab[i][1]+"\t"+OpTab[i][2]);
                System.out.println("----------------------------");

                //print literal table
                System.out.println("\n\n   LITERAL TABLE                      ");
                System.out.println("-----------------");
                System.out.println("LITERAL\tADDRESS");
                System.out.println("-----------------");
                for(int i=0;i<litTabLine;i++)
                        System.out.println(LitTab[i][0]+"\t"+LitTab[i][1]);
                System.out.println("------------------");


                //intialization of POOLTAB
                for(int i=0;i<litTabLine;i++)
                {
                        if(LitTab[i][0]!=null && LitTab[i+1][0]!=null ) //if literals are present
                        {
```

```
                                        if(i==0)
                                        {
                                                PoolTab[poolTabLine]=i+1;
                                                poolTabLine++;
                                        }
                                        else
if(Integer.parseInt(LitTab[i][1])<(Integer.parseInt(LitTab[i+1][1]))-1)
                                        {
                                                PoolTab[poolTabLine]=i+2;
                                                poolTabLine++;
                                        }
                                }
                        }
                        //print pool table
                        System.out.println("\n\n   POOL TABLE                 ");
                        System.out.println("-----------------");
                        System.out.println("LITERAL NUMBER");
                        System.out.println("-----------------");
                        for(int i=0;i<poolTabLine;i++)
                                System.out.println(PoolTab[i]);
                        System.out.println("------------------");


                  // Always close files.
                  bufferedReader.close();
        }
}

/*
OUTPUT-
neha@neha-1011PX:~/neha_SPOS$ javac SymTab.java
neha@neha-1011PX:~/neha_SPOS$ java SymTab input.txt
____________________________________________________________
        START   100
        READ    A
LABLE   MOVER A,B
        LTORG
                ='5'
                ='1'
                ='6'
                ='7'
        MOVEM           A,B
        LTORG
                ='2'
LOOP    READ    B
A       DS      1
B       DC      '1'
                ='1'
```

```
          END
_____________________________________________________________________________


          SYMBOL TABLE
--------------------------
SYMBOL              ADDRESS             LENGTH
--------------------------
LABLE     102       1
LOOP      111       1
A         112       1
B         113       1
--------------------------


          OPCODE TABLE
----------------------------
MNEMONIC            CLASS     INFO
----------------------------
READ                IS        (04,1)
MOVER               IS        (04,1)
LTORG               AD        R11
MOVEM                         IS        (04,1)
LTORG               AD        R11
READ                IS        (04,1)
DS                  DL        R7
DC                  DL        R7
END                 AD        R11
----------------------------


  LITERAL TABLE
-----------------
LITERALADDRESS
-----------------
='5'      104
='1'      105
='6'      106
='7'      107
='2'      110
='1'      114
------------------


  POOL TABLE
-----------------
LITERAL NUMBER
-----------------
```

```
1
5
6
------------------


*/
```

```
/*
Problem Statement: Design suitable data structures and implement pass-I of a two-pass assembler
for pseudo-
machine in Java using object oriented feature. Implementation should consist of a few
instructions from each category and few assembler directives.
*/
import java.io.*;
class SymTab
{
        public static void main(String args[])throws Exception
        {
                FileReader FP=new FileReader(args[0]);
                BufferedReader bufferedReader = new BufferedReader(FP);

                String line=null;
                int line_count=0,LC=0,symTabLine=0,opTabLine=0,litTabLine=0,poolTabLine=0;

                 //Data Structures
                 final int MAX=100;
                 String SymbolTab[][]=new String[MAX][3];
                 String OpTab[][]=new String[MAX][3];
                 String LitTab[][]=new String[MAX][2];
                 int PoolTab[]=new int[MAX];
                 int litTabAddress=0;
/*---------------------------------------------------------------------------------------------------*/


System.out.println("________________________________________________________");
                  while((line = bufferedReader.readLine()) != null)
                   {
                         String[] tokens = line.split("\t");
                        if(line_count==0)
                        {
                                LC=Integer.parseInt(tokens[2]);
                                //set LC to operand of START
                                for(int i=0;i<tokens.length;i++)          //for printing the input
program
                                        System.out.print(tokens[i]+"\t");
                                System.out.println("");
                        }
                        else
                        {
                                 for(int i=0;i<tokens.length;i++) //for printing the input program
                                        System.out.print(tokens[i]+"\t");
                                 System.out.println("");
```

```
					if(!tokens[0].equals(""))
					{

						//Inserting into Symbol Table
						SymbolTab[symTabLine][0]=tokens[0];
						SymbolTab[symTabLine][1]=Integer.toString(LC);
						SymbolTab[symTabLine][2]=Integer.toString(1);
						symTabLine++;
					}
					else
if(tokens[1].equalsIgnoreCase("DS")||tokens[1].equalsIgnoreCase("DC"))
					{
						//Entry into symbol table for declarative statements
						SymbolTab[symTabLine][0]=tokens[0];
						SymbolTab[symTabLine][1]=Integer.toString(LC);
						SymbolTab[symTabLine][2]=Integer.toString(1);
						symTabLine++;
					}

					if(tokens.length==3 && tokens[2].charAt(0)=='=')
					{
						//Entry of literals into literal table
						LitTab[litTabLine][0]=tokens[2];
						LitTab[litTabLine][1]=Integer.toString(LC);
						litTabLine++;
					}

					else if(tokens[1]!=null)
					{
							//Entry of Mnemonic in opcode table
						OpTab[opTabLine][0]=tokens[1];


	if(tokens[1].equalsIgnoreCase("START")||tokens[1].equalsIgnoreCase("END")||tokens[1].eq
ualsIgnoreCase("ORIGIN")||tokens[1].equalsIgnoreCase("EQU")||tokens[1].equalsIgnoreCase("LTOR
G"))			//if Assembler Directive
						{
							OpTab[opTabLine][1]="AD";
							OpTab[opTabLine][2]="R11";

						}
						else
if(tokens[1].equalsIgnoreCase("DS")||tokens[1].equalsIgnoreCase("DC"))
						{
							OpTab[opTabLine][1]="DL";
```

```
                    OpTab[opTabLine][2]="R7";

                }
                else
                {
                    OpTab[opTabLine][1]="IS";
                    OpTab[opTabLine][2]="(04,1)";
                }
            opTabLine++;
            }
      }
      line_count++;
      LC++;
     }


System.out.println("______________________________________________________");

        //print symbol table
        System.out.println("\n\n            SYMBOL TABLE              ");
        System.out.println("-------------------------");
        System.out.println("SYMBOL\tADDRESS\tLENGTH");
        System.out.println("-------------------------");
        for(int i=0;i<symTabLine;i++)

System.out.println(SymbolTab[i][0]+"\t"+SymbolTab[i][1]+"\t"+SymbolTab[i][2]);
        System.out.println("-------------------------");


        //print opcode table
        System.out.println("\n\n            OPCODE TABLE              ");
        System.out.println("---------------------------");
        System.out.println("MNEMONIC\tCLASS\tINFO");
        System.out.println("---------------------------");
        for(int i=0;i<opTabLine;i++)

System.out.println(OpTab[i][0]+"\t\t"+OpTab[i][1]+"\t"+OpTab[i][2]);
        System.out.println("---------------------------");

        //print literal table
        System.out.println("\n\n   LITERAL TABLE                        ");
        System.out.println("-----------------");
        System.out.println("LITERAL\tADDRESS");
        System.out.println("-----------------");
        for(int i=0;i<litTabLine;i++)
```

```
                System.out.println(LitTab[i][0]+"\t"+LitTab[i][1]);
            System.out.println("------------------");


            //intialization of POOLTAB
            for(int i=0;i<litTabLine;i++)
            {
                if(LitTab[i][0]!=null && LitTab[i+1][0]!=null ) //if literals are present
                {
                    if(i==0)
                    {
                        PoolTab[poolTabLine]=i+1;
                        poolTabLine++;
                    }
                    else
if(Integer.parseInt(LitTab[i][1])<(Integer.parseInt(LitTab[i+1][1]))-1)
                    {
                        PoolTab[poolTabLine]=i+2;
                        poolTabLine++;
                    }
                }
            }
            //print pool table
            System.out.println("\n\n   POOL TABLE                ");
            System.out.println("-----------------");
            System.out.println("LITERAL NUMBER");
            System.out.println("-----------------");
            for(int i=0;i<poolTabLine;i++)
                System.out.println(PoolTab[i]);
            System.out.println("------------------");


        // Always close files.
        bufferedReader.close();
    }
}

/*
OUTPUT-
neha@neha-1011PX:~/neha_SPOS$ javac SymTab.java
neha@neha-1011PX:~/neha_SPOS$ java SymTab input.txt
______________________________________________________
    START   100
    READ    A
LABLE   MOVER A,B
```

```
          LTORG
                    ='5'
                    ='1'
                    ='6'
                    ='7'
          MOVEM               A,B
          LTORG
                    ='2'
LOOP      READ      B
A         DS        1
B         DC        '1'
                    ='1'
          END
__________________________________________________________


          SYMBOL TABLE
--------------------------
SYMBOL              ADDRESS             LENGTH
--------------------------
LABLE     102       1
LOOP      111       1
A         112       1
B         113       1
--------------------------


          OPCODE TABLE
----------------------------
MNEMONIC            CLASS     INFO
----------------------------
READ                IS        (04,1)
MOVER               IS        (04,1)
LTORG               AD        R11
MOVEM                         IS        (04,1)
LTORG               AD        R11
READ                IS        (04,1)
DS                  DL        R7
DC                  DL        R7
END                 AD        R11
----------------------------


  LITERAL TABLE
-----------------
```

```
LITERALADDRESS
-----------------
='5'	104
='1'	105
='6'	106
='7'	107
='2'	110
='1'	114
------------------


   POOL TABLE
-----------------
LITERAL NUMBER
-----------------
1
5
6
------------------


*/
```

```
/*
Problem Statement: Implement Pass-II of two pass assembler for pseudo-machine in Java using
object oriented
features. The output of assignment-1 (intermediate file and symbol table) should be
input for this assignment.
*/
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashMap;

public class Pass2 {
        public static void main(String[] Args) throws IOException{
                BufferedReader b1 = new BufferedReader(new FileReader("intermediate.txt"));
            BufferedReader b2 = new BufferedReader(new FileReader("symtab.txt"));
            BufferedReader b3 = new BufferedReader(new FileReader("littab.txt"));
            FileWriter f1 = new FileWriter("Pass2.txt");
            HashMap<Integer, String> symSymbol = new HashMap<Integer, String>();
            HashMap<Integer, String> litSymbol = new HashMap<Integer, String>();
            HashMap<Integer, String> litAddr = new HashMap<Integer, String>();
            String s;
            int symtabPointer=1,littabPointer=1,offset;
            while((s=b2.readLine())!=null){
                String word[]=s.split("\t\t\t");
                symSymbol.put(symtabPointer++,word[1]);
            }
            while((s=b3.readLine())!=null){
                String word[]=s.split("\t\t");
                litSymbol.put(littabPointer,word[0]);
                litAddr.put(littabPointer++,word[1]);
            }
            while((s=b1.readLine())!=null){
                if(s.substring(1,6).compareToIgnoreCase("IS,00")==0){
                        f1.write("+ 00 0 000\n");
                }
                else if(s.substring(1,3).compareToIgnoreCase("IS")==0){
                        f1.write("+ "+s.substring(4,6)+" ");
                        if(s.charAt(9)==')'){
                                f1.write(s.charAt(8)+" ");
                                offset=3;
                        }
                        else{
```

```
                f1.write("0 ");
                offset=0;
            }
            if(s.charAt(8+offset)=='S')

f1.write(symSymbol.get(Integer.parseInt(s.substring(10+offset,s.length()-1)))+"\n");
            else

f1.write(litAddr.get(Integer.parseInt(s.substring(10+offset,s.length()-1)))+"\n");
        }
        else if(s.substring(1,6).compareToIgnoreCase("DL,01")==0){
            String s1=s.substring(10,s.length()-1),s2="";
            for(int i=0;i<3-s1.length();i++)
                s2+="0";
            s2+=s1;
            f1.write("+ 00 0 "+s2+"\n");
        }
        else{
            f1.write("\n");
        }
     }
     f1.close();
     b1.close();
     b2.close();
     b3.close();
    }
}

/*
OUTPUT:
neha@neha-1011PX:~/Desktop/neha_SPOS/Turn1/A2$ javac Pass2.java
neha@neha-1011PX:~/Desktop/neha_SPOS/Turn1/A2$ java Pass2
neha@neha-1011PX:~/Desktop/neha_SPOS/Turn1/A2$ cat Pass2.txt

intermediate code -
(AD,01)(C,200)
(IS,04)(1)(L,1)
(IS,05)(1)(S,1)
(IS,04)(1)(S,1)
(IS,04)(3)(S,3)
(IS,01)(3)(L,2)
(IS,07)(6)(S,4)
(DL,01)(C,5)
(DL,01)(C,1)
(IS,02)(1)(L,3)
```

```
(IS,07)(1)(S,5)
(IS,00)
(AD,03)(S,2)+2
(IS,03)(3)(S,3)
(AD,03)(S,6)+1
(DL,02)(C,1)
(DL,02)(C,1)
(AD,02)
(DL,01)(C,1)

Symbol Table --
A                       211                     1
LOOP                    202                     1
B                       212                     1
NEXT                    208                     1
BACK                    202                     1
LAST                    210                     1

literal table --
5               206
1               207
1               213


machine code --

+ 04 1 206
+ 05 1 211
+ 04 1 211
+ 04 3 212
+ 01 3 207
+ 07 6 208
+ 00 0 005
+ 00 0 001
+ 02 1 213
+ 07 1 202
+ 00 0 000
+ 03 3 212     */
```

PASS-1 MACROPROCESSOR :

MAIN PROGRAM:

```
import java.util.*;
import java.io.*;
class MACRO
{
static String mnt[][]=new String[5][3]; //assuming 5
macros in 1
program
static String ala[][]=new String[10][2]; //assuming 2
arguments in
each macro
static String mdt[][]=new String[20][1]; //assuming 4
LOC for each
macro
static int mntc=0,mdtc=0,alac=0;
public static void main(String args[])
{
pass1();
System.out.println("\n*********PASS-1
MACROPROCESSOR***********\n");
System.out.println("MACRO NAME TABLE (MNT)\n");
System.out.println("i macro loc\n");
display(mnt,mntc,3);
System.out.println("\n");
System.out.println("ARGUMENT LIST ARRAY(ALA) for
Pass1\n");
display(ala,alac,2);
System.out.println("\n");
System.out.println("MACRO DEFINITION TABLE (MDT)\n");
display(mdt,mdtc,1);
System.out.println("\n");
}
static void pass1()
{
int index=0,i;
String s,prev="",substring;
try
{
BufferedReader inp = new BufferedReader(new
FileReader("input.txt"));
File op = new File("pass1_output.txt");
if (!op.exists())
```

```
op.createNewFile();
BufferedWriter output = new BufferedWriter(new
FileWriter(op.getAbsoluteFile()));
while((s=inp.readLine())!=null)
{
if(s.equalsIgnoreCase("MACRO"))
{
prev=s;
for(;!(s=inp.readLine()).equalsIgnoreCase("MEND");mdt
c++,prev=s)
{
if(prev.equalsIgnoreCase("MACRO"))
{
StringTokenizer st=new StringTokenizer(s);
String str[]=new String[st.countTokens()];
for(i=0;i<str.length;i++)
str[i]=st.nextToken();
mnt[mntc][0]=(mntc+1)+""; //mnt formation
mnt[mntc][1]=str[0];
mnt[mntc++][2]=(++mdtc)+"";
st=new StringTokenizer(str[1],","); //tokenizing the
arguments
String string[]=new String[st.countTokens()];
for(i=0;i<string.length;i++)
{
string[i]=st.nextToken();
ala[alac][0]=alac+""; //ala table formation
index=string[i].indexOf("=");
if(index!=-1)
ala[alac++][1]=string[i].substring(0,index);
else
ala[alac++][1]=string[i];
}
}
else //automatically eliminates tagging of arguments
in definition
{ //mdt formation
index=s.indexOf("&");
substring=s.substring(index);
for(i=0;i<alac;i++)
if(ala[i][1].equals(substring))
s=s.replaceAll(substring,"#"+ala[i][0]);
}
mdt[mdtc-1][0]=s;
}
mdt[mdtc-1][0]=s;
}
else
{
output.write(s);
output.newLine();
}
}
output.close();
}
catch(FileNotFoundException ex)
{
System.out.println("UNABLE TO END FILE ");
}
catch(IOException e)
{
e.printStackTrace();
}
}
static void display(String a[][],int n,int m)
{
int i,j;
for(i=0;i<n;i++)
{
for(j=0;j<m;j++)
System.out.print(a[i][j]+" ");
System.out.println();
}
}
}
/* INPUT
START
MACRO
INCR &ARG3 &ARG2
ADD AREG &ARG1
MOVER BREG &ARG1
MEND
MACRO
PVG &ARG2 &ARG1
SUB AREG &ARG2
MOVER CREG & ARG1
```

```
MEND
INCR
DECR
DATA2
END
*/
```

/* OUTPUT

```
pvgcoen-3@pvgcoen3-ThinkCentre-M700:~/AA$ javac
MACRO.java
pvgcoen-3@pvgcoen3-ThinkCentre-M700:~/AA$ java MACRO
*********PASS-1 MACROPROCESSOR***********
MACRO NAME TABLE (MNT)
i macro loc
1 INCR 1
2 PVG 5
ARGUMENT LIST ARRAY(ALA) for Pass1
0 &ARG3
1 &ARG2
MACRO DEFINITION TABLE (MDT)
INCR &ARG3 &ARG2
ADD AREG &ARG1
MOVER BREG &ARG1
MEND
PVG &ARG2 &ARG1
SUB AREG #1
MOVER CREG & ARG1
MEND
*/
```

```
/*
Problem Statement: Design suitable data structures and implement pass-I of
a two-pass macro-processor using
OOP features in Java
*/
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"+(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:

neha@neha-1011PX:~/Desktop/neha_SPOS/Turn1/A3$ javac macroPass1.java
neha@neha-1011PX:~/Desktop/neha_SPOS/Turn1/A3$ java macroPass1

neha@neha-1011PX:~/Desktop/neha_SPOS/Turn1/A3$ cat intermediate.txt
M1 10,20,&b=CREG
M2 100,200,&u=AREG,&v=BREG

neha@neha-1011PX:~/Desktop/neha_SPOS/Turn1/A3$ cat mnt.txt
M1      2       2       1       1
M2      2       2       7       3
M3      2       0       13      4

neha@neha-1011PX:~/Desktop/neha_SPOS/Turn1/A3$ 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

neha@neha-1011PX:~/Desktop/neha_SPOS/Turn1/A3$ cat kpdt.txt
a       AREG
b       -
u       CREG
v       DREG

*/
```

```
//TWO PASS MACROPROCESSOR
import java.util.*;
import java.io.*;
class MntTuple {
String name;
int index;
MntTuple(String s, int i) {
name = s;
index = i;
}
public String toString() {
return("[" + name + ", " + index + "]");
}
}
class MacroProcessor {
static List<MntTuple> mnt;
static List<String> mdt;
static int mntc;
static int mdtc;
static int mdtp;
static BufferedReader input;
static List<List <String>> ala;
static Map<String, Integer> ala_macro_binding;
public static void main(String args[]) throws
Exception {
initializeTables();
System.out.println("===== PASS 1 =====\n");
pass1();
System.out.println("\n===== PASS 2 =====\n");
pass2();
}
static void pass1() throws Exception {
String s = new String();
input = new BufferedReader(new InputStreamReader(new
FileInputStream("input.txt")));
PrintWriter output = new PrintWriter(new
FileOutputStream("output_pass1.txt"), true);
while((s = input.readLine()) != null) {
if(s.equalsIgnoreCase("MACRO")) {
processMacroDefinition();
} else {
output.println(s);
}
```

```
}
System.out.println("ALA:");
showAla(1);
System.out.println("\nMNT:");
showMnt();
System.out.println("\nMDT:");
showMdt();
}
static void processMacroDefinition() throws Exception
{
String s = input.readLine();
String macro_name = s.substring(0, s.indexOf(" "));
mnt.add(new MntTuple(macro_name, mdtc));
mntc++;
pass1Ala(s);
StringTokenizer st = new StringTokenizer(s, " ,",
false);
String x = st.nextToken();
for(int i=x.length() ; i<12 ; i++) {
x += " ";
}
String token = new String();
int index;
token = st.nextToken();
x += token;
while(st.hasMoreTokens()) {
token = st.nextToken();
x += "," + token;
}
mdt.add(x);
mdtc++;
addIntoMdt(ala.size()-1);
}
static void pass1Ala(String s) {
StringTokenizer st = new StringTokenizer(s, " ,",
false);
String macro_name = st.nextToken();
List<String> l = new ArrayList<>();
int index;
while(st.hasMoreTokens()) {
String x = st.nextToken();
if((index = x.indexOf("=")) != -1) {
x = x.substring(0, index);
}
```

```
l.add(x);
}
ala.add(l);
ala_macro_binding.put(macro_name,
ala_macro_binding.size());
}
static void addIntoMdt(int ala_number) throws
Exception {
String temp = new String();
String s = new String();
List l = ala.get(ala_number);
boolean isFirst;
while(!s.equalsIgnoreCase("MEND")) {
isFirst = true;
s = input.readLine();
String line = new String();
StringTokenizer st = new StringTokenizer(s, " ,",
false);
temp = st.nextToken();
for(int i=temp.length() ; i<12 ; i++) {
temp += " ";
}
line += temp;
while(st.hasMoreTokens()) {
temp = st.nextToken();
if(temp.startsWith("&")) {
int x = l.indexOf(temp);
temp = ",#" + x;
isFirst = false;
} else if(!isFirst) {
temp = "," + temp;
}
line += temp;
}
mdt.add(line);
mdtc++;
}
}
static void showAla(int pass) throws Exception {
PrintWriter out = new PrintWriter(new
FileOutputStream("out_ala_pass" + pass + ".txt"),
true);
for(List l : ala) {
System.out.println(l);
```

```
out.println(l);
}
}
static void showMnt() throws Exception {
PrintWriter out = new PrintWriter(new
FileOutputStream("out_mnt.txt"), true);
for(MntTuple l : mnt) {
System.out.println(l);
out.println(l);
}
}
static void showMdt() throws Exception {
PrintWriter out = new PrintWriter(new
FileOutputStream("out_mdt.txt"), true);
for(String l : mdt) {
System.out.println(l);
out.println(l);
}
}
static void pass2() throws Exception {
input = new BufferedReader(new InputStreamReader(new
FileInputStream("output_pass1.txt")));
PrintWriter output = new PrintWriter(new
FileOutputStream("output_pass2.txt"), true);
String token = new String();
String s;
while((s = input.readLine()) != null) {
StringTokenizer st = new StringTokenizer(s, " ",
false);
while(st.hasMoreTokens()) {
token = st.nextToken();
if(st.countTokens() > 2) {
token = st.nextToken();
}
MntTuple x = null;
for(MntTuple m : mnt) {
if(m.name.equalsIgnoreCase(token)) {
x = m;
break;
}
}
if(x != null) {
mdtp = x.index;
List<String> l = pass2Ala(s);
mdtp++;
String temp = new String();
while(!(temp =
mdt.get(mdtp)).trim().equalsIgnoreCase("MEND")) {
String line = new String();
StringTokenizer st2 = new
StringTokenizer(temp, " ,",false);
for(int i=0 ; i<12 ; i++) {
line += " ";
}
String opcode = st2.nextToken();
line += opcode;
for(int i=opcode.length() ; i<24 ;
i++) {
line += " ";
}
line += st2.nextToken();
while(st2.hasMoreTokens()) {
String token2 = st2.nextToken();
int index;
if((index = token2.indexOf("#"))
!= -1) {
line += "," +
l.get(Integer.parseInt(token2.substring(index+1,index
+2)));
}
}
mdtp++;
output.println(line);
System.out.println(line);
}
break;
} else {
output.println(s);
System.out.println(s);
break;
}
}
}
System.out.println("\nALA:");
showAla(2);
}
static List<String> pass2Ala(String s) {
StringTokenizer st = new StringTokenizer(s, " ",
false);
int num_tokens = st.countTokens();
String macro_name = st.nextToken();
int ala_no = ala_macro_binding.get(macro_name);
List<String> l = ala.get(ala_no);
int ctr = 0;
StringTokenizer st2 = null;
try {
st2 = new StringTokenizer(st.nextToken(), ",",
false);
while(st2.hasMoreTokens()) {
l.set(ctr, st2.nextToken());
ctr++;
}
} catch(Exception e) {
// do nothing
}
if(ctr < num_tokens) {
String s2 = mdt.get(mdtp);
StringTokenizer st3 = new StringTokenizer(s2, " ,",
false);
String token = new String();
int index = 0;
while(st3.hasMoreTokens()) {
token = st3.nextToken();
if((index = token.indexOf("=")) != -1) {
try {
l.set(ctr++, token.substring(index+1,
token.length()));
} catch(Exception e) {
// do nothing
}
}
}
}
ala.set(ala_no, l);
return l;
}
static void initializeTables() {
mnt = new LinkedList<>();
mdt = new ArrayList<>();
ala = new LinkedList<>();
mntc = 0;
mdtc = 0;
```

```
ala_macro_binding = new HashMap<>();
}
}
/*
_______________________________________________
INPUT
MACRO
INCR1 &FIRST,&SECOND=DATA9
A 1,&FIRST
L 2,&SECOND
MEND
MACRO
INCR2 &ARG1,&ARG2=DATA5
L 3,&ARG1
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 1F
DROP 8
END
_______________________________________________
OUTPUT
pvgcoen-3@pvgcoen3-ThinkCentre-M700:~/PRACT4$ javac
MacroProcessor.java
pvgcoen-3@pvgcoen3-ThinkCentre-M700:~/PRACT4$ java
MacroProcessor
===== PASS 1 =====
ALA:
[&FIRST, &SECOND]
[&ARG1, &ARG2]
MNT:
[INCR1, 0]
[INCR2, 4]
MDT:
INCR1 &FIRST,&SECOND=DATA9
A 1,#0
L 2,#1
MEND
```

```
INCR2 &ARG1,&ARG2=DATA5
L 3,#0
ST 4,#1
MEND
===== PASS 2 =====
PRG2 START
USING *,BASE
A 1,DATA1
L 2,DATA9
L 3,DATA3
ST 4,DATA4
FOUR DC F'4'
FIVE DC F'5'
BASE EQU 8
TEMP DS 1F
DROP 8
END
ALA:
[DATA1, DATA9]
[DATA3, DATA4]
*/
```

```
/*
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.
*/
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,initial
izedParams[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:
neha@neha-1011PX:~/Desktop/neha_SPOS/Turn1/A4$ javac macroPass2.java
neha@neha-1011PX:~/Desktop/neha_SPOS/Turn1/A4$ java macroPass2
```

```
neha@neha-1011PX:~/Desktop/neha_SPOS/Turn1/A4$ cat Pass2.txt
+ MOVE AREG,10
+ ADD AREG,='1'
+ MOVER AREG,20
+ ADD AREG,='5'
+ MOVER &AREG,100
+ MOVER &BREG,200
+ ADD &AREG,='15'
+ ADD &BREG,='10'

*/
```

FIFO PAGE REPLACEMENT :

```
import java.io.*;
public class FIFO {
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;
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)
{
buffer[pointer] = reference[i];
fault++;
pointer++;
if(pointer == frames)
pointer = 0;
}
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:
20
Please enter the reference string:
7
0
1
2
0
3
0
4
2
3
0
3
2
1
2
0
1
7
0
1
7 7 7 2 2 2 2 4 4 4 0 0 0 0 0 0 0 7
7 7
```

```
-1 0 0 0 0 3 3 3 2 2 2 2 2 1 1 1 1 1
0 0
-1 -1 1 1 1 1 0 0 0 3 3 3 3 3 2 2 2 2
2 1
The number of Hits: 5
Hit Ratio: 0.25
The number of Faults: 15
---------------------------------
```

LRU Page Replacement algorithm in java

code in Java:

```
import java.io.*;
import java.util.*;
public class LRU {
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[];
ArrayList<Integer> stack = new ArrayList<Integer>();
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++)
{
if(stack.contains(reference[i]))
{
stack.remove(stack.indexOf(reference[i]));
}
stack.add(reference[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 min_loc = ref_len;
for(int j = 0; j < frames; j++)
{
if(stack.contains(buffer[j]))
{
int temp = stack.indexOf(buffer[j]);
if(temp < min_loc)
{
min_loc = temp;
pointer = j;
}
}
}
}
buffer[pointer] = reference[i];
fault++;
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:
20
Please enter the reference string:
7
0
1
2
0
3
0
4
2
3
0
3
2
1
2
0
1
7
0
1
7  7  7  2  2  2  2  4  4  4  0  0  0  1  1  1  1  1  1
1
-1  0  0  0  0  0  0  0  0  3  3  3  3  3  3  0  0  0  0
0
-1  -1  1  1  1  3  3  3  2  2  2  2  2  2  2  2  2  7  7
7
The number of Hits: 8
Hit Ratio: 0.4
The number of Faults: 12
--------------------------------
```

Optimal Page Replacement algorithm in java

code in Java:

```
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:-

```
/*
Problem Statement :
Write a Java Program (using OOP features) to implement paging simulation using
1. Least Recently Used (LRU)
2. Optimal algorithm

                                                ****Optimal****
*/
import java.util.*;
import java.io.*;


class Optimal
{
        public static void main(String args[])throws IOException
        {
                BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
                int numberOfFrames, numberOfPages, flag1, flag2, flag3, i, j, k, pos = 0, max;
                int faults = 0;
                int temp[] = new int[10];

                System.out.println("Enter number of Frames: ");
                numberOfFrames = Integer.parseInt(br.readLine());
                int frame[] = new int[numberOfFrames];


                System.out.println("Enter number of Pages: ");
                numberOfPages = Integer.parseInt(br.readLine());

                int pages[] = new int[numberOfPages];
                System.out.println("Enter the pages: ");
                for(i=0; i<numberOfPages; i++)
                        pages[i] = Integer.parseInt(br.readLine());

                for(i = 0; i < numberOfFrames; i++)
            frame[i] = -1;


                for(i = 0; i < numberOfPages; ++i){
                    flag1 = flag2 = 0;

                    for(j = 0; j < numberOfFrames; ++j){
                      if(frame[j] == pages[i]){
                          flag1 = flag2 = 1;
                          break;
                        }
                    }
```

```
if(flag1 == 0){
    for(j = 0; j < numberOfFrames; ++j){
        if(frame[j] == -1){
            faults++;
            frame[j] = pages[i];
            flag2 = 1;
            break;
        }
    }
}

if(flag2 == 0){
    flag3 =0;

    for(j = 0; j < numberOfFrames; ++j){
        temp[j] = -1;

        for(k = i + 1; k < numberOfPages; ++k){
            if(frame[j] == pages[k]){
                temp[j] = k;
                break;
            }
        }
    }

    for(j = 0; j < numberOfFrames; ++j){
        if(temp[j] == -1){
            pos = j;
            flag3 = 1;
            break;
        }
    }

    if(flag3 ==0){
        max = temp[0];
        pos = 0;

        for(j = 1; j < numberOfFrames; ++j){
            if(temp[j] > max){
                max = temp[j];
                pos = j;
            }
        }
    }

    frame[pos] = pages[i];
    faults++;
}
```

```
//                System.out.print();

                  for(j = 0; j < numberOfFrames; ++j){
                    System.out.print("\t"+ frame[j]);
                  }
                }

                System.out.println("\n\nTotal Page Faults: "+ faults);



       }


}

//7 0 1 2 0 3 0 4 2 3 0 3 2




/*
Problem Statement :
Write a Java Program (using OOP features) to implement paging simulation using
1. Least Recently Used (LRU)
2. Optimal algorithm
                                           ****LRU****
*/
import java.io.*;
   class lru
    {
    public static void main(String args[])throws IOException
     {
               BufferedReader obj=new BufferedReader(new InputStreamReader(System.in));
               int f,page=0,ch,pgf=0,n,chn=0;
               boolean flag;
               int pages[];                    //pgf-page fault

              System.out.println("1.LRU");
              int pt=0;
       System.out.println("enter no. of frames: ");
               f=Integer.parseInt(obj.readLine());
              int frame[]=new int[f];


              for(int i=0;i<f;i++)
              {
                  frame[i]=-1;
              }

              System.out.println("enter the no of pages ");
              n=Integer.parseInt(obj.readLine());

            pages=new int[n];
              System.out.println("enter the page no ");

              for(int j=0;j<n;j++)
              pages[j]=Integer.parseInt(obj.readLine());

              int pg=0;
              for(pg=0;pg<n;pg++)
       {
                  page=pages[pg];
                  flag=true;
                  for(int j=0;j<f;j++)
                  {
                      if(page==frame[j])
                      {
                          flag=false;
                          break;
                       }
                  }
                  int temp,h=3,i;
                  if(flag)
              {
                  if( frame[1]!=-1 && frame[2]!=-1 && frame[0]!=-1)
                      {
                          temp=pages[pg-3];
                          if(temp==pages[pg-2] || temp==pages[pg-1])
                              temp=pages[pg-4];

                          for(i=0;i<f;i++)
                              if(temp==frame[i])
                                  break;
                          frame[i]=pages[pg];
                      }
                      else
                      {
                          if(frame[0]==-1)
                              frame[0]=pages[pg];
                          else if(frame[1]==-1)
                              frame[1]=pages[pg];
                          else if(frame[2]==-1)
                              frame[2]=pages[pg];
                      }


                      System.out.print("frame :");
                      for(int j=0;j<f;j++)
                      System.out.print(frame[j]+"   ");
                      System.out.println();
                      pgf++;
                  }
                  else
                  {
                      System.out.print("frame :");
                      for(int j=0;j<f;j++)
                      System.out.print(frame[j]+"  ");
                      System.out.println();
                  }

              }//for

       System.out.println("Page fault:"+pgf);

   }//main
   }//class


   /*
   OUTPUT:-

   akshay@akshay-1011PX:~/Desktop/SPOS/LRU$ javac lru.java
   akshay@akshay-1011PX:~/Desktop/SPOS/LRU$ java lru
   1.LRU
   enter no. of frames:
   4
   enter the no of pages
   10
   enter the page no
   1
   0
   1
   2
   3
   7
   8
   1
   5
   2
   frame :1   -1   -1   -1
   frame :1   0   -1   -1
   frame :1  0  -1  -1
   frame :1   0   2   -1
   frame :1   3   2   -1
   frame :7   3   2   -1
   frame :7   3   8   -1
   frame :7   1   8   -1
   frame :5   1   8   -1
   frame :5   1   2   -1
   Page fault:9
   akshay@akshay-1011PX:~/Desktop/SPOS/LRU$
   */
```

```
import java.util.Scanner;
class fcfs{
public static void main(String args[]){
int
burst_time[],process[],waiting_time[],tat[],i,j,n,tot
al=0,pos,temp;
float wait_avg, TAT_avg;
Scanner s = new Scanner(System.in);
System.out.print("Enter number of process: ");
n = s.nextInt();
process = new int[n];
burst_time = new int[n];
waiting_time = new int[n];
tat = new int[n];
System.out.println("\nEnter Burst time:");
for(i=0;i<n;i++)
{
System.out.print("\nProcess["+(i+1)+"]: ");
burst_time[i] = s.nextInt();;
process[i]=i+1; //Process Number
}
//First process has 0 waiting time
waiting_time[0]=0;
//calculate waiting time
for(i=1;i<n;i++)
{
waiting_time[i]=0;
for(j=0;j<i;j++)
waiting_time[i]+=burst_time[j];
total+=waiting_time[i];
}
//Calculating Average waiting time
wait_avg=(float)total/n;
total=0;
System.out.println("\nProcess\t Burst Time \tWaiting
Time\tTurnaround
Time");
for(i=0;i<n;i++)
{
tat[i]=burst_time[i]+waiting_time[i];
total+=tat[i];//Calculating
TurnaroundTimetotal+=tat[i];
System.out.println("\n
p"+process[i]+"\t\t"+burst_time[i]+"\t\t"+waiting_tim
e[i]+"\t\t
"+tat[i]);
}
//Calculation of Average Turnaround Time
TAT_avg=(float)total/n;
System.out.println("\n\nAverage Waiting Time:
"+wait_avg);
System.out.println("\nAverage Turnaround Time:
"+TAT_avg);
}
}
/* OUTPUT
D:\SPOS>java fcfs
Enter number of process: 4
Enter Burst time:
Process[1]: 3
Process[2]: 5
Process[3]: 2
Process[4]: 10
Process Burst Time Waiting Time Turnaround Time
p1 3 0 3
p2 5 3 8
p3 2 8 10
p4 10 10 20
Average Waiting Time: 5.25
Average Turnaround Time: 10.25
*/
```

```
                    1.FCFS
*/
import java.io.*;
import java.util.Scanner;
public class FCFS
{
          public static void main(String args[])
          {
                    int i,no_p,burst_time[],TT[],WT[];
                    float avg_wait=0,avg_TT=0;
                    burst_time=new int[50];
                    TT=new int[50];
                    WT=new int[50];
                    WT[0]=0;
                    Scanner s=new Scanner(System.in);
                    System.out.println("Enter the number of process: ");
                    no_p=s.nextInt();
                    System.out.println("\nEnter Burst Time for processes:");
                    for(i=0;i<no_p;i++)
                    {
                              System.out.print("\tP"+(i+1)+":  ");
                              burst_time[i]=s.nextInt();
                    }

                    for(i=1;i<no_p;i++)
                    {
                              WT[i]=WT[i-1]+burst_time[i-1];
                              avg_wait+=WT[i];
                    }
                    avg_wait/=no_p;

                    for(i=0;i<no_p;i++)
                    {
                              TT[i]=WT[i]+burst_time[i];
                              avg_TT+=TT[i];
                    }
                    avg_TT/=no_p;

          System.out.println("\n************************************************************
*******");
                    System.out.println("\tProcesses:");

          System.out.println("**************************************************************
*****");
                    System.out.println("    Process\tBurst Time\tWaiting Time\tTurn Around Time");
                    for(i=0;i<no_p;i++)
                    {
```

```
			System.out.println("\tP"+(i+1)+"\t  "+burst_time[i]+"\t\t  "+WT[i]+"\t\t "+TT[i]);

		}
		System.out.println("\n--------------------------------------------------------------");
		System.out.println("\nAverage waiting time : "+avg_wait);
		System.out.println("\nAverage Turn Around time : "+avg_TT+"\n");
	}
}

/*Output:
Enter the number of process:
3

Enter Burst Time for processes:
	P1:  24
	P2:  3
	P3:  3

**************************************************************
	Processes:
**************************************************************
   Process        Burst Time        Waiting Time     Turn Around Time
	P1	  24		   0		    24
	P2	  3		   24		    27
	P3	  3		   27		    30

--------------------------------------------------------------
Average waiting time : 17.0
Average Turn Around time : 27.0  */
```

```
/*Round Robin(Preemptive)*/
import java.util.*;
import java.io.*;
class RoundR
{
	public static void main(String args[])
	{
		int Process[]=new int[10];
		int a[]=new int[10];
		int Arrival_time[]=new int[10];
```

```
        int Burst_time[]=new int[10];
        int WT[]=new int[10];
        int TAT[]=new int[10];
        int Pno,sum=0;;
        int TimeQuantum;

System.out.println("\nEnter the no. of Process::");
        Scanner sc=new Scanner(System.in);
        Pno=sc.nextInt();
        System.out.println("\nEnter each process::");
        for(int i=0;i<Pno;i++)
        {
            Process[i]=sc.nextInt();
        }

System.out.println("\nEnter the Burst Time of each process::");
        for(int i=0;i<Pno;i++)
        {
            Burst_time[i]=sc.nextInt();
        }
System.out.println("\nEnter the Time Quantum::");
TimeQuantum=sc.nextInt();
        do{
        for(int i=0;i<Pno;i++)
        {
            if(Burst_time[i]>TimeQuantum)
            {
                Burst_time[i]-=TimeQuantum;
                for(int j=0;j<Pno;j++)
                {
                    if((j!=i)&&(Burst_time[j]!=0))
                WT[j]+=TimeQuantum;
            }
        }
        else
        {
            for(int j=0;j<Pno;j++)
            {
                if((j!=i)&&(Burst_time[j]!=0))
                WT[j]+=Burst_time[i];
            }
            Burst_time[i]=0;
         }
      }
        sum=0;
        for(int k=0;k<Pno;k++)
        sum=sum+Burst_time[k];
     } while(sum!=0);
```

```
                    for(int i=0;i<Pno;i++)
                                TAT[i]=WT[i]+a[i];
                     System.out.println("process\t\tBT\tWT\tTAT");
                     for(int i=0;i<Pno;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(int j=0;j<Pno;j++)
                     {
                           avg_wt+=WT[j];
                     }
                     for(int j=0;j<Pno;j++)
                     {
                           avg_tat+=TAT[j];
                     }
                       System.out.println("average waiting time "+(avg_wt/Pno)+"\n Average turn around
time"+(avg_tat/Pno));
            }
}

/*OUTPUT::
unix@unix-HP-280-G1-
MT:~/TEA33$ java RoundR
Enter the no. of Process::
5
Enter each process::
1
2
3
4
5

Enter the Burst Time of each process::
2
1
8
4
5
Enter the Time Quantum::
2
process         BT      WT      TAT
process1        0       0       0
process2        0       2       2
process3        0       12      12
process4        0       9       9
```

```
process5          0          13         13
average waiting time 7.2
Average turn around time7.2       */
```

Round Robin

```
import java.util.Scanner;
public class Roundfinal1 {
public static void main(String args[]) {
Scanner s = new Scanner(System.in);
int wtime[],btime[],rtime[],num,quantum,total;
wtime = new int[10];
btime = new int[10];
rtime = new int[10];
System.out.print("Enter number of processes(MAX 10):
");
num = s.nextInt();
System.out.print("Enter burst time");
for(int i=0;i<num;i++) {
System.out.print("\nP["+(i+1)+"]: ");
btime[i] = s.nextInt(); rtime[i] = btime[i]; wtime[i]=0; }
System.out.print("\n\nEnter quantum: "); quantum =
s.nextInt();
int rp = num; int i=0; int time=0; System.out.print("0");
wtime[0]=0; while(rp!=0) { if(rtime[i]>quantum)
{
rtime[i]=rtime[i]-quantum;
System.out.print(" | P["+(i+1)+"] | ");
time+=quantum;
System.out.print(time);
}
else if(rtime[i]<=quantum && rtime[i]>0)
{time+=rtime[i];
rtime[i]=rtime[i]-rtime[i];
System.out.print(" | P["+(i+1)+"] | ");
rp--;
System.out.print(time);
}
i++;
if(i==num)
{
i=0;
}
}
}
}
```

3. Priority

```
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];
}
```

```
/*                    2. SJF(Non-Preemptive)          */
import java.util.Scanner;
class SJF1{
public static void main(String args[]){
int burst_time[],process[],waiting_time[],tat[],i,j,n,total=0,pos,temp;
float wait_avg,TAT_avg;
Scanner s = new Scanner(System.in);

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

```
n = s.nextInt();

process = new int[n];
burst_time = new int[n];
waiting_time = new int[n];
tat = new int[n];

System.out.println("\nEnter Burst time:");
for(i=0;i<n;i++)
{
System.out.print("\nProcess["+(i+1)+"]: ");
burst_time[i] = s.nextInt();;
process[i]=i+1; //Process Number
}

//Sorting
for(i=0;i<n;i++)
{
pos=i;
for(j=i+1;j<n;j++)
{
if(burst_time[j]<burst_time[pos])
pos=j;
}

temp=burst_time[i];
burst_time[i]=burst_time[pos];
burst_time[pos]=temp;

temp=process[i];
process[i]=process[pos];
process[pos]=temp;
}
//First process has 0 waiting time
waiting_time[0]=0;
//calculate waiting time
for(i=1;i<n;i++)
{
waiting_time[i]=0;
for(j=0;j<i;j++)
waiting_time[i]+=burst_time[j];
total+=waiting_time[i];
}

//Calculating Average waiting time
wait_avg=(float)total/n;
total=0;
```

```
System.out.println("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");
for(i=0;i<n;i++)
{
tat[i]=burst_time[i]+waiting_time[i]; //Calculating Turnaround Time
total+=tat[i];
System.out.println("\n p"+process[i]+"\t\t "+burst_time[i]+"\t\t "+waiting_time[i]+"\t\t "+tat[i]);
}

//Calculation of Average Turnaround Time
TAT_avg=(float)total/n;
System.out.println("\n\nAverage Waiting Time: "+wait_avg);
System.out.println("\nAverage Turnaround Time: "+TAT_avg);

}
}
```

```
/* 2. SJF(Preemptive)*/
import java.util.Scanner;

class sjf_swap1{
public static void main(String args[])

{
int
burst_time[],process[],waiting_time[],tat[],arr_time[],completion_time[],i,j,n,total=0,total_comp=0,
pos,temp;
float wait_avg,TAT_avg;
Scanner s = new Scanner(System.in);
 System.out.print("Enter number of process: ");
n = s.nextInt();
 process = new int[n];
burst_time = new int[n];
waiting_time = new int[n];
arr_time=new int[n];
tat = new int[n];
completion_time=new int[n];
```

```
//burst time
System.out.println("\nEnter Burst time:");
for(i=0;i<n;i++)
{
System.out.print("\nProcess["+(i+1)+"]: ");
burst_time[i] = s.nextInt();;
process[i]=i+1; //Process Number
}

//arrival time
System.out.println("\nEnter arrival time:");
for(i=0;i<n;i++)
{
System.out.print("\nProcess["+(i+1)+"]: ");
arr_time[i] = s.nextInt();;
process[i]=i+1; //Process Number
}

//Sorting
for(i=0;i<n;i++)
{
pos=i;
for(j=i+1;j<n;j++)
{
if(burst_time[j]<burst_time[pos])
pos=j;
}

temp=burst_time[i];
burst_time[i]=burst_time[pos];
burst_time[pos]=temp;

temp=process[i];
process[i]=process[pos];
process[pos]=temp;

System.out.println("process"+process[i]);
}
//completion
time new
for(i=1;i<n;i++)
{
completion_time[i]=0;
for(j=0;j<i;j++)
completion_time[i]+=burst_time[j];
 total_comp+=completion_time[i];
}
```

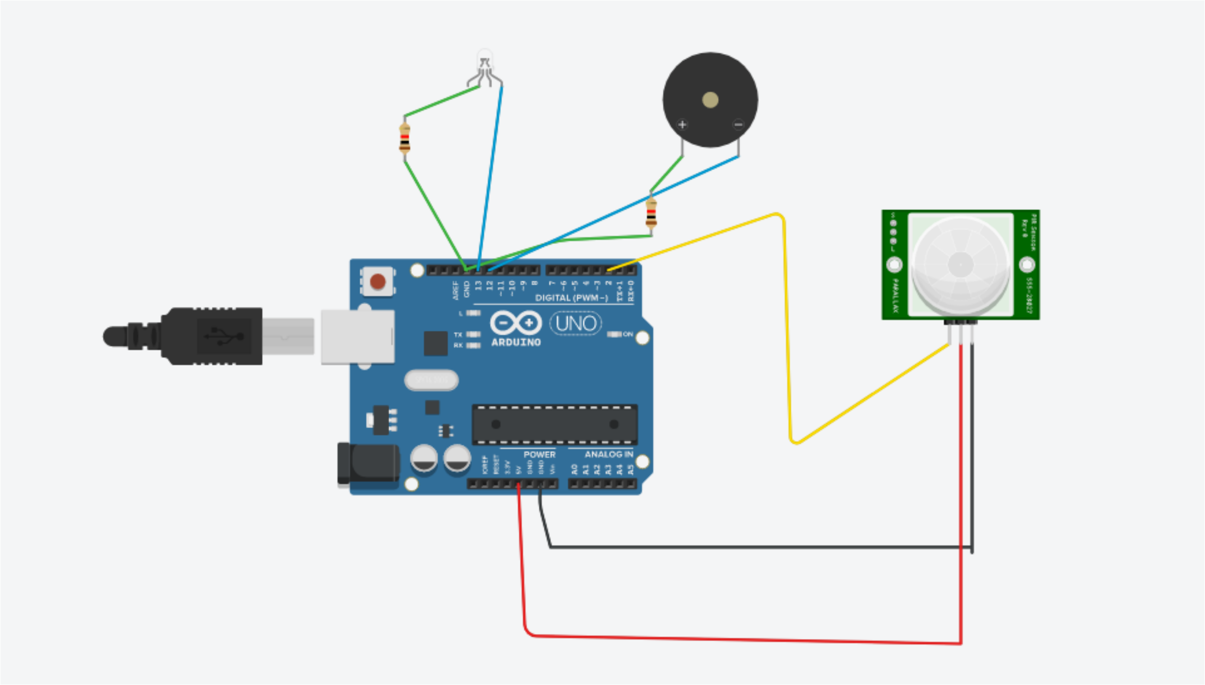
```
//First process has 0 waiting
time
waiting_time[0]=0;
//calculate

waiting time
for(i=1;i<n;i++)
{
waiting_time[i]=0;
for(j=0;j<i;j++)
waiting_time[i]+=burst_time[j];
total+=waiting_time[i];
}
```

PR:01

Title: Understanding the connectivity of Raspberry-Pi / Adriano with IR sensor. Write an application to detect obstacle and notify user using LEDs.

CODE:

```
int pirsensor=0;

void setup()

{

  pinMode(12,OUTPUT);

  pinMode(13,OUTPUT);

  pinMode(2,INPUT);

}


void loop()

{

 pirsensor=digitalRead(2);

  if(pirsensor==HIGH)

  {

   digitalWrite(13,HIGH);

    tone(12,500,500);

  }

  digitalWrite(13,LOW);

}
```

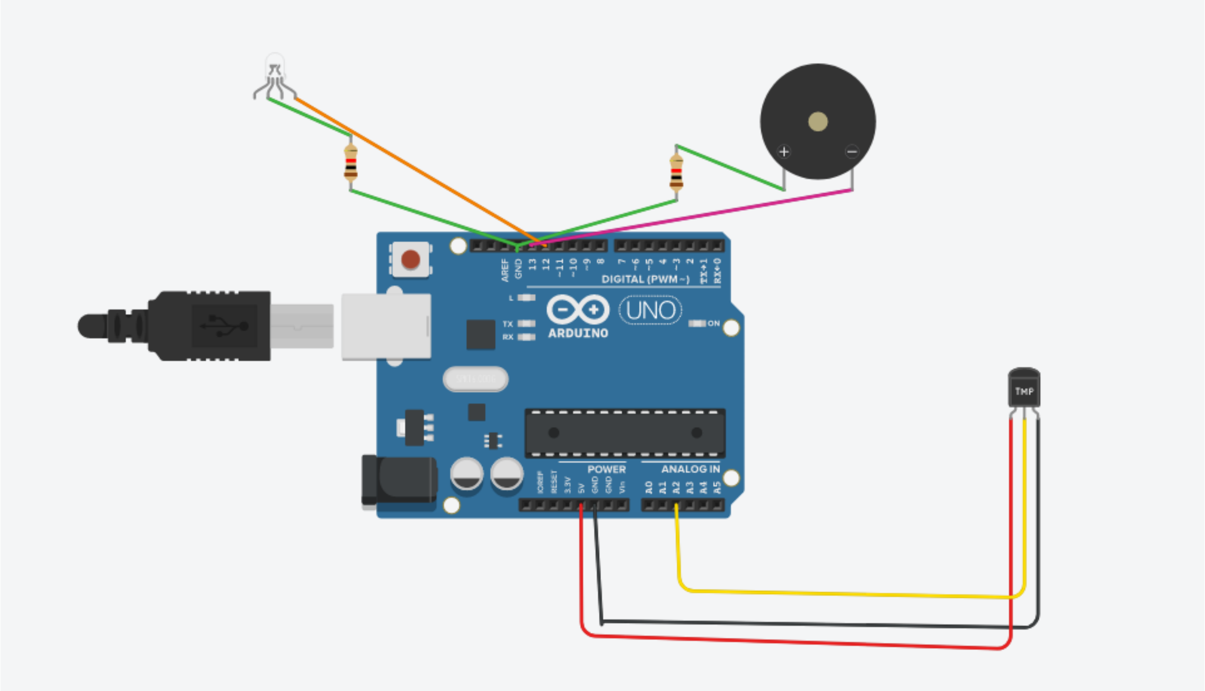


PR:02

Title: Understanding the connectivity of Raspberry-Pi /Beagle board circuit with temperature sensor. Write an application to read the environment temperature. If temperature crosses a threshold value, generate alerts using LEDs.

CODE:

```
int tsensor;

void setup()

{

  pinMode(A2,INPUT);

  pinMode(13,OUTPUT);

  pinMode(12,OUTPUT);

}


 void loop()
 {
  tsensor=analogRead(A2);
  if(tsensor >= 200)
  {
    digitalWrite(13,HIGH);
    tone(5,500,500);
  }

digitalWrite(13,LOW);
 }
```



+
−
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM ~)
L
TX
RX
ARDUINO
UNO
ON
IOREF
RESET
3.3V
5V
GND
GND
Vin
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
TMP