```
/*
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].equals
IgnoreCase("ORIGIN")||tokens[1].equalsIgnoreCase("EQU")||tokens[1].equalsIgnoreCase("LTORG"))
        //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 */

```
/* Problem Statement: Write a JAVA program (using oop features) to implement following
1. FCFS
2. SJF(Preemptive)
3. Priority(Non-  Preemptive)
4. Round Robin(Preemptive)

                    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 */

```
/*                    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];
}
```

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

System.out.println("\nPro_number\t Burst Time \tcompletion_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"+process[i]+"\t\t "+burst_time[i]+"\t\t
"+completion_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\nAWT: "+wait_avg);
System.out.println("\nATAT: "+TAT_avg);

}
}
```

```
/*
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
```

Practical 5

Understanding the connectivity of raspberry-pi/Arduino with IR sensor. Write an application to detect obstacle and notify user using leds.

```
int obstaclePin=13;

int hasObstacle=HIGH;
void setup() {
  pinMode(obstaclePin,INPUT);
  pinMode(12,OUTPUT)
  Serial.begin(9600);
}

void loop() {
  hasObstacle=digitalRead(obstaclePin);
  if(hasObstacle==LOW){
    Serial.println("Stop Something is ahead!!");
    digitalWrite(12,HIGH);
  }else{
      Serial.println("Path is clear");
      digitalWrite(12,LOW);
  }
  delay(2000);
}
```

Practical 6

Understanding the connectivity of raspberry-pi/Arduino with DHT11/22 sensor. Write an application to notify different levels using LEDs

```
#include <dht.h>
#define outPin 8

dht DHT;

void setup(){
  Serial.begin(9600);

  pinMode(12,OUTPUT);
}

void loop(){
  int readData=DHT.read11(outPin);

  float t = DHT.temperature;
  float h = DHT.humidity;

  if(t > 31){
    digitalWrite(12,HIGH);
  }
  else{
    digitalWrite(12,LOW);
  }
  Serial.print("Temperature = ");
  Serial.print(t);
  Serial.print("C|");
  Serial.print((t*9.0)/5.0+32.0);
  Serial.println("F");
  Serial.print("Humidity = ");
  Serial.print(h);
  Serial.println("%");
  Serial.println("");

  delay(2000);
}
```

Practical 7

Understanding the connectivity of raspberry-pi/Arduino with Ultrasonic sensor. Write an application for distance measurement and notify it on the serial monitor.

```
const int pingPin = 7; // Trigger Pin of Ultrasonic Sensor
const int echoPin = 6; // Echo Pin of Ultrasonic Sensor

void setup() {
  Serial.begin(9600);
}

void loop() {
  long duration, inches, cm;
  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(pingPin, LOW);
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);
  inches = microsecondsToInches(duration);
  cm = microsecondsToCentimeters(duration);
  Serial.print(inches);
  Serial.print("in, ");
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();
  delay(1000);
}

long microsecondsToInches(long microseconds) {
  return microseconds / 74 / 2;
}

long microsecondsToCentimeters(long microseconds) {
  return microseconds / 29 / 2;
}
```

Practical 8

Understanding the Understanding the connectivity of raspberry-pi/Arduino with water level sensor. Write an application to detect water level and notify user using leds and monitor.

```
#define sensorPin A0
#define buzzer 6
// Value for storing water level
int val = 0;

void setup() {
pinMode(buzzer, OUTPUT);
digitalWrite(buzzer,LOW);
}

void loop() {

  int level = analogRead(sensorPin);
  if(level>450)
  {
  digitalWrite(buzzer,HIGH);
  }
  else
  {
  digitalWrite(buzzer,LOW);
  }
  delay(1000);
}
```

Practical 9

Understanding the connectivity of raspberry-pi/Arduino with temperature sensor. Write an application to read the environment temperature. If temperature crosses a threshold value, generate alerts using leds.

```
#include <dht.h>
#define outPin 8

dht DHT;

void setup(){
  Serial.begin(9600);

  pinMode(12,OUTPUT);
}

void loop(){
  int readData=DHT.read11(outPin);

  float t = DHT.temperature;
  float h = DHT.humidity;

  if(t > 31){
    digitalWrite(12,HIGH);
  }
  else{
    digitalWrite(12,LOW);
  }
  Serial.print("Temperature = ");
  Serial.print(t);
  Serial.print("C|");
  Serial.print((t*9.0)/5.0+32.0);
  Serial.println("F");
  Serial.print("Humidity = ");
  Serial.print(h);
  Serial.println("%");
  Serial.println("");

  delay(2000);
}
```

Practical 10

Understanding the connectivity of raspberry-pi/Arduino with water level and soil moisture sensor. Write an application to implement a smart gardening system.

```
void setup() {
  pinMode(A0,INPUT);
  pinMode(A1,INPUT);
  Serial.begin(9600);
}

void loop() {
  int val1=analogRead(A0);
  int val2=analogRead(A1);
  Serial.print("Water Level= ");
  Serial.print(val1);
  Serial.print("Soil Moisture= ");
  Serial.print(val2);
  delay(2000);
}
```