1. Design suitable data structures and implement pass-I of a two-pass assembler for pseudo-machine. And generate symbol table, literal table and intermediate code for given sample input:

2. Design suitable data structure and implement pass II of a two pass assembler for pseudo-machine for provide sample input:

3. Write a program to simulate FCFS CPU scheduling Algorithm

```
import java.text.ParseException;
class FCFS {
   static void findWaitingTime(int processes[], int n,        int bt[], int wt[]) {
      wt[0] = 0;
      for (int i = 1; i < n; i++) {
         wt[i] = bt[i - 1] + wt[i - 1];
      }
   }
   static void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[]) {
      for (int i = 0; i < n; i++) {
         tat[i] = bt[i] + wt[i];
           }
   }
   static void findavgTime(int processes[], int n, int bt[]) {
      int wt[] = new int[n], tat[] = new int[n];
           int total_wt = 0, total_tat = 0;
           findWaitingTime(processes, n, bt, wt);
           findTurnAroundTime(processes, n, bt, wt, tat);
           System.out.printf("Processes Burst time Waiting"+" time Turn around time\n");
      for (int i = 0; i < n; i++) {
         total_wt = total_wt + wt[i];
         total_tat = total_tat + tat[i];
         System.out.printf(" %d ", (i + 1));
         System.out.printf("          %d ", bt[i]);
         System.out.printf("          %d", wt[i]);
         System.out.printf("          %d\n", tat[i]);
           }
           float s = (float)total_wt /(float) n;
           int t = total_tat / n;
           System.out.printf("Average waiting time = %f", s);
           System.out.printf("\n");
           System.out.printf("Average turn around time = %d ", t);
   }
   public static void main(String[] args) throws ParseException {
      int processes[] = {1, 2, 3};
           int n = processes.length;
           int burst_time[] = {10, 5, 8};
      findavgTime(processes, n, burst_time);
   }
}
```

4. Write a program to simulate Preemptive SJF CPU scheduling Algorithm

```
import java.io.*;
import java.util.*;
public class SJF {
   public static void main(String[] args){
          Scanner input = new Scanner(System.in);
          int n;
          int[][] A = new int[100][4];
          int total = 0;
          float avg_wt, avg_tat;
          System.out.println("Enter number of process:");
          n = input.nextInt();
          System.out.println("Enter Burst Time:");
          for (int i = 0; i < n; i++) {
        System.out.print("P" + (i + 1) + ": ");
        A[i][1] = input.nextInt();
        A[i][0] = i + 1;
          }
          for (int i = 0; i < n; i++) {
        int index = i;
        for (int j = i + 1; j < n; j++) {
                    if (A[j][1] < A[index][1]) {
              index = j;
                    }
        }
        int temp = A[i][1];
        A[i][1] = A[index][1];
        A[index][1] = temp;
        temp = A[i][0];
        A[i][0] = A[index][0];
        A[index][0] = temp;
     }
          A[0][2] = 0;
          for (int i = 1; i < n; i++) {
        A[i][2] = 0;
        for (int j = 0; j < i; j++) {
          A[i][2] += A[j][1];
        }
        total += A[i][2];
          }
     avg_wt = (float)total / n;
          total = 0;
          System.out.println("P\tBT\tWT\tTAT");
          for (int i = 0; i < n; i++) {
        A[i][3] = A[i][1] + A[i][2];
        total += A[i][3];
        System.out.println("P" + A[i][0] + "\t"+ A[i][1] + "\t" + A[i][2]+ "\t" + A[i][3]);
          }
          avg_tat = (float)total / n;
          System.out.println("Average Waiting Time= "+ avg_wt);
          System.out.println("Average Turnaround Time= "+ avg_tat);
          }
}
```

5. Write a program to simulate Non-preemptive priority CPU scheduling algorithm.

```
import java.util.*;
class Process {
   int pid;
   int bt;
   int priority;
   Process(int pid, int bt, int priority){
           this.pid = pid;
           this.bt = bt;
           this.priority = priority;
   }
   public int prior() { return priority; }
}
public class Priority {
   public void findWaitingTime(Process proc[], int n,int wt[]){
      wt[0] = 0;
           for (int i = 1; i < n; i++)
         wt[i] = proc[i - 1].bt + wt[i - 1];
   }
   public void findTurnAroundTime(Process proc[], int n,int wt[], int tat[]){
      for (int i = 0; i < n; i++)
           tat[i] = proc[i].bt + wt[i];
   }
   public void findavgTime(Process proc[], int n){
      int wt[] = new int[n], tat[] = new int[n],
      total_wt = 0, total_tat = 0;
      findWaitingTime(proc, n, wt);
      findTurnAroundTime(proc, n, wt, tat);
      System.out.print("\nProcesses Burst time Waiting time Turn around time\n");
      for (int i = 0; i < n; i++) {
         total_wt = total_wt + wt[i];
         total_tat = total_tat + tat[i];
         System.out.print(" " + proc[i].pid + "\t\t"+ proc[i].bt + "\t " + wt[i]+ "\t\t " + tat[i] + "\n");
      }
      System.out.print("\nAverage waiting time = "+ (float)total_wt / (float)n);
      System.out.print("\nAverage turn around time = "+ (float)total_tat / (float)n);
   }
   public void priorityScheduling(Process proc[], int n){
      Arrays.sort(proc, new Comparator<Process>() {
         @Override
         public int compare(Process a, Process b){
            return b.prior() - a.prior();
         }
      });
      System.out.print("Order in\" which processes gets executed \n");
      for (int i = 0; i < n; i++)
         System.out.print(proc[i].pid + " ");
         findavgTime(proc, n);
   }
   public static void main(String[] args){
      Priority ob = new Priority();
      int n = 3;
```

```
        Process proc[] = new Process[n];
        proc[0] = new Process(1, 10, 2);
        proc[1] = new Process(2, 5, 0);
        proc[2] = new Process(3, 8, 1);
        ob.priorityScheduling(proc, n);
    }
}
```

6. Write a program to simulate preemptive Round Robin CPU scheduling algorithm.

```
public class RoundRobin {
   static void findWaitingTime(int processes[], int n, int bt[], int wt[], int quantum){
      int rem_bt[] = new int[n];
           for (int i = 0 ; i < n ; i++)
           rem_bt[i] = bt[i];
           int t = 0;
           while(true){
          boolean done = true;
          for (int i = 0 ; i < n; i++){
            if (rem_bt[i] > 0){
               done = false;
               if (rem_bt[i] > quantum){
                  t += quantum;
                  rem_bt[i] -= quantum;
               }
               else{
                  t = t + rem_bt[i];
                  wt[i] = t - bt[i];
                  rem_bt[i] = 0;
               }
                     }
          }
          if (done == true)
                     break;
           }
   }
   static void findTurnAroundTime(int processes[], int n,int bt[], int wt[], int tat[]){
      for (int i = 0; i < n ; i++)
         tat[i] = bt[i] + wt[i];
   }
   static void findavgTime(int processes[], int n, int bt[],int quantum){
      int wt[] = new int[n], tat[] = new int[n];
           int total_wt = 0, total_tat = 0;
           findWaitingTime(processes, n, bt, wt, quantum);
           findTurnAroundTime(processes, n, bt, wt, tat);
           System.out.println("PN " + " B " +" WT " + " TAT");
           for (int i=0; i<n; i++){
          total_wt = total_wt + wt[i];
          total_tat = total_tat + tat[i];
          System.out.println(" " + (i+1) + "\t\t" + bt[i] +"\t " +wt[i] +"\t\t " + tat[i]);
           }
           System.out.println("Average waiting time = " +(float)total_wt / (float)n);
           System.out.println("Average turn around time = " +(float)total_tat / (float)n);
   }
   public static void main(String[] args){
      int processes[] = { 1, 2, 3};
           int n = processes.length;
           int burst_time[] = {10, 5, 8};
           int quantum = 2;
           findavgTime(processes, n, burst_time, quantum);
   }
}
```

7. Write a program to simulate LRU Page Replacement algorithm

```
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
class LRU {
   static int pageFaults(int pages[], int n, int capacity) {
      HashSet<Integer> s = new HashSet<>(capacity);
          HashMap<Integer, Integer> indexes = new HashMap<>();
          int page_faults = 0;
          for (int i=0; i<n; i++) {
         if (s.size() < capacity) {
            if (!s.contains(pages[i])) {
               s.add(pages[i]);
               page_faults++;
                     }
            indexes.put(pages[i], i);
         }
         else{
            if (!s.contains(pages[i])) {
               int lru = Integer.MAX_VALUE, val=Integer.MIN_VALUE;
               Iterator<Integer> itr = s.iterator();
               while (itr.hasNext()) {
                  int temp = itr.next();
                              if (indexes.get(temp) < lru) {
                     lru = indexes.get(temp);
                     val = temp;
                              }
               }
               s.remove(val);
               indexes.remove(val);
               s.add(pages[i]);
               page_faults++;
                     }
            indexes.put(pages[i], i);
         }
          }
          return page_faults;
   }
   public static void main(String args[]) {
      int pages[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2};
          int capacity = 4;
          System.out.println(pageFaults(pages, pages.length, capacity));
   }
}
```

1. Interfacing LED bar with the Arduino UNO board

```
void setup(){
        pinMode(1,OUTPUT);
        pinMode(2,OUTPUT);
        pinMode(3,OUTPUT);
        pinMode(4,OUTPUT);
        Serial.begin(9600);
}
void loop(){
        digitalWrite(1,HIGH);
        delay(1000);
        digitalWrite(1,LOW);
        delay(1000);
        digitalWrite(2,HIGH);
        delay(1000);
        digitalWrite(2,LOW);
        delay(1000);
        digitalWrite(3,HIGH);
        delay(1000);
        digitalWrite(3,LOW);
        delay(1000);
        digitalWrite(4,HIGH);
        delay(1000);
        digitalWrite(4,LOW);
        delay(1000);
}
```

2. Interfacing the Piezo Buzzer with the Arduino board for Generating sound and Music

```
void setup(){
        pinMode(1,OUTPUT);
        Serial.beign(9600);
}
void loop(){
        digitalWrite(1,HIGH);
        delay(1000);
        digitalWrite(1,LOW);
        delay(2000);
}
```

3. Measuring Distance of an object using the Ultrasonic Sensor

```
const int pingPin = 7;
const int echoPin = 6;
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;
}
```

4. Interfacing Light Sensor

```
void setup(){
        pinMode(A0, INPUT);
        Serial.begin(9600);
}
void loop(){
        int read=analogRead(A0);
        Serial.print("Light Intensity is: ");
        Serial.print(read);
        delay(3000);
}
```

5. Write an application to read temperature from environment if temperature crosses threshold value then it notifies with buzzer.

```
#include<dht.h>
DHT dht;
void setup(){
        pinMode(2,OUTPUT);
        pinMode(1,OUTPUT);
        Serial.begin(9600);
}
void loop(){
        int readTemp=DHT.read11(2);
        float t=DHT.temperature;
        Serial.print("Temperature");
        Serial.print(t);
        Serial.print("C|");
        Serial.print((t*9.0)/5.0+35.0);
        if(t>100){
                digitalWrite(1,HIGH);
        }else{
                digitalWrite(1,LOW);
        }
        delay(2000);
}
```

6. 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.

```
#include<dht.h>
DHT dht;
void setup(){
        pinMode(2,OUTPUT);
        pinMode(1,OUTPUT);
        Serial.begin(9600);
}
void loop(){
        int readTemp=DHT.read11(2);
        int t=DHT.temperature;
        Serial.print("Temperature");
        Serial.print(t);
        Serial.print("C|");
        Serial.print((t*9.0)/5.0+35.0);
        if(t>100){
                digitalWrite(1,HIGH);
        }else{
                digitalWrite(1,LOW);
        }
        delay(2000);
}
```

7. Implement water level sensor.

```
void setup(){
        pinMode(A0,INPUT);
        pinMode(1,OUTPUT);
        Serial.begin(9600);
}
void loop(){
        int read=analogRead(A0);
        Serial.print("Water Level is: ");
        Serial.print(read);
        delay(3000);
        if(read>100){
                digitalWrite(1,HIGH);
        }else{
                digitalWrite(1,LOW);
        }
}
```

8. Implement DHT11/DHT22 with LED light

```
#include<dht.h>
DHT dht;
void setup(){
        pinMode(2,OUTPUT);
        pinMode(1,OUTPUT);
        Serial.begin(9600);
}
void loop(){
        int readTemp=DHT.read11 (2);
        float t=DHT.temperature;
        float h=DHT.humidity;
        Serial.print("Temperature");
        Serial.print(t);
        Serial.print("C|");
        Serial.print((t*9.0)/5.0+35.0);
        Serial.print("F");
        Serial.print("Humidity");
        Serial.print(h);
        if(t>100){
                digitalWrite(1,HIGH);
        }else{
                digitalWrite(1,LOW);
        }
        delay(2000);
}
```

9. Implement Obstacle Detector using Arduino

```
int hasObstacle=HIGH;
void setup(){
        pinMode(1,INPUT);
        pinMode(2,OUTPUT);
        Serial.beign(9600);
}
void loop(){
        hasObstacle=digitalRead(1);
        if(hasObstacle==LOW){
                Serial.print("Stop something is ahead");
                digitalWrite(2,HIGH);
        }else{
                Serial.print("Path is clear");
                digitalWrite(2,LOW);
        }
        delay(3000);
}
```

10. Implement Soil Moisture sensor using Arduino

```
void setup(){
        pinMode(A0,INPUT);
        pinMode(1,OUTPUT);
        Serial.begin(9600);
}
void loop(){
        int read=analogRead(A0);
        Serial.print("Soil Moisture: ");
        Serial.print(read);
        delay(3000);
        if(read>100){
                digitalWrite(1,HIGH);
        }else{
                digitalWrite(1,LOW);
        }
}
```