Title: Design suitable Data structures and implement Pass-I and Pass-II of a two-pass assembler for pseudo-machine. Implementation should consist of a few instructions from each category and few assembler directives. The output of Pass-I (intermediate code file and symbol table) should be input for Pass-II

Pass 1:

Code:

```
import java.util.*;
import java.io.*;
public class pass1
{
        static int address=0;
        static int sadd[]=new int[10];
      static int ladd[]=new int[10];
      public static void main(String args[])
      {
            BufferedReader br;
            OutputStream oo;
            String input=null;

            String IS[]={"ADD","SUB","MUL","MOV"};
            String UserReg[]={"AREG","BREG","CREG","DREG"};
            String AD[]={"START","END"};
            String DL[]={"DC","DS"};
            int lc=0;
            int scount=0,lcount=0;
            int flag=0,flag2=0,stored=0;

            String tokens[]=new String[30];
            String tt=null;

            String sv[]=new String[10];
            String lv[]=new String[10];

            try
            {
                  br=new BufferedReader(new FileReader("input.txt"));
                  File f = new File("IM.txt");
                  File f1 = new File("ST.txt");
                  File f2 = new File("LT.txt");
                  PrintWriter p = new PrintWriter(f);
                  PrintWriter p1 = new PrintWriter(f1);
                  PrintWriter p2 = new PrintWriter(f2);
                        int k=0,l=0;
                  while ((input = br.readLine()) != null){
                        StringTokenizer st = new StringTokenizer(input," ");
                        while (st.hasMoreTokens()){
                                    tt=st.nextToken();
                                 //System.out.println(tt);
                                          if(tt.matches("\\d*")&& tt.length() > 2)
                                          {
                                              lc=Integer.parseInt(tt);
```

```
                p.println(lc);
                address=lc-1;
            }
            else
            {
                for(int i=0;i<AD.length;i++){
                        if(tt.equals(AD[i])){
                                p.print("AD "+(i+1)+" ");
                               }
                              }
                          for(int i=0;i<IS.length;i++){
                           if(tt.equals(IS[i])){
                            p.print("IS "+(i+1)+" ");
                           }
                }
                for(int i=0;i<UserReg.length;i++)
                        {
                           if(tt.equals(UserReg[i]))
                           {
                                 p.print((i+1)+" ");
                                 flag=1;
                           }

                }
                for(int i=0;i<DL.length;i++)
                          {
                           if(tt.equals(DL[i]))
                           {
                                 p.print("DL "+(i+1)+" ");
                           }
                }
                if(tt.length()==1 && !(st.hasMoreTokens()) &&
flag==1)
                {
                 if ( Arrays.asList(sv).contains(tt) ){
                           for(int i=0;i<scount;i++){
                                if(sv[i].equals(tt)){
                                  p.print("S"+i);
                                  flag2=1;
                                }
                                  else
                                {
                                  flag2=0;
                                }
                                }

                        }
                        else
                        {
                            p.print("S"+scount);
                            sv[scount]=tt;
                            flag2=1;
```

```
                                        scount++;
                                    }
                                }
                                if(tt.length()==1 && (st.hasMoreTokens()))
                                {
                                        p.print(tt+" ");
                                        sadd[k]=address;k++;

                                }

                                if(tt.charAt(0)=='=')
                                {
                                        p.print("L"+lcount);
                                        lv[lcount]=tt;
                                        lcount++;
                                }
                                if(!st.hasMoreTokens())
                                    {
                                        p.println();
                                    }

                                    if(tt.equals("DS"))
                                    {
                                       int a=Integer.parseInt(st.nextToken());
                                       address=address+a-1;
                                       p.println();
                                    }
                                }
                            }
                            //System.out.println();
                            address++;
        } p.close();
        address--;

        for(int i=0;i<lcount;i++)
        {
                ladd[i]=address;
                address++;
        }

        for(int i=0;i<scount;i++)
        {
                p1.println(i+"\t"+sv[i]+"\t"+sadd[i]);
        }p1.close();

        for(int i=0;i<lcount;i++)
        {
                p2.println(i+"\t"+lv[i]+"\t"+ladd[i]);
        }p2.close();
}
catch(Exception e)
{
```

```
                e.printStackTrace();
            }
        }
}
```

Pass 2:

Code:

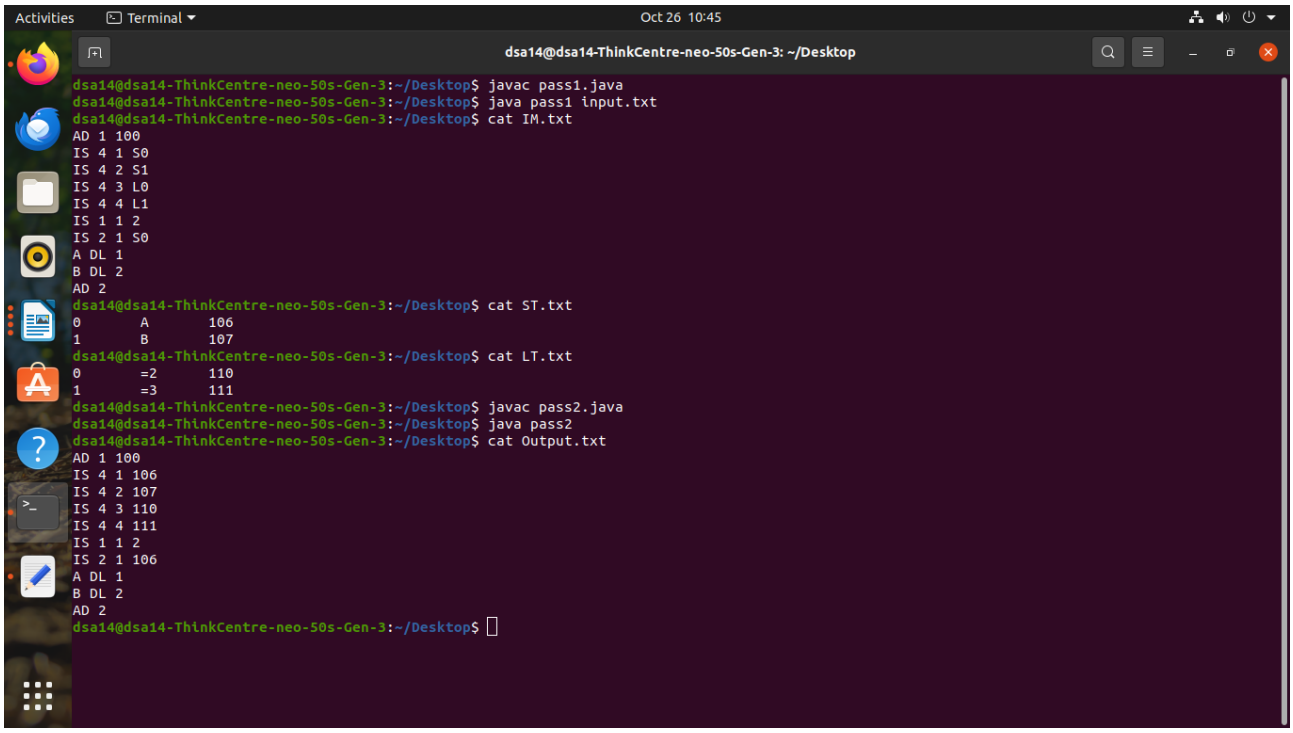
```
import java.util.*;
import java.io.*;
class pass2
{
    public static void main(String args[])
    {
        String value=null;
        BufferedReader br,br1,br2;

        String input=null;
        String t=null;
        String t1=null;
        String ss=null,ll=null;
        String pvalue,address;
        try
        {
            br=new BufferedReader(new FileReader("IM.txt"));
            File f=new File("Output.txt");
            PrintWriter p=new PrintWriter(f);

            while((input=br.readLine())!=null)
            {
                StringTokenizer st=new StringTokenizer(input," ");
                 while (st.hasMoreTokens())
                 {
                  t=st.nextToken();
              //  System.out.println(t);
                        if(t.equals("AD") || t.equals("IS") || t.equals("DL"))
                        {
                            p.print(t+" ");
                        }
                        else if(t.matches("\\d*")&& t.length() > 0 && st.hasMoreTokens())
                        {
                            p.print(t+" ");
                        }

                        else if(t.matches("\\d*")&& t.length() > 0 && !(st.hasMoreTokens()))
                        {
                            p.println(t);
                        }
                        else
                        {
                            br1=new BufferedReader(new FileReader("ST.txt"));
                                br2=new BufferedReader(new FileReader("LT.txt"));
                             if(t.charAt(0)=='S')
```

```
                                {
                                        char a;
                                        int aa;
                                        a=t.charAt(1);
                                        aa = Character.getNumericValue(a);
                                            while((t1=br1.readLine())!=null)
                                            {
                                                StringTokenizer st1=new StringTokenizer(t1,"\t");
                                                ss=st1.nextToken();
                                                int index=Integer.parseInt(ss);
                                                if(index==aa)
                                                {
                                                  pvalue=st1.nextToken();
                                                    address=st1.nextToken();
                                                    p.println(address);
                                                }
                                            }
                                }
                                else if(t.charAt(0)=='L')
                                {
                                    char a;
                                        int aa;
                                        a=t.charAt(1);
                                        aa = Character.getNumericValue(a);
                                            while((t1=br2.readLine())!=null)
                                            {
                                                StringTokenizer st2=new StringTokenizer(t1,"\t");
                                                ss=st2.nextToken();
                                                int index=Integer.parseInt(ss);
                                                if(index==aa)
                                                {
                                                  pvalue=st2.nextToken();
                                                    address=st2.nextToken();
                                                    p.println(address);
                                                }
                                            }
                                }
                                else
                                {
                                    p.print(t+" ");
                                }
                            }
                        }
                    }p.close();
                }
                catch(Exception e)
                {
                    e.printStackTrace();
                }
            }
        }
```

Input file:

START 100
MOV AREG A
MOV BREG B
MOV CREG =2
MOV DREG =3
ADD AREG BREG
SUB AREG A
A DC 05
B DS 03
END

Output:



Title: Design suitable data structures and implement Pass-I and Pass-II of a two-pass macro-processor. The output of Pass-I (MNT, MDT and intermediate code file without any macro definitions) should be input for Pass-II.

Pass 1:
java file:

```
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()),paramNo++);
```

```
                                        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();
        }
}
```

Pass 2:

java file:

```
import java.io.*;
import java.util.HashMap;
import java.util.Vector;

public class macroPass2 {
        public static void main(String[] Args) throws IOException{
                BufferedReader b1 = new BufferedReader(new FileReader("intermediate.txt"));
                BufferedReader b2 = new BufferedReader(new FileReader("mnt.txt"));
                BufferedReader b3 = new BufferedReader(new FileReader("mdt.txt"));
                BufferedReader b4 = new BufferedReader(new FileReader("kpdt.txt"));
                FileWriter f1 = new FileWriter("Pass2.txt");
                HashMap<Integer,String> aptab=new HashMap<Integer,String>();
                HashMap<String,Integer> aptabInverse=new HashMap<String,Integer>();
                HashMap<String,Integer> mdtpHash=new HashMap<String,Integer>();
                HashMap<String,Integer> kpdtpHash=new HashMap<String,Integer>();
                HashMap<String,Integer> kpHash=new HashMap<String,Integer>();
                HashMap<String,Integer> macroNameHash=new HashMap<String,Integer>();
                Vector<String>mdt=new Vector<String>();
                Vector<String>kpdt=new Vector<String>();
```

```
String s,s1;
int i,pp,kp,kpdtp,mdtp,paramNo;
while((s=b3.readLine())!=null)
        mdt.addElement(s);
while((s=b4.readLine())!=null)
        kpdt.addElement(s);
while((s=b2.readLine())!=null){
        String word[]=s.split("\t");
        s1=word[0]+word[1];
        macroNameHash.put(word[0],1);
        kpHash.put(s1,Integer.parseInt(word[2]));
        mdtpHash.put(s1,Integer.parseInt(word[3]));
        kpdtpHash.put(s1,Integer.parseInt(word[4]));
}
while((s=b1.readLine())!=null){
        String b1Split[]=s.split("\\s");
        if(macroNameHash.containsKey(b1Split[0])){
                pp= b1Split[1].split(",").length-b1Split[1].split("=").length+1;
                kp=kpHash.get(b1Split[0]+Integer.toString(pp));
                mdtp=mdtpHash.get(b1Split[0]+Integer.toString(pp));
                kpdtp=kpdtpHash.get(b1Split[0]+Integer.toString(pp));
                String actualParams[]=b1Split[1].split(",");
                paramNo=1;
                for(int j=0;j<pp;j++){
                        aptab.put(paramNo, actualParams[paramNo-1]);
                        aptabInverse.put(actualParams[paramNo-1],paramNo);
                        paramNo++;
                }
                i=kpdtp-1;
                for(int j=0;j<kp;j++){
                        String temp[]=kpdt.get(i).split("\t");
                        aptab.put(paramNo,temp[1]);
                        aptabInverse.put(temp[0],paramNo);
                        i++;
                        paramNo++;
                }
                i=pp+1;
                while(i<=actualParams.length){
                        String initializedParams[]=actualParams[i-1].split("=");
```

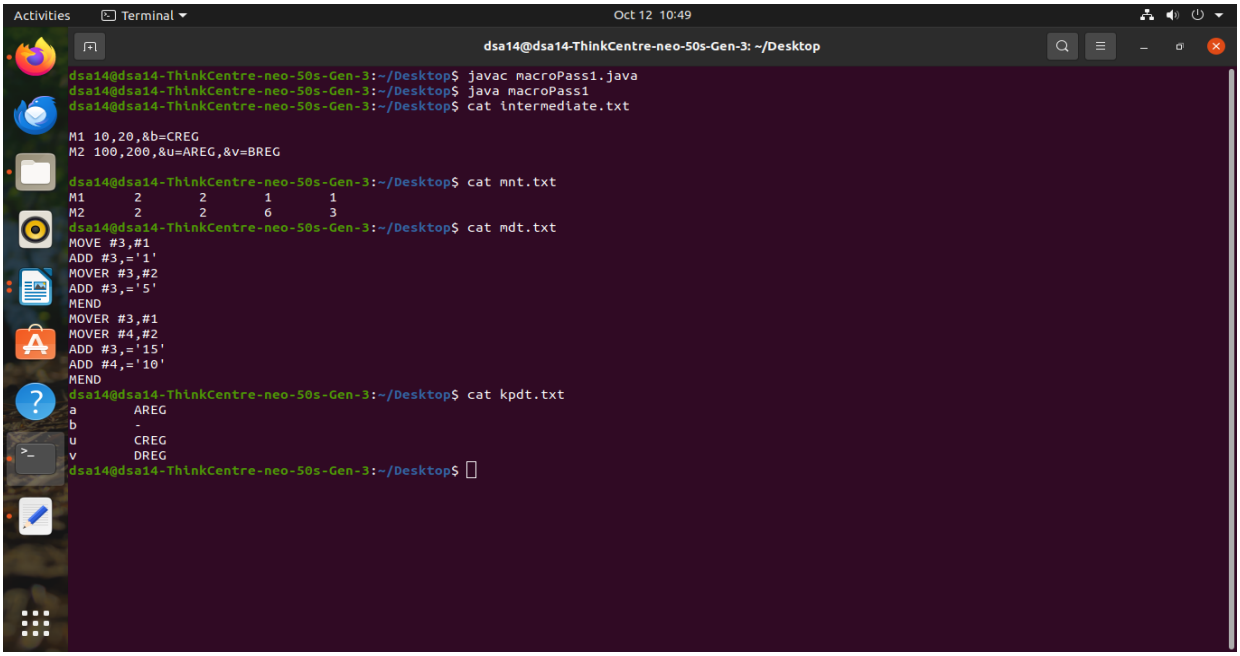
```
        aptab.put(aptabInverse.get(initializedParams[0].substring(1,initializedParams[0].length())
),initializedParams[1].substring(0,initializedParams[1].length()));
                                        i++;
                                }
                                i=mdtp-1;
                                while(mdt.get(i).compareToIgnoreCase("MEND")!=0){
                                        f1.write("+ ");
                                        for(int j=0;j<mdt.get(i).length();j++){
                                                if(mdt.get(i).charAt(j)=='#')
                                                        f1.write(aptab.get(Integer.parseInt("" +
mdt.get(i).charAt(++j))));
                                                else
                                                        f1.write(mdt.get(i).charAt(j));
                                        }
                                        f1.write("\n");
                                        i++;
                                }
                                aptab.clear();
                                aptabInverse.clear();
                        }
                        else
                                f1.write("+ "+s+"\n");
                }
                b1.close();
                b2.close();
                b3.close();
                b4.close();
                f1.close();
        }
}
```

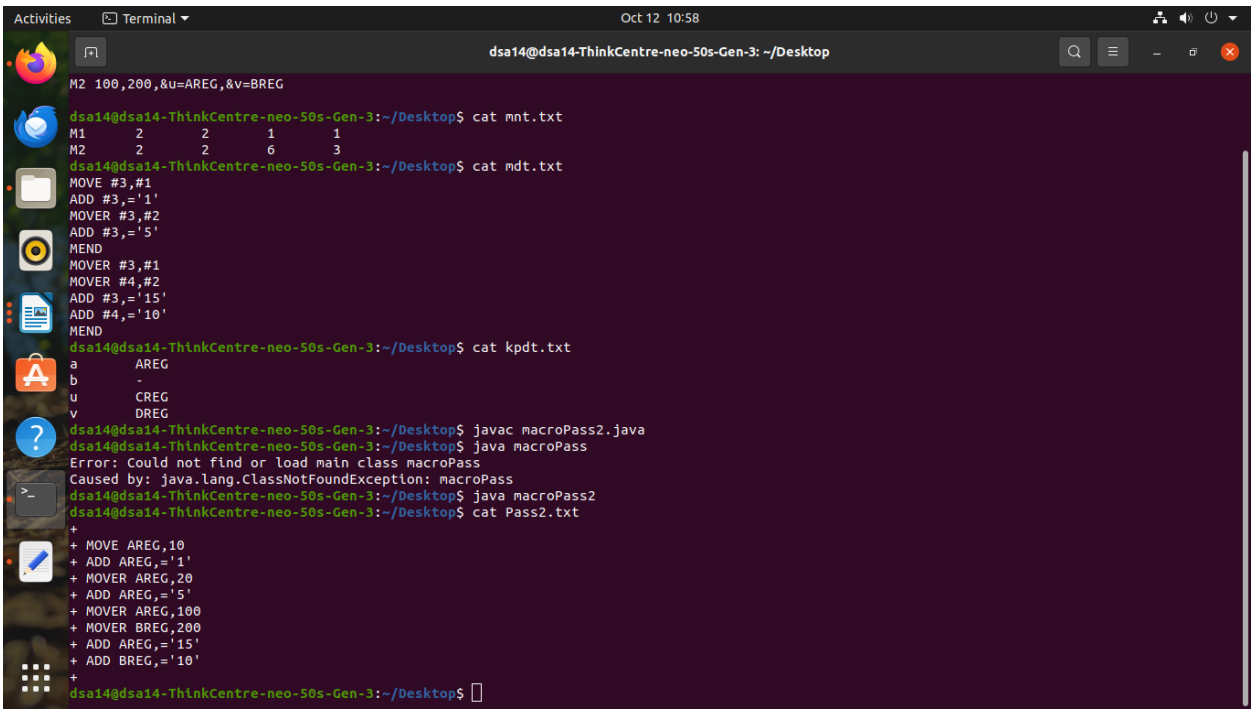
Input file:

```
MACRO M1 &x,&y,&a=AREG,&b=
MOVE &a,&x
ADD &a,='1'
MOVER &a,&y
ADD &a,='5'
```

```
MEND
MACRO M2 &p,&q,&u=CREG,&v=DREG
MOVER &u,&p
MOVER &v,&q
ADD &u,='15'
ADD &v,='10'
MEND
M1 10,20,&b=CREG
M2 100,200,&u=AREG,&v=BREG
```

OUTPUT:





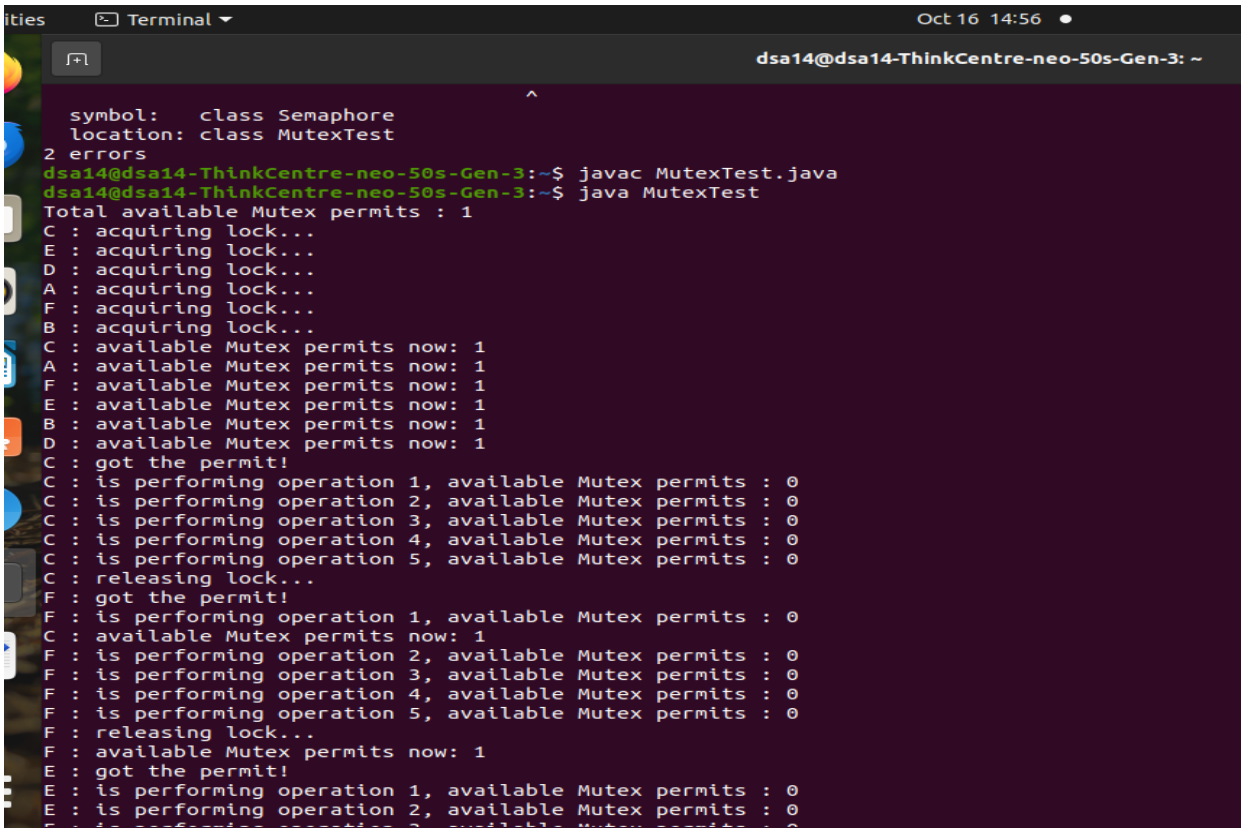
Title: Write a program to solve Classical Problems of Synchronization using Mutex and Semaphore

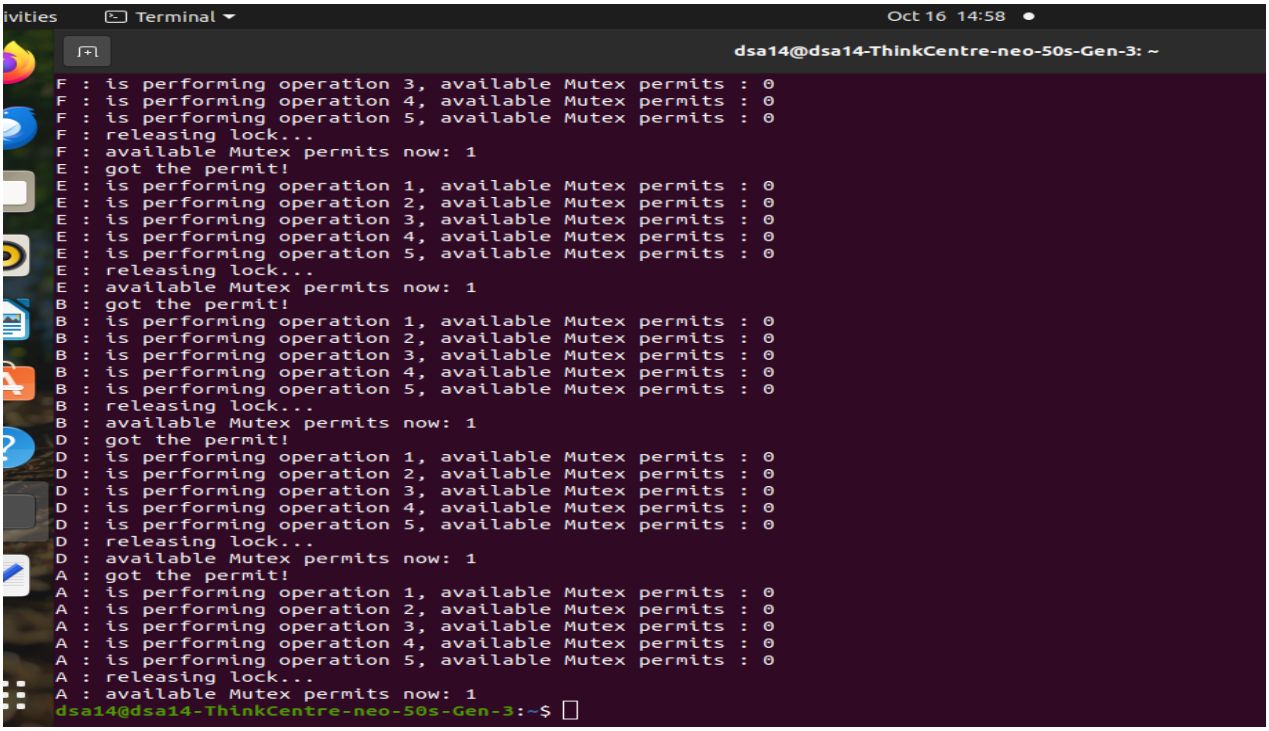
Code:

```
import java.util.concurrent.Semaphore;

public class MutexTest {
   static Semaphore semaphore = new Semaphore(1);

   static class MyLockerThread extends Thread {
      String name = "";

      MyLockerThread(String name) {
         this.name = name;
      }

      public void run() {
         try {
            System.out.println(name + " : acquiring lock...");
            System.out.println(name + " : available Mutex permits now: " +
semaphore.availablePermits());
            semaphore.acquire();
            System.out.println(name + " : got the permit!");
            try {
               for (int i = 1; i <= 5; i++) {
                  System.out.println(name + " : is performing operation " + i + ", available Mutex
permits : " + semaphore.availablePermits());
                  // sleep 1 second
                  Thread.sleep(1000);
               }
            } finally {
               // Release the permit after a successful acquire
               System.out.println(name + " : releasing lock...");
               semaphore.release();
               System.out.println(name + " : available Mutex permits now: " +
semaphore.availablePermits());
            }
         } catch (InterruptedException e) {
            e.printStackTrace();
         }
      }
   }

   public static void main(String[] args) {
      System.out.println("Total available Mutex permits : " + semaphore.availablePermits());
      MyLockerThread t1 = new MyLockerThread("A");
      t1.start();
      MyLockerThread t2 = new MyLockerThread("B");
      t2.start();
      MyLockerThread t3 = new MyLockerThread("C");
      t3.start();
      MyLockerThread t4 = new MyLockerThread("D");
```

```
        t4.start();
        MyLockerThread t5 = new MyLockerThread("E");
        t5.start();
        MyLockerThread t6 = new MyLockerThread("F");
        t6.start();

        // Join all threads to ensure they complete before the main program exits
        try {
            t1.join();
            t2.join();
            t3.join();
            t4.join();
            t5.join();
            t6.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

Output:




ivities
Terminal
Oct 16 14:58
dsa14@dsa14-ThinkCentre-neo-50s-Gen-3: ~
F : is performing operation 3, available Mutex permits : 0
F : is performing operation 4, available Mutex permits : 0
F : is performing operation 5, available Mutex permits : 0
F : releasing lock...
F : available Mutex permits now: 1
E : got the permit!
E : is performing operation 1, available Mutex permits : 0
E : is performing operation 2, available Mutex permits : 0
E : is performing operation 3, available Mutex permits : 0
E : is performing operation 4, available Mutex permits : 0
E : is performing operation 5, available Mutex permits : 0
E : releasing lock...
E : available Mutex permits now: 1
B : got the permit!
B : is performing operation 1, available Mutex permits : 0
B : is performing operation 2, available Mutex permits : 0
B : is performing operation 3, available Mutex permits : 0
B : is performing operation 4, available Mutex permits : 0
B : is performing operation 5, available Mutex permits : 0
B : releasing lock...
B : available Mutex permits now: 1
D : got the permit!
D : is performing operation 1, available Mutex permits : 0
D : is performing operation 2, available Mutex permits : 0
D : is performing operation 3, available Mutex permits : 0
D : is performing operation 4, available Mutex permits : 0
D : is performing operation 5, available Mutex permits : 0
D : releasing lock...
D : available Mutex permits now: 1
A : got the permit!
A : is performing operation 1, available Mutex permits : 0
A : is performing operation 2, available Mutex permits : 0
A : is performing operation 3, available Mutex permits : 0
A : is performing operation 4, available Mutex permits : 0
A : is performing operation 5, available Mutex permits : 0
A : releasing lock...
A : available Mutex permits now: 1
dsa14@dsa14-ThinkCentre-neo-50s-Gen-3:~$

Title: Write a program to simulate CPU Scheduling Algorithms: FCFS, SJF (Preemptive), Priority (Non-Preemptive) and Round Robin (Preemptive).

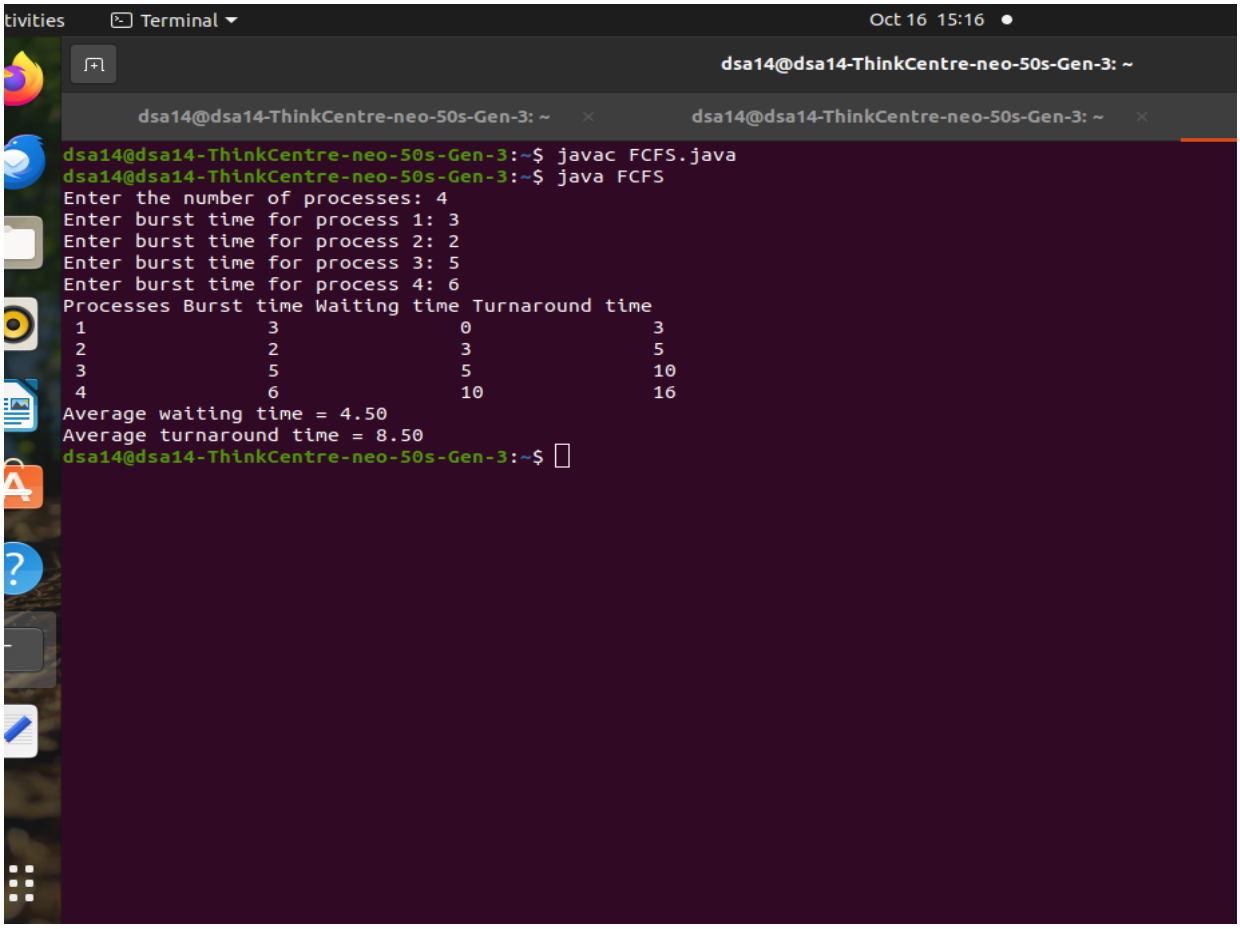
1) FCFS

Code:

```
import java.util.Scanner;

class FCFS {

   // Function to find the waiting time for all processes
   static void findWaitingTime(int processes[], int n, int bt[], int wt[]) {
      // Waiting time for the first process is 0
      wt[0] = 0;

      // Calculating waiting time
      for (int i = 1; i < n; i++) {
         wt[i] = bt[i - 1] + wt[i - 1];
      }
   }

   // Function to calculate the turnaround time
   static void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[]) {
      // Calculating turnaround time by adding bt[i] + wt[i]
      for (int i = 0; i < n; i++) {
         tat[i] = bt[i] + wt[i];
      }
   }

   // Function to calculate average time
   void findavgTime(int processes[], int n, int bt[]) {
      int wt[] = new int[n], tat[] = new int[n];
      int total_wt = 0, total_tat = 0;

      // Function to find waiting time of all processes
      findWaitingTime(processes, n, bt, wt);

      // Function to find turnaround time for all processes
      findTurnAroundTime(processes, n, bt, wt, tat);

      // Display processes along with all details
      System.out.printf("Processes Burst time Waiting time Turnaround time\n");

      // Calculate total waiting time and total turnaround time
      for (int i = 0; i < n; i++) {
         total_wt = total_wt + wt[i];
         total_tat = total_tat + tat[i];
         System.out.printf(" %d \t\t %d \t\t %d \t\t %d\n", (i + 1), bt[i], wt[i], tat[i]);
      }

      // Calculate and display average waiting time and average turnaround time
      float avg_wt = (float) total_wt / n;
      float avg_tat = (float) total_tat / n;
```

```
        System.out.printf("Average waiting time = %.2f\n", avg_wt);
        System.out.printf("Average turnaround time = %.2f\n", avg_tat);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of processes: ");
        int n = sc.nextInt();
        int processes[] = new int[n];
        int burst_time[] = new int[n];

        for (int i = 0; i < n; i++) {
            System.out.print("Enter burst time for process " + (i + 1) + ": ");
            burst_time[i] = sc.nextInt();
        }

        FCFS fcfs = new FCFS();
        fcfs.findavgTime(processes, n, burst_time);
    }
}
```

Output:



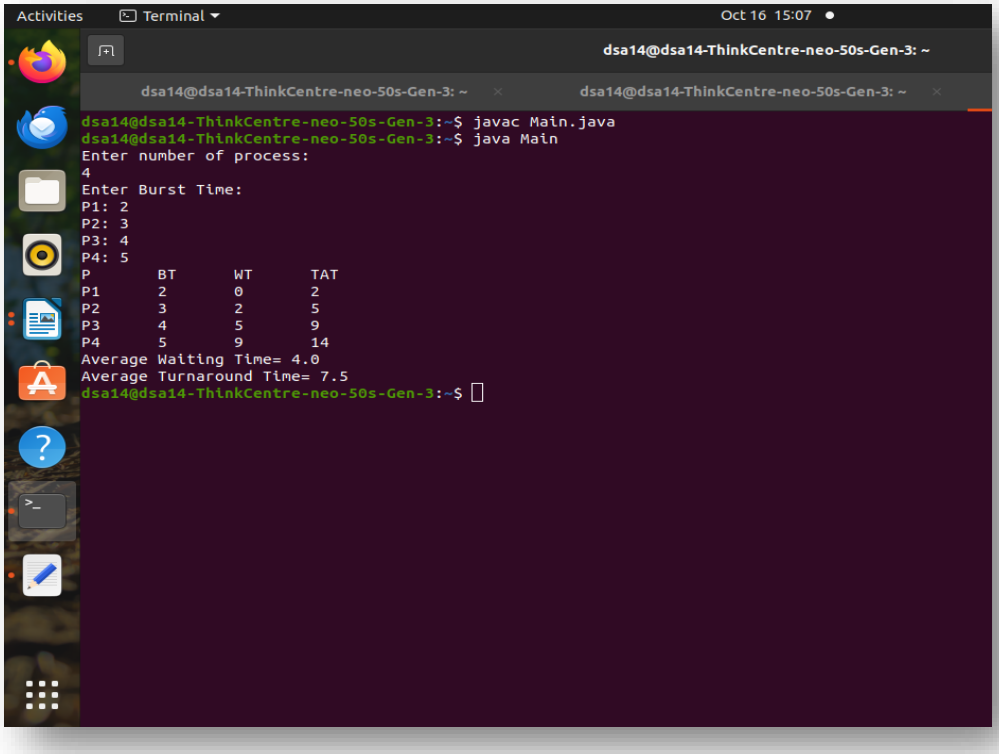
2) SJF (Preemptive):

Code:

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

public class Main {
	public static void main(String[] args)
	{
		Scanner input = new Scanner(System.in);
		int n;
		// Matrix for storing Process Id, Burst
		// Time, Average Waiting Time & Average
		// Turn Around Time.
		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++) {
			// User Input Burst Time and alloting
			// Process Id.
			System.out.print("P" + (i + 1) + ": ");
			A[i][1] = input.nextInt();
			A[i][0] = i + 1;
		}
		for (int i = 0; i < n; i++) {
			// Sorting process according to their
			// Burst Time.
			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;
		// Calculation of Waiting Times
		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;
                // Calculation of Turn Around Time and printing the
                // data.
                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);
        }
}
```

Output:



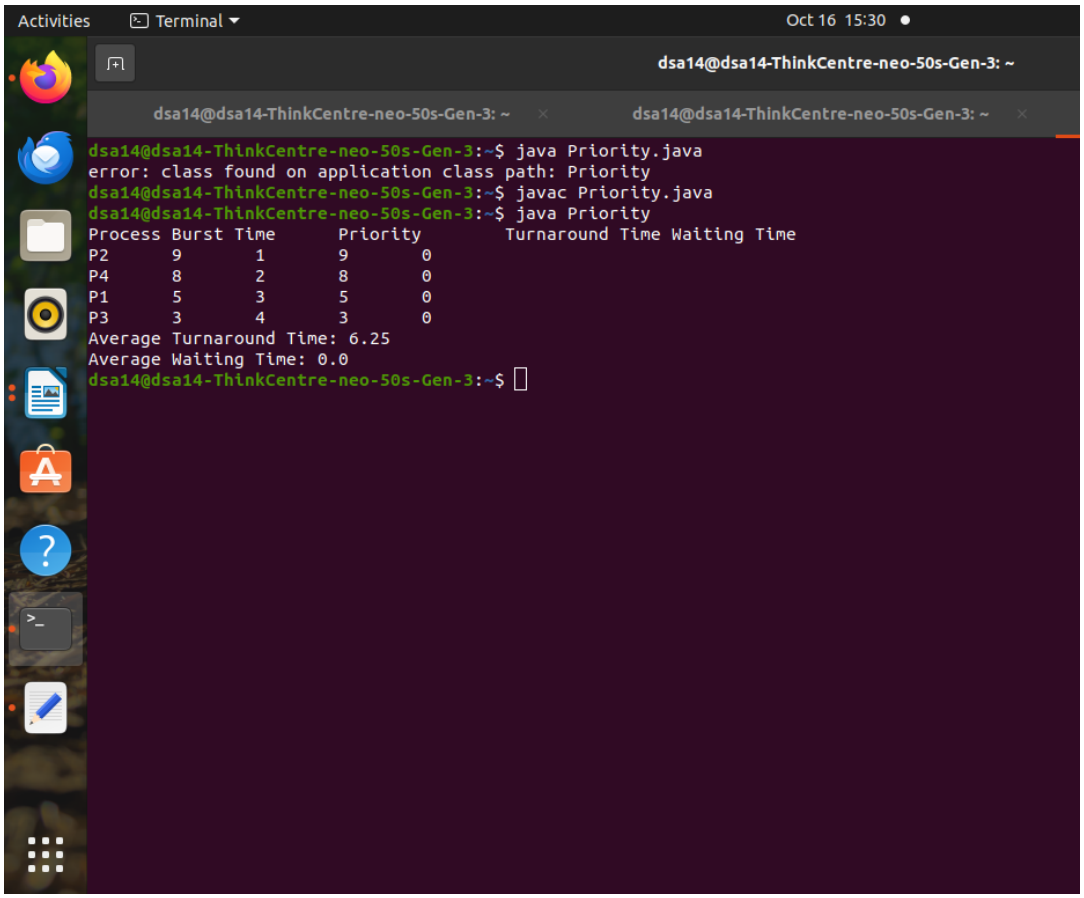
3) Priority:

Code:

```
class Priority {
   void priority(String processes[], int n, int burstTime[], int priority[]) {
      int numberOfProcess = n;
      int temp;
      String temp2;

      // Sorting processes by priority using Bubble Sort
      for (int i = 0; i < numberOfProcess - 1; i++) {
         for (int j = 0; j < numberOfProcess - 1; j++) {
            if (priority[j] > priority[j + 1]) {
               temp = priority[j];
               priority[j] = priority[j + 1];
               priority[j + 1] = temp;

               temp = burstTime[j];
               burstTime[j] = burstTime[j + 1];
               burstTime[j + 1] = temp;

               temp2 = processes[j];
               processes[j] = processes[j + 1];
               processes[j + 1] = temp2;
            }
         }
      }

      // TAT - Turn Around Time and Waiting Time
      int TAT[] = new int[numberOfProcess];
      int waitingTime[] = new int[numberOfProcess];

      TAT[0] = burstTime[0];
      waitingTime[0] = 0;

      for (int i = 1; i < numberOfProcess; i++) {
         TAT[i] = waitingTime[i - 1] + burstTime[i];
         waitingTime[i] = TAT[i] - burstTime[i];
      }

      // Calculate Average Turnaround Time and Average Waiting Time
      int totalTAT = 0;
      int totalWaitingTime = 0;

      for (int i = 0; i < numberOfProcess; i++) {
         totalTAT += TAT[i];
         totalWaitingTime += waitingTime[i];
      }

      double avgTAT = (double) totalTAT / numberOfProcess;
      double avgWaitingTime = (double) totalWaitingTime / numberOfProcess;

      // Display the results
```

```
        System.out.println("Process\tBurst Time\tPriority\tTurnaround  Time\tWaiting Time");
        for (int i = 0; i < numberOfProcess; i++) {
            System.out.println(processes[i]  + "\t" + burstTime[i]  + "\t" + priority[i]  + "\t" + TAT[i] + "\t"
+ waitingTime[i]);
        }

        System.out.println("Average Turnaround Time: " + avgTAT);
        System.out.println("Average Waiting Time:  " + avgWaitingTime);
    }

    public static void main(String[] args) {
        Priority scheduler = new Priority();

        String processes[] = {"P1", "P2", "P3", "P4"};
        int burstTime[]  = {5, 9, 3, 8};
        int priority[] = {3, 1, 4, 2};
        int n = processes.length;

        scheduler.priority(processes,  n, burstTime,  priority);
    }
}
```

Output:



4) RR:

Code:

```
class RR {
   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 += rem_bt[i];
                  wt[i] = t - bt[i];
                  rem_bt[i] = 0;
               }
            }
         }
         if (done)
            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];
   }

   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("Processes  Burst time  Waiting time  Turn around time");
      for (int i = 0; i < n; i++) {
         total_wt += wt[i];
         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 / n);
      System.out.println("Average turn around time = " + (float) total_tat / n);
   }

   public static void main(String[] args) {
      RR scheduler = new RR();
```

```
        int processes[] = {1, 2, 3, 4};
        int burstTime[] = {10, 5, 8, 4};
        int quantum = 2;
        int n = processes.length;

        scheduler.findavgTime(processes, n, burstTime, quantum);
    }
}
```

Output: