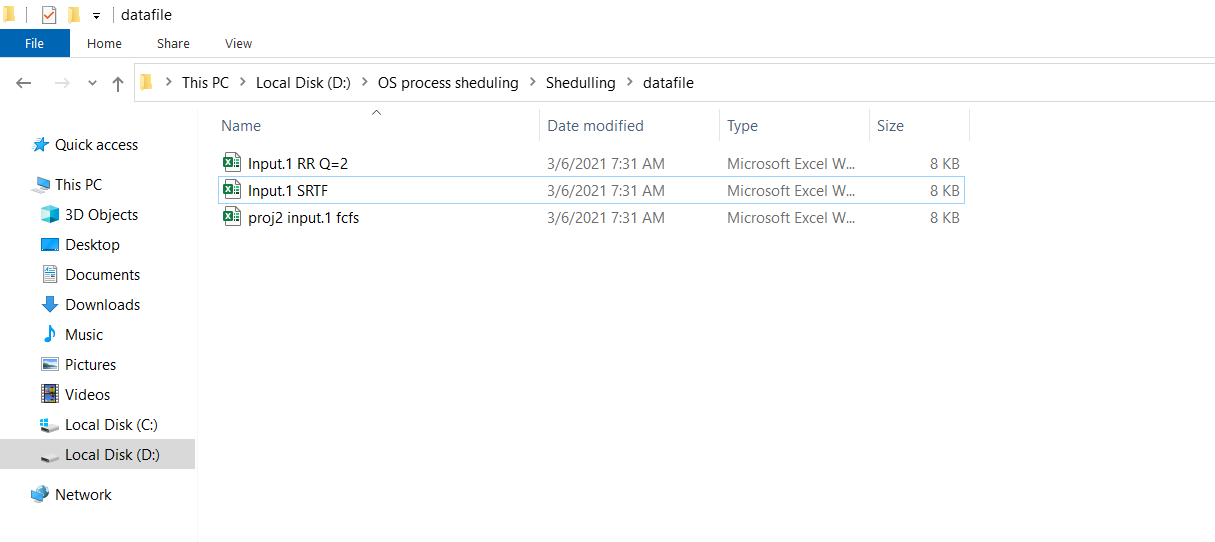
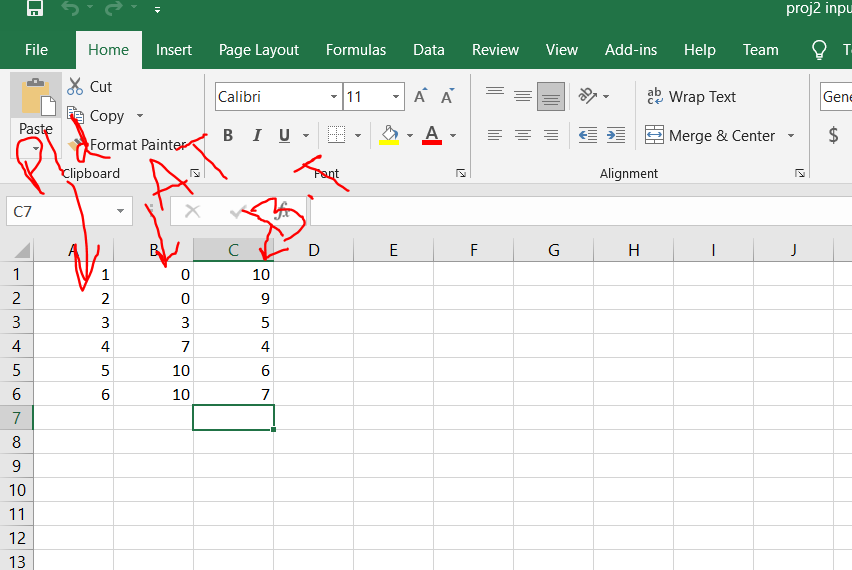
# Task 1: FCFS

# Code:

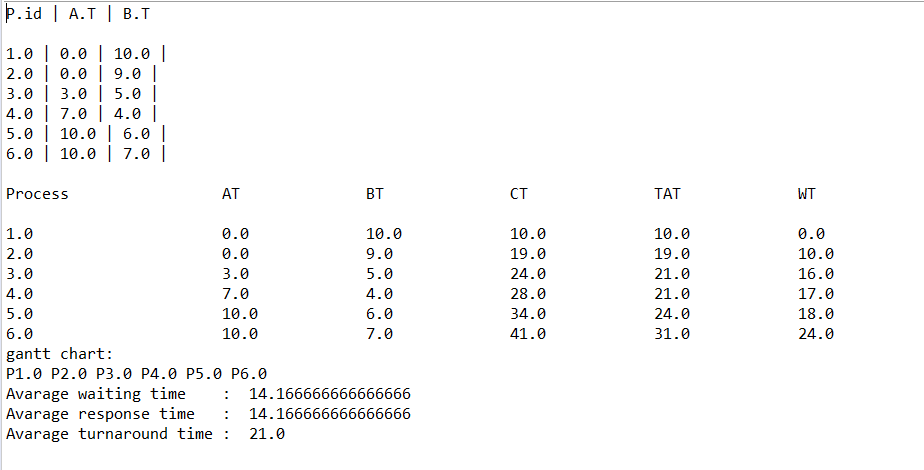
|  |
| --- |
| **package** Excel;  **import** java.io.FileInputStream;  **import** java.io.FileNotFoundException;  **import** java.io.IOException;  **import** org.apache.poi.xssf.usermodel.\*;  **public** **class** ReadingExcel {  **public** **static** **void** main(String[] args) **throws** IOException {    **int** numberOfProcesses = 6; //we have total 6 processes  **double** pid[] = **new** **double**[numberOfProcesses];  **double** bt[] = **new** **double**[numberOfProcesses];  **double** ar[] = **new** **double**[numberOfProcesses];  **double** ct[] = **new** **double**[numberOfProcesses];  **double** ta[] = **new** **double**[numberOfProcesses];  **double** wt[] = **new** **double**[numberOfProcesses];  **double** resp\_time[] = **new** **double**[numberOfProcesses];      //This part of the code is just for reading from file  String excelFilePath=".\\datafile\\proj2 input.1 fcfs.xlsx"; //path of data file for fcfs  FileInputStream inputstream=**new** FileInputStream(excelFilePath);    XSSFWorkbook workbook=**new** XSSFWorkbook(inputstream);  XSSFSheet sheet=workbook.getSheetAt(0); //XSSFSheet sheet=workbook.getSheet("Sheet1");    System.***out***.println("P.id | A.T | B.T\n");      //// USING FOR LOOP  **int** rows=sheet.getLastRowNum(); //sheet no of rows = no of processes = 6  **int** cols=sheet.getRow(1).getLastCellNum(); // no of column    **for**(**int** r=0;r<=rows;r++)  {  XSSFRow row=sheet.getRow(r); //0    **for**(**int** c=0;c<cols;c++)  {  XSSFCell cell=row.getCell(c);    **switch**(cell.getCellType())  {  **case** ***STRING***: System.***out***.print(cell.getStringCellValue()); **break**;  **case** ***NUMERIC***: {System.***out***.print(cell.getNumericCellValue());  **if**(c == 0) {  pid[r] = cell.getNumericCellValue();  }  **if**(c == 1) {  ar[r] = cell.getNumericCellValue();  }    **if**(c == 2) {  bt[r] = cell.getNumericCellValue();  }    **break**;}  **case** ***BOOLEAN***: System.***out***.print(cell.getBooleanCellValue()); **break**;  }  System.***out***.print(" | "); // this is for printing space between each data  }  System.***out***.println();  }        //this part is use for swap arrival time and pid of processors  **double** temp;  **for** (**int** i = 0; i < numberOfProcesses; i++) {  **for** (**int** j = i+1; j < numberOfProcesses; j++) {  **if**(ar[i] > ar[j]) {  temp = ar[i];  ar[i] = ar[j];  ar[j] = temp;  temp = pid[i];  pid[i] = pid[j];  pid[j] = temp;  temp = bt[i];  bt[i] = bt[j];  bt[j] = temp;  }  }  }  **double** total\_CT =0, total\_WT =0, total\_TAT = 0, total\_resp\_time=0, total\_BT = 0;    System.***out***.println();  ct[0] = bt[0] + ar[0]; // Formulas  **for**(**int** i = 1; i < numberOfProcesses; i++) {  ct[i] = ct[i - 1] + bt[i];  }  **for**(**int** i = 0; i < numberOfProcesses; i++) {  ta[i] = ct[i] - ar[i];  wt[i] = ta[i] - bt[i];  resp\_time[i] = wt[i];  }  System.***out***.println("Process\t\t\tAT\t\tBT\t\tCT\t\tTAT\t\tWT\n");  **for**(**int** i = 0; i < numberOfProcesses; i++) {  System.***out***.println(pid[i]+"\t\t\t" + ar[i] + "\t\t" + bt[i]+ "\t\t" + ct[i]+ "\t\t" + ta[i]+ "\t\t" + wt[i]);  }  System.***out***.println("gantt chart: ");  **for**(**int** i = 0; i < numberOfProcesses; i++) {  System.***out***.print("P" + pid[i] +" ");  }  System.***out***.println();    **for**(**int** i = 0; i < numberOfProcesses; i++) {  total\_WT = total\_WT + wt[i];  total\_resp\_time = total\_resp\_time + resp\_time[i] ;  total\_TAT = total\_TAT + ta[i];  }  System.***out***.println("Avarage waiting time : "+total\_WT/6);  System.***out***.println("Avarage response time : "+total\_resp\_time/6);  System.***out***.println("Avarage turnaround time : "+total\_TAT/6);      }  } |

# File





# Output

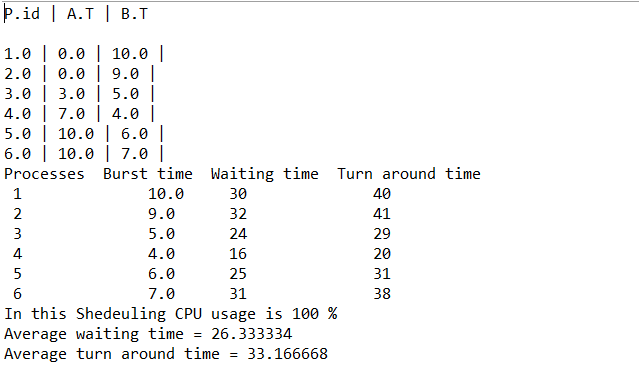


Note: CPU usage is 100 %

# Task 2 Round Robin

# Code

|  |
| --- |
| **package** Excel;  **import** java.io.FileInputStream;  **import** org.apache.poi.xssf.usermodel.XSSFCell;  **import** org.apache.poi.xssf.usermodel.XSSFRow;  **import** org.apache.poi.xssf.usermodel.XSSFSheet;  **import** org.apache.poi.xssf.usermodel.XSSFWorkbook;  **import** java.io.FileInputStream;  **import** java.io.FileNotFoundException;  **import** java.io.IOException;  **import** org.apache.poi.xssf.usermodel.\*;  //Java program for implementation of RR scheduling  **public** **class** Round\_Robin  {  // Method to find the waiting time for all  // processes  **static** **void** findWaitingTime(**double** processes[], **int** n,  **double**[] bt, **int**[] wt, **int** quantum)  {  // Make a copy of burst times bt[] to store remaining  // burst times.  **double** rem\_bt[] = **new** **double**[n];  **for** (**int** i = 0 ; i < n ; i++)  rem\_bt[i] = bt[i];    **double** t = 0; // Current time    // Keep traversing processes in round robin manner  // until all of them are not done.  **while**(**true**)  {  **boolean** done = **true**;    // Traverse all processes one by one repeatedly  **for** (**int** i = 0 ; i < n; i++)  {  // If burst time of a process is greater than 0  // then only need to process further  **if** (rem\_bt[i] > 0)  {  done = **false**; // There is a pending process    **if** (rem\_bt[i] > quantum)  {  // Increase the value of t i.e. shows  // how much time a process has been processed  t += quantum;    // Decrease the burst\_time of current process  // by quantum  rem\_bt[i] -= quantum;  }    // If burst time is smaller than or equal to  // quantum. Last cycle for this process  **else**  {  // Increase the value of t i.e. shows  // how much time a process has been processed  t = t + rem\_bt[i];    // Waiting time is current time minus time  // used by this process  wt[i] = (**int**) (t - bt[i]);    // As the process gets fully executed  // make its remaining burst time = 0  rem\_bt[i] = 0;  }  }  }    // If all processes are done  **if** (done == **true**)  **break**;  }  }    // Method to calculate turn around time  **static** **void** findTurnAroundTime(**double**[] pid, **int** n,  **double**[] bt, **int** wt[], **int** tat[])  {  // calculating turnaround time by adding  // bt[i] + wt[i]  **for** (**int** i = 0; i < n ; i++)  tat[i] = (**int**) (bt[i] + wt[i]);  }    // Method to calculate average time  **static** **void** findavgTime(**double**[] pid, **int** n, **double**[] bt,  **int** quantum)  {  **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*(pid, n, bt, wt, quantum);    // Function to find turn around time for all processes  *findTurnAroundTime*(pid, n, bt, wt, tat);    // Display processes along with all details  System.***out***.println("Processes " + " Burst time " +  " Waiting time " + " Turn around time");    // Calculate total waiting time and total turn  // around time  **for** (**int** i=0; i<n; i++)  {  total\_wt = total\_wt + wt[i];  total\_tat = total\_tat + tat[i];  System.***out***.println(" " + (i+1) + "\t\t" + bt[i] +"\t " +  wt[i] +"\t\t " + tat[i]);  }  System.***out***.println("In this Shedeuling CPU usage is 100 % ");  System.***out***.println("Average waiting time = " +  (**float**)total\_wt / (**float**)n);  System.***out***.println("Average turn around time = " +  (**float**)total\_tat / (**float**)n);  }    // Driver Method  **public** **static** **void** main(String[] args) **throws** IOException  {    **int** numberOfProcesses = 6; //we have total 6 processes  **double** pid[] = **new** **double**[numberOfProcesses];  **double** bt[] = **new** **double**[numberOfProcesses];  **double** ar[] = **new** **double**[numberOfProcesses];  **double** ct[] = **new** **double**[numberOfProcesses];  **double** ta[] = **new** **double**[numberOfProcesses];  **double** wt[] = **new** **double**[numberOfProcesses];  **double** resp\_time[] = **new** **double**[numberOfProcesses];      //This part of the code is just for reading from file  String excelFilePath=".\\datafile\\Input.1 RR Q=2.xlsx"; //path of data file for Round Robin  FileInputStream inputstream=**new** FileInputStream(excelFilePath);    XSSFWorkbook workbook=**new** XSSFWorkbook(inputstream);  XSSFSheet sheet=workbook.getSheetAt(0); //XSSFSheet sheet=workbook.getSheet("Sheet1");    System.***out***.println("P.id | A.T | B.T\n");      //// USING FOR LOOP  **int** rows=sheet.getLastRowNum(); //sheet no of rows = no of processes = 6  **int** cols=sheet.getRow(1).getLastCellNum(); // no of column    **for**(**int** r=0;r<=rows;r++)  {  XSSFRow row=sheet.getRow(r); //0    **for**(**int** c=0;c<cols;c++)  {  XSSFCell cell=row.getCell(c);    **switch**(cell.getCellType())  {  **case** ***STRING***: System.***out***.print(cell.getStringCellValue()); **break**;  **case** ***NUMERIC***: {System.***out***.print(cell.getNumericCellValue());  **if**(c == 0) {  pid[r] = cell.getNumericCellValue();  }  **if**(c == 1) {  ar[r] = cell.getNumericCellValue();  }    **if**(c == 2) {  bt[r] = cell.getNumericCellValue();  }    **break**;}  **case** ***BOOLEAN***: System.***out***.print(cell.getBooleanCellValue()); **break**;  }  System.***out***.print(" | "); // this is for printing space between each data  }  System.***out***.println();  }      // Time quantum  **int** quantum = 2;  *findavgTime*(pid, numberOfProcesses , bt, quantum);  }  } |



# Task 3: SRTF

# Code

|  |
| --- |
| **package** Excel;  **import** java.io.\*;  **import** org.apache.poi.xssf.usermodel.XSSFCell;  **import** org.apache.poi.xssf.usermodel.XSSFRow;  **import** org.apache.poi.xssf.usermodel.XSSFSheet;  **import** org.apache.poi.xssf.usermodel.XSSFWorkbook;  **public** **class** SRTF {    **public** **static** **void** main(String args[]) **throws** IOException  {  BufferedReader br = **new** BufferedReader(**new** InputStreamReader(System.***in***));          **int** numberOfProcesses = 6; //we have total 6 processes  **double** pid[] = **new** **double**[numberOfProcesses];  **double** bt[] = **new** **double**[numberOfProcesses];  **double** ar[] = **new** **double**[numberOfProcesses];  **double** ct[] = **new** **double**[numberOfProcesses];  **double** ta[] = **new** **double**[numberOfProcesses];  **double** wt[] = **new** **double**[numberOfProcesses];  **double** resp\_time[] = **new** **double**[numberOfProcesses];  //This part of the code is just for reading from file  String excelFilePath=".\\datafile\\Input.1 RR Q=2.xlsx"; //path of data file for Round Robin  FileInputStream inputstream=**new** FileInputStream(excelFilePath);  XSSFWorkbook workbook=**new** XSSFWorkbook(inputstream);  XSSFSheet sheet=workbook.getSheetAt(0); //XSSFSheet sheet=workbook.getSheet("Sheet1");  System.***out***.println("P.id | A.T | B.T\n");  //// USING FOR LOOP  **int** rows=sheet.getLastRowNum(); //sheet no of rows = no of processes = 6  **int** cols=sheet.getRow(1).getLastCellNum(); // no of column  **for**(**int** r=0;r<=rows;r++)  {  XSSFRow row=sheet.getRow(r); //0    **for**(**int** c=0;c<cols;c++)  {  XSSFCell cell=row.getCell(c);    **switch**(cell.getCellType())  {  **case** ***STRING***: System.***out***.print(cell.getStringCellValue()); **break**;  **case** ***NUMERIC***: {System.***out***.print(cell.getNumericCellValue());  **if**(c == 0) {  pid[r] = cell.getNumericCellValue();  }  **if**(c == 1) {  ar[r] = cell.getNumericCellValue();  }    **if**(c == 2) {  bt[r] = cell.getNumericCellValue();  }    **break**;}  **case** ***BOOLEAN***: System.***out***.print(cell.getBooleanCellValue()); **break**;  }  System.***out***.print(" | "); // this is for printing space between each data  }  System.***out***.println();  }          **int** n = numberOfProcesses;    **int** proc[][] = **new** **int**[n + 1][4];//proc[][0] is the AT array,[][1] - BT,[][2] - WT,[][3] - TT  **for**(**int** i = 1; i <= n; i++)  {  proc[i][0] = (**int**)ar[i-1];    proc[i][1] = (**int**)bt[i-1];  }  System.***out***.println();    //Calculation of Total Time and Initialization of Time Chart array  **int** total\_time = 0;  **for**(**int** i = 1; i <= n; i++)  {  total\_time += proc[i][1];  }  **int** time\_chart[] = **new** **int**[total\_time];    **for**(**int** i = 0; i < total\_time; i++)  {  //Selection of shortest process which has arrived  **int** sel\_proc = 0;  **int** min = 99999;  **for**(**int** j = 1; j <= n; j++)  {  **if**(proc[j][0] <= i)//Condition to check if Process has arrived  {  **if**(proc[j][1] < min && proc[j][1] != 0)  {  min = proc[j][1];  sel\_proc = j;  }  }  }    //Assign selected process to current time in the Chart  time\_chart[i] = sel\_proc;    //Decrement Remaining Time of selected process by 1 since it has been assigned the CPU for 1 unit of time  proc[sel\_proc][1]--;    //WT and TT Calculation  **for**(**int** j = 1; j <= n; j++)  {  **if**(proc[j][0] <= i)  {  **if**(proc[j][1] != 0)  {  proc[j][3]++;//If process has arrived and it has not already completed execution its TT is incremented by 1  **if**(j != sel\_proc)//If the process has not been currently assigned the CPU and has arrived its WT is incremented by 1  proc[j][2]++;  }  **else** **if**(j == sel\_proc)//This is a special case in which the process has been assigned CPU and has completed its execution  proc[j][3]++;  }  }    //Printing the Time Chart  **if**(i != 0)  {  **if**(sel\_proc != time\_chart[i - 1])  //If the CPU has been assigned to a different Process we need to print the current value of time and the name of  //the new Process  {  System.***out***.print("--" + i + "--P" + sel\_proc);  }  }  **else**//If the current time is 0 i.e the printing has just started we need to print the name of the First selected Process  System.***out***.print(i + "--P" + sel\_proc);  **if**(i == total\_time - 1)//All the process names have been printed now we have to print the time at which execution ends  System.***out***.print("--" + (i + 1));    }  System.***out***.println();  System.***out***.println();    //Printing the WT and TT for each Process  System.***out***.println("P\t WT \t TT ");  **for**(**int** i = 1; i <= n; i++)  {  System.***out***.printf("%d\t%2dms\t%2dms",i,proc[i][2],proc[i][3]);  System.***out***.println();  }    System.***out***.println();    //Printing the average WT & TT  **float** WT = 0,TT = 0;  **for**(**int** i = 1; i <= n; i++)  {  WT += proc[i][2];  TT += proc[i][3];  }  WT /= n;  TT /= n;    System.***out***.println("The Average WT is: " + WT + "ms");  System.***out***.println("The Average TT is: " + TT + "ms");  System.***out***.println("The Avarage cpu usage is 100 % for this");  }    } |

# Output

