Program:  **Round Robin Scheduling**

Input:

#include <iostream>

#include<fstream>

#include <string>

#include <conio.h>

using namespace std;

int P[50], at[50], bt[50], ct[50], qt, rqi[50], c = 0, st, flg = 0, tm = 0, node = 0, pnt = 0, btm[50], tt[50], wt[50];

float Avgtat, Avgwt;

string gc = "";

string gl = "";

void bubble\_sort(int n)

{

for (int i = 0; i < n - 1; i++)

{

for (int j = 0; j < n - i - 1; j++)

{

if (at[j] > at[j + 1])

{

swap(at[j], at[j + 1]); //swap arrival times

swap(bt[j], bt[j + 1]); //swap burst times

swap(P[j], P[j + 1]); //swap process id's

}

}

}

}

void SearchStack01(int pnt, int tm, int n)

{

for (int i = pnt + 1; i < n; i++)

{

if (at[i] <= tm)

{

rqi[node] = i + 1;

node++;

}

}

}

void SearchStack02(int pnt, int tm, int n)

{

for (int i = pnt + 1; i < n; i++)

{

int fl = 0;

for (int j = 0; j < node; j++)

{

if (rqi[j] == i + 1)

{

fl++;

}

}

if (at[i] <= tm && fl == 0 && btm[i] != 0)

{

rqi[node] = i + 1;

node++;

}

}

}

void AddQue(int pnt)

{

rqi[node] = pnt + 1;

node++;

}

void gt(int Pr, int duration)

{

gl += ".";

gc += "|P" + to\_string(Pr) + "";

for (int i = 0; i < duration; i++)

{

gl += "-";

}

for (int i = 0; i < duration - 1; i++)

{

gc += " ";

}

gl += ".";

}

int main()

{

int n=0,qt;

ifstream infile("data.txt",ios::in);

if (!infile.is\_open())

{

cout << "No file";

}

infile >> qt;

while (1)

{

infile >> P[n] >> at[n] >> bt[n];

n++;

if (infile.eof())

break;

}

//Temporary array for burst time

infile.close();

for (int x = 0; x < n; x++)

{

btm[x] = bt[x];

}

cout << "\n==============Inputs==================\n";

cout << "\nThe time quantum is: " << qt << endl;

cout << "\nProcesses\tArrival Time\tBrust Time\n";

for (int i = 0; i < n; i++)

{

cout << " " << P[i] << "\t\t" << at[i] << "\t\t" << bt[i];

cout << "\n";

}

bubble\_sort(n);

do {

if (flg == 0)

{

st = at[0];

if (btm[0] <= qt)

{

tm = st + btm[0];

gt(P[0], btm[0]);

btm[0] = 0;

SearchStack01(pnt, tm, n);

}

else

{

btm[0] = btm[0] - qt;

tm = st + qt;

gt(P[0], qt);

SearchStack01(pnt, tm, n);

AddQue(pnt);

}

}

else

{

pnt = rqi[0] - 1;

st = tm;

for (int i = 0; i < node && node != 1; i++)

{

rqi[i] = rqi[i + 1];

}

node--;

if (btm[pnt] <= qt)

{

tm = st + btm[pnt];

gt(P[pnt], btm[pnt]);

btm[pnt] = 0;

SearchStack02(pnt, tm, n);

}

else

{

btm[pnt] = btm[pnt] - qt;

tm = st + qt;

gt(P[pnt], qt);

SearchStack02(pnt, tm, n);

AddQue(pnt);

}

}

if (btm[pnt] == 0)

{

ct[pnt] = tm;

}

flg++;

} while (node != 0);

cout << "\nGantt chart: " << endl;

cout << gl << "\n" << gc << "\n" << gl << "\n";

cout << "\n==============Table==================\n";

cout << "\nProcesses\tArrival Time\tBrust Time\tFinish Time\tTurn around Time\tWaiting Time\n";

for (int i = 0; i < n; i++)

{

tt[i] = ct[i] - at[i];

wt[i] = tt[i] - bt[i];

cout << " " << P[i] << "\t\t" << at[i] << "\t\t" << bt[i] << "\t\t" << ct[i] << "\t\t" << tt[i] << "\t\t\t" << wt[i];

cout << "\n";

Avgtat = Avgtat + tt[i];

Avgwt = Avgwt + wt[i];

}

Avgtat = Avgtat / n;

Avgwt = Avgwt / n;

cout << "\nAverage Turn Around Time : " << Avgtat;

cout << "\nAverage Waiting Time : " << Avgwt;

\_getch();

}

Output:



