|  |  |
| --- | --- |
|  | **DEPARTMENT OF COMPUTER ENGINEERING** |

**PBLE 1**

|  |  |
| --- | --- |
| Semester | S.E. Semester IV – Computer Engineering |
| Subject | Operating System |
| Subject Professor In-charge | Prof. Pankaj Vanvari |
| Assisting Teachers | Prof. Pankaj Vanvari |

|  |  |
| --- | --- |
| Roll Numbers | Name of Students |
| 21102A0014 | Deep Salunkhe |
| 21102A0003 | Omkar Patil |
| 21102A0005 | Pranav Redij |

**Description:**

**Code:**

#include <iostream>

#include <list>

#include <vector>

#include <queue>

#include <algorithm>

using namespace std;

struct ProcessControlBlock

{

    int pid;

    int BT;

    int AT;

    int CT;

    int waitingtime;

    int TAT;

    int FBT;

    char type;

};

 vector<ProcessControlBlock> PCB;

 int n;

int sortByarrival(ProcessControlBlock a, ProcessControlBlock b)

{

    if (a.AT < b.AT)

        return 1;

    else

        return 0;

}

int sortByburst(ProcessControlBlock a, ProcessControlBlock b)

{

    if (a.BT < b.BT)

        return 1;

    else

        return 0;

}

int sortByRemaining(ProcessControlBlock a, ProcessControlBlock b)

{

    if (a.FBT < b.FBT)

        return 1;

    else

        return 0;

}

void FCFS()

{

   vector<ProcessControlBlock> Arrived;

    queue<ProcessControlBlock> running;

    vector<ProcessControlBlock> ans;

    vector<ProcessControlBlock> Active;

    vector<ProcessControlBlock>blocked;

     sort(PCB.begin(),PCB.end(),sortByarrival);

    for(int i=0;i<n;i++)

    {

        PCB[i].FBT=PCB[i].BT;

    }

    int envruntime=30;

    int inpcb=0;

    int time=0;

    int currBT=0;

    int index=0;

    float avgTAT=0;

    float avgWT=0;

    cout<<"For how much time the env must be active";

   cin>>envruntime;

    while(envruntime--)

    {

         while(time==PCB[inpcb].AT)

        {

            Arrived.push\_back(PCB[inpcb]);

            inpcb++;

        }

        if(!blocked.empty())

        {

            running.push(blocked[0]);

           cout<< blocked[0].pid<<endl;;

            blocked.erase(blocked.begin()+0);

        }

        if(running.empty() && !Arrived.empty())

        {

*// cout<<Arrived.size()<<endl;*

            running.push(Arrived[0]);

           cout<< Arrived[0].pid<<endl;;

            Arrived.erase(Arrived.begin()+0);

        }

*// if(running.front().FBT>Arrived[0].FBT)*

*// {*

*//  running.pop();*

*//  running.push(Arrived[0]);*

*// }*

        if(running.front().FBT==0)

        {

        ans.push\_back(running.front());

            ans[index].waitingtime=time-ans[index].BT-ans[index].AT;

            ans[index].CT=time;

            ans[index].TAT=time-ans[index].AT;

*// cout<<time<<endl;*

            index=index+1;

            running.pop();

*// cout<<Arrived.size()<<endl;*

            time--;

        }

        while(!Arrived.empty() && Arrived[0].type=='b'&& Arrived[0].AT<=time)

        {

            blocked.push\_back(running.front());

            running.pop();

            running.front()=Arrived[0];

             Arrived.erase(Arrived.begin()+0);

             cout<<running.front().pid<<endl;

        }

*// cout<<time<<endl;*

        time++;

        running.front().FBT--;

    }

     cout<<"\t\tSJF"<<endl;

    cout<<"PID"<<"\t"<<"BT"<<"\t"<<"AT"<<"\t"<<"WT"<<"\t"<<"TAT"<<endl;

    for(int i=0;i<n;i++)

    {

         cout<<ans[i].pid<<" \t"<<ans[i].BT<<"  \t"<<ans[i].AT<<"  \t"<<ans[i].waitingtime<<"\t"<<ans[i].TAT<<endl;

         avgTAT=avgTAT+ans[i].TAT;

         avgWT=avgWT+ans[i].waitingtime;

    }

    cout << "Avg Waiting for FCFS:" << (avgWT/ n) << endl;

    cout << "Avg TAT for FCFS:    " << (avgTAT/ n) << endl;

}

void SJF()

{

    vector<ProcessControlBlock> Arrived;

    queue<ProcessControlBlock> running;

    vector<ProcessControlBlock> ans;

    vector<ProcessControlBlock> Active;

     sort(PCB.begin(),PCB.end(),sortByarrival);

    for(int i=0;i<n;i++)

    {

        PCB[i].FBT=PCB[i].BT;

    }

    int envruntime=30;

    int inpcb=0;

    int time=0;

    int currBT=0;

    int index=0;

    float avgTAT=0;

    float avgWT=0;

    cout<<"For how much time the env must be active";

   cin>>envruntime;

    while(envruntime--)

    {

         while(time==PCB[inpcb].AT)

        {

            Arrived.push\_back(PCB[inpcb]);

            inpcb++;

            sort(Arrived.begin(),Arrived.end(),sortByburst);

        }

        if(running.empty() && !Arrived.empty())

        {

*// cout<<Arrived.size()<<endl;*

            running.push(Arrived[0]);

           cout<< Arrived[0].pid<<endl;;

            Arrived.erase(Arrived.begin()+0);

        }

*// if(running.front().FBT>Arrived[0].FBT)*

*// {*

*//  running.pop();*

*//  running.push(Arrived[0]);*

*// }*

        if(running.front().FBT==0)

        {

        ans.push\_back(running.front());

            ans[index].waitingtime=time-ans[index].BT-ans[index].AT;

            ans[index].CT=time;

            ans[index].TAT=time-ans[index].AT;

*// cout<<time<<endl;*

            index=index+1;

            running.pop();

*// cout<<Arrived.size()<<endl;*

            time--;

        }

*// cout<<time<<endl;*

        time++;

        running.front().FBT--;

    }

     cout<<"\t\tSJF"<<endl;

    cout<<"PID"<<"\t"<<"BT"<<"\t"<<"AT"<<"\t"<<"WT"<<"\t"<<"TAT"<<endl;

    for(int i=0;i<n;i++)

    {

         cout<<ans[i].pid<<" \t"<<ans[i].BT<<"  \t"<<ans[i].AT<<"  \t"<<ans[i].waitingtime<<"\t"<<ans[i].TAT<<endl;

         avgTAT=avgTAT+ans[i].TAT;

         avgWT=avgWT+ans[i].waitingtime;

    }

    cout << "Avg Waiting for FCFS:" << (avgWT/ n) << endl;

    cout << "Avg TAT for FCFS:    " << (avgTAT/ n) << endl;

}

void SRTN()

{

    vector<ProcessControlBlock> Arrived;

    queue<ProcessControlBlock> running;

    vector<ProcessControlBlock> ans;

    vector<ProcessControlBlock> Active;

    sort(PCB.begin(),PCB.end(),sortByarrival);

    for(int i=0;i<n;i++)

    {

        PCB[i].FBT=PCB[i].BT;

    }

    int envruntime=30;

    int inpcb=0;

    int time=0;

    int currBT=0;

    int index=0;

    float avgTAT=0;

    float avgWT=0;

    cout<<"For how much time the env must be active";

   cin>>envruntime;

    while(envruntime--)

    {

         while(time==PCB[inpcb].AT)

        {

            Arrived.push\_back(PCB[inpcb]);

            inpcb++;

            sort(Arrived.begin(),Arrived.end(),sortByRemaining);

        }

        if(running.empty())

        {

            cout<<Arrived.size()<<endl;

            running.push(Arrived[0]);

        }

        if(running.front().FBT>Arrived[0].FBT)

        {

            running.pop();

            running.push(Arrived[0]);

        }

        if(running.front().FBT==0)

        {

        ans.push\_back(running.front());

            ans[index].waitingtime=time-ans[index].BT-ans[index].AT;

            ans[index].CT=time;

            ans[index].TAT=time-ans[index].AT;

            cout<<time<<endl;

            index=index+1;

            running.pop();

            Arrived.erase(Arrived.begin()+0);

            cout<<Arrived.size()<<endl;

*// time--;*

        }

        cout<<time<<endl;

        time++;

        running.front().FBT--;

        Arrived[0].FBT--;

    }

     cout<<"\t\tSRTN"<<endl;

    cout<<"PID"<<"\t"<<"BT"<<"\t"<<"AT"<<"\t"<<"WT"<<"\t"<<"TAT"<<endl;

    for(int i=0;i<n;i++)

    {

         cout<<ans[i].pid<<" \t"<<ans[i].BT<<"  \t"<<ans[i].AT<<"  \t"<<ans[i].waitingtime<<"\t"<<ans[i].TAT<<endl;

         avgTAT=avgTAT+ans[i].TAT;

         avgWT=avgWT+ans[i].waitingtime;

    }

    cout << "Avg Waiting for FCFS:" << (avgWT/ n) << endl;

    cout << "Avg TAT for FCFS:    " << (avgTAT/ n) << endl;

}

void RR()

{

     vector<ProcessControlBlock> Arrived;

    queue<ProcessControlBlock> running;

    vector<ProcessControlBlock> ans;

    vector<ProcessControlBlock> Active;

     sort(PCB.begin(), PCB.end(), sortByarrival);

  for(int i=0;i<n;i++)

    {

        PCB[i].FBT=PCB[i].BT;

    }

    int envruntime = 30;

    int inpcb = 0;

    int time = 0;

    int currBT = 0;

    int index = 0;

    int timequantum = 0;

    float avgTAT = 0;

    float avgWT = 0;

    int temp;

    cout << "Enter the Time quantum: ";

    cin >> timequantum;

    cout << "For how much time the env must be active: ";

    cin >> envruntime;

    while (envruntime)

    {

        while (time == PCB[inpcb].AT && inpcb < n)

        {

            Arrived.push\_back(PCB[inpcb]);

            inpcb++;

        }

        if (running.empty() && !Arrived.empty())

        {

            running.push(Arrived[0]);

            Arrived.erase(Arrived.begin() + 0);

        }

        if (!running.empty())

        {

            running.front().FBT--;

            time++;

            if (running.front().FBT == 0)

            {

                ans.push\_back(running.front());

                ans[index].CT = time;

                ans[index].TAT = time - ans[index].AT;

                ans[index].waitingtime = ans[index].TAT - ans[index].BT;

                index++;

*// cout << time << "c" << endl;*

*// for (int i = 0; i < Arrived.size(); i++)*

*// {*

*//     cout << Arrived[i].pid << endl;*

*// }*

                running.pop();

                if (!Arrived.empty())

                {

                    running.push(Arrived[0]);

                    Arrived.erase(Arrived.begin() + 0);

*// cout<<running.front().FBT<<endl;*

                }

*//  time++;*

            }

            else if (time % timequantum == 0)

            {

                while (time == PCB[inpcb].AT && inpcb < n)

                {

                    Arrived.push\_back(PCB[inpcb]);

                    inpcb++;

                }

                ProcessControlBlock temp = running.front();

                running.pop();

                Arrived.push\_back(temp);

*// cout << time << "p" << endl;*

*// time--;*

            }

        }

        envruntime--;

    }

*// sort(ans.begin(),ans.end(),sortByarrival);*

    cout << "\t\tRound Robin" << endl;

    cout << "PID"

         << "\t"

         << "BT"

         << "\t"

         << "AT"

         << "\t"

         << "WT"

         << "\t"

         << "TAT"

         << "\t"

         << "CT" << endl;

    for (int i = 0; i < n; i++)

    {

        cout << ans[i].pid << " \t" << ans[i].BT << "  \t" << ans[i].AT << "  \t" << ans[i].waitingtime << "\t" << ans[i].TAT << "\t" << ans[i].CT << endl;

        avgTAT = avgTAT + ans[i].TAT;

        avgWT = avgWT + ans[i].waitingtime;

    }

    cout << "Avg Waiting for FCFS:" << (avgWT / n) << endl;

    cout << "Avg TAT for FCFS:    " << (avgTAT / n) << endl;

}

int main()

{

    int noOfProcess;

    cout << "ENTER THE NO. PROCESS =>"

         << "\t";

    cin >> noOfProcess;

    cout << endl;

     n = noOfProcess;

*// vector<ProcessControlBlock> PCB;*

    for (int i = 0; i < n; i++)

    {

        ProcessControlBlock dummy;

        cout << "Enter the pid          " << i + 1 << "\t";

        cin >> dummy.pid;

        cout << "Enter the Burst Time   " << i + 1 << "\t";

        cin >> dummy.BT;

        cout << "Enter the Arrival Time " << i + 1 << "\t";

        cin >> dummy.AT;

        cout << "Enter the type" << i + 1 << "\t";

        cin>>dummy.type;

        cout << endl;

        PCB.push\_back(dummy);

    }

    while (1)

    {

        cout << "Select the shedulilng algorithm" << endl;

        cout << "1:FCFS" << endl;

        cout << "2:SJF" << endl;

        cout << "3:SRTN" << endl;

        cout << "4:RR" << endl;

        cout << "5:Exit" << endl;

        cout << "Your Choice" << endl;

        int choice;

        cin >> choice;

        switch (choice)

        {

        case 1:

            FCFS();

            break;

        case 2:

            SJF();

            break;

        case 3:

            SRTN();

            break;

        case 4:

            RR();

            break;

        case 5:

            return 0;

        default:

            cout << "Invalid input" << endl;

        }

    }

}

**Result/ Output:**

**Text

Description automatically generated**