OS LAB 2

NAME: Aditya Anand

ROLL NO.: 20124009

BRANCH: IT

|  |  |  |  |
| --- | --- | --- | --- |
| S No. | Title | Date of Implementation | Remarks |
| 1 | Program to implement Round Robin Process of CPU Scheduling | 17-01-2022 |  |

ROUND ROBIN CPU SCHEDULING

CRITERIA: Time Quantum

MODE: Pre-emptive

GIVEN: Time Quantum and list of processes with their arrival and burst time.

CODE:

#include<bits/stdc++.h>

using namespace :: std;

class process{

    public:

        int id;

        int arrivalTime;

        int burstTime;

        int remainingBurstTime;

        int completionTime;

        int TAT;

        int WT;

        int RT;

        bool isRunning;

};

void round\_robin\_scheduling(vector<process> &v, int quantum){

    queue<process> q;

    stack<process> s;

    int cur\_time = 0;

    int id=0;

    while(true){

        // Add all the processes that have arrived to the ready queue.

        while(id<v.size() && v[id].arrivalTime<=cur\_time){

            q.push(v[id]);

            id++;

        }

        // Add the last process from running queue at the end of ready queue if it is not completed.

        if(!s.empty()){

            process p = s.top();

            if(p.remainingBurstTime>0){

                q.push(p);

            }

        }

        // If ready queue is empty => no process to be completed => stop.

        if(q.empty()){

            break;

        }

        // Pick the front process from the ready queue for processing by the CPU.

        process rning\_proc = q.front();

        q.pop();

        // Store the Response time for a process the first time it reaches the CPU.

        if(rning\_proc.isRunning==false){

            v[rning\_proc.id].RT = cur\_time-v[rning\_proc.id].arrivalTime;

            rning\_proc.isRunning=true;

        }

        // If the remaining burst time > quantum, the process is not complete (CONTEXT SWITCHING)

        if(rning\_proc.remainingBurstTime>=quantum){

            rning\_proc.remainingBurstTime-=quantum;

            cur\_time+=quantum;

        }

        // Process is complete

        else{

            cur\_time+=rning\_proc.remainingBurstTime;

            rning\_proc.remainingBurstTime=0;

        }

        s.push(rning\_proc);

        // If process is complete, store the completeion time for the process

        if(rning\_proc.remainingBurstTime==0){

            v[rning\_proc.id].completionTime = cur\_time;

        }

    }

    return;

}

int main(){

    cout<<"ROUND ROBIN CPU SCHEDULING ALGORITHM C++ IMPLEMENTATION\n";

    cout<<"Name: Aditya Anand\tRoll No.:20124009\t Branch: IT\n\n\n";

    int quantum = 0;

    int n=0;

    cout<<"Enter the number of processes: ";

    cin>>n;

    cout<<"Enter value of time quantum: ";

    cin>>quantum;

    cout<<"Enter the arrival times and burst times of "<<n<<" processes: \n";

    vector<process> v(n);

    for(int i=0; i<n; i++){

        cin>>v[i].arrivalTime>>v[i].burstTime;

        v[i].id = i;

        v[i].isRunning = false;

        v[i].remainingBurstTime = v[i].burstTime;

    }

    round\_robin\_scheduling(v, quantum);

    int t\_TAT=0;

    int t\_CT=0;

    for(int i=0; i<n; i++){

        v[i].TAT = v[i].completionTime-v[i].arrivalTime;

        v[i].WT = v[i].TAT-v[i].burstTime;

        t\_TAT+=v[i].TAT;

        t\_CT+=v[i].completionTime;

    }

    for(auto p:v){

        cout<<"Process: "<<p.id<<"\tArrival Time:"<<p.arrivalTime<<"\tBurst Time:"<<p.burstTime<<"\tCompletion Time:"<<p.completionTime;

        cout<<"\tTurn Around Time:"<<p.TAT<<"\tWaiting Time:"<<p.WT<<"\tResponse Time:"<<p.RT<<"\n";

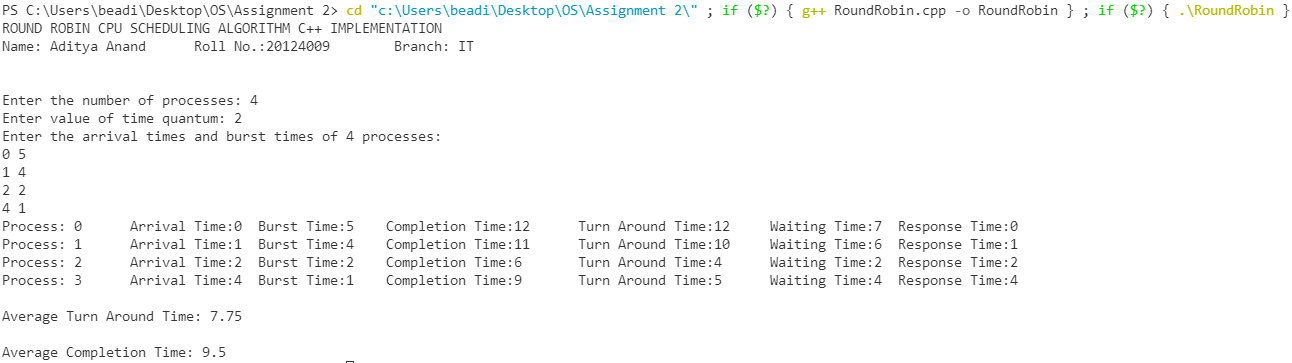
    }

    cout<<"\nAverage Turn Around Time: "<<(float)((1.0\*t\_TAT)/(1.0\*n))<<"\n";

    cout<<"\nAverage Completion Time: "<<(float)((1.0\*t\_CT)/(1.0\*n))<<"\n";

    return 0;

}

RESULT: