**Student Name: KAPTAN YADAV**

**Student ID: 11705472**

**Email Address: kaptanyadav007@gmail.com**

**GitHub Link:**

**Description:**

Write a program for multilevel queue scheduling algorithm. There must be three queues generated. There must be specific range of priority associated with every queue. Now prompt the user to enter number of processes along with their priority and burst time. Each process must occupy the respective queue with specific priority range according to its priority. Apply Round robin algorithm with quantum time 4 on queue with highest priority range. Apply priority scheduling algorithm on the queue with medium range of priority and First come first serve algorithm on the queue with lowest range of priority. Each and every queue should get a quantum time of 10 seconds. Cpu will keep on shifting between queues after every 10 seconds  i.e. to apply round robin algorithm OF 10 seconds on over all structure.

Calculate Waiting time and turnaround time for every process. The input for number of processes  should be given by the user.

**Algorithm:**

**FCFS:**

First Come First Served (**FCFS**) is a Non-Preemptive scheduling **algorithm**. FIFO (First In First Out) strategy assigns priority to process in the order in which they request the processor. The process that requests the CPU first is allocated the CPU first.

**RoundRobin**

Round-robin is one of the algorithms employed by process and network schedulers in computing. As the term is generally used, time slices are assigned to each process in equal portions and in circular order, handling all processes without priority.

**Snippet:**

#include<stdio.h>

#include <stdlib.h>

struct process{

int priority;

int burst\_time;

int pid;

int waiting\_time;

int turnaround\_time;

int remaining\_time;

int arrival\_time;};

void getInput();

void calcWaitingTime(struct process \*q,int);

void calcTurnAroundTime(struct process \*q,int);

void printQueue(struct process \*q,int size);

void RoundRobin();

void PrioSorting();

void FCFS();

void printQueueI(struct process);

void printQueue(struct process \*,int);

// About queues

//Queues

//Q1 = Batch Process having Low Priority with FCFS Algorithm

//Q2 = Interactive Process having Medium Priority with Priority Based Sorting

//Q3 = System Process having High Priority with Round Robin Algorith Quantum Time=4

int q1\_n=0,q2\_n=0,q3\_n=0,n=0; //N=Total Process

struct process \*q1,\*q2,\*q3;

//Time Quantum

int time\_quantum = 4;

void getInput(){

printf("\n Total Number of Process:\t");

scanf("%d",&n);

//Allocatig Memory

q1 = (struct process \*)malloc(n\*sizeof(struct process));

q2 = (struct process \*)malloc(n\*sizeof(struct process));

q3 = (struct process \*)malloc(n\*sizeof(struct process));

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

struct process p;

printf("\n\t\tProcess %d\n=============================================\n\n",i+1);

p.arrival\_time = (rand())%(n+1);

printf("PId:\t");

scanf("%d",&p.pid);

printf("Priority (1-9):\t");

scanf("%d",&p.priority);

printf("\nBurst Time: %d\t",p.burst\_time);

scanf("%d",&p.burst\_time);

p.remaining\_time = p.burst\_time;

if(p.priority>0 && p.priority<=3){

q1[q1\_n++] = p;

}else if(p.priority>3 && p.priority<=6){

q2[q2\_n++] = p;

}else{

q3[q3\_n++] = p;

}

}

}

void printQueue(struct process \*q,int size){

calcWaitingTime(q,size);

calcTurnAroundTime(q,size);

printf("\nPId\t\tPriority\t\tBurst Time\t\tWaiting Time\t\tTurnAround Time\t\tArrival");

printf("\n================================================================================\n");

for(int i=0;i<size;i++){

printQueueI(q[i]);

}

printf("\n\n");

}

void printQueueI(struct process p){

printf("\n%d\t\t%d\t\t\t%d\t\t\t%d\t\t\t%d\t\t%d",p.pid,p.priority,p.burst\_time,p.waiting\_time,p.turnaround\_time,p.arrival\_time);

}

void calcWaitingTime(struct process \*q,int size){

q[0].waiting\_time = 0;

for(int i=1;i<size;i++){

q[i].waiting\_time = q[i-1].waiting\_time + q[i-1].burst\_time;

}

}

void calcTurnAroundTime(struct process \*q,int size){

q[0].waiting\_time = 0;

for(int i=0;i<size;i++){

q[i].turnaround\_time = q[i].waiting\_time + q[i].burst\_time;

}

}

void RoundRobinAlgo(struct process \*q,int size){

int time=0,i=0,remain=size,flag=0,wait\_time=0,tat\_time=0,total\_times=0;

for(time=0,i=0;remain!=0;){

struct process p = q[i];

if(p.remaining\_time<=time\_quantum && p.remaining\_time>0){

time += p.remaining\_time;

p.remaining\_time = 0;

flag = 1;

}else if(p.remaining\_time>time\_quantum){

p.remaining\_time -= time\_quantum;

time += time\_quantum;

}

if(p.remaining\_time==0 && flag==1){

remain--;

printf("\n%d\t\t%d\t\t\t%d\t\t\t%d\t\t\t%d",p.pid,p.priority,p.burst\_time,p.waiting\_time,p.turnaround\_time);

wait\_time += time -p.arrival\_time - p.burst\_time;

tat\_time += time -p.arrival\_time;

flag = 0;

}

if(i==remain-1){

i=0;

}else if(q[i+1].arrival\_time<time){

i++;

}else{

i=0;

}

q[i] = p;

}

printf("\nAverage Waiting Time= %f\n",wait\_time\*1.0/n);

printf("Avg Turnaround Time = %f\n",tat\_time\*1.0/n);

}

void RoundRobin(){

printf("\n\n===========================================================================");

printf("\n\t\tRound Robin\t");

printf("\n===========================================================================\n\n");

printf("\nPId\t\tPriority\t\tBurst Time\t\tWaiting Time\t\tTurnAround Time");

printf("\n================================================================================\n");

calcWaitingTime(q3,q3\_n);

calcTurnAroundTime(q3,q3\_n);

RoundRobinAlgo(q3,q3\_n);

}

void PrioSortingAlgorithm(struct process \*q,int size){

for(int i=0;i<size;i++){

for(int j=0;j<size;j++){

if(q[j].priority>q[i].priority){

struct process t = q[i];

q[i] = q[j];

q[j] = t;

}

}

}

}

void PrioSorting(){

printf("\n\n===========================================================================");

printf("\n\t\tPriority Sorting\t");

printf("\n===========================================================================\n\n");

PrioSortingAlgorithm(q2,q2\_n);

printQueue(q2,q2\_n);

}

void FCFSAlgorithm(struct process \*q,int size){

for(int i=0;i<size;i++){

for(int j=0;j<size;j++){

if(q[j].arrival\_time>q[i].arrival\_time){

struct process t = q[i];

q[i] = q[j];

q[j] = t;

}

}

}

}

void FCFS(){

printf("\n\n===========================================================================");

printf("\n\t\tFirst Come First Serve\t");

printf("\n===========================================================================\n\n");

FCFSAlgorithm(q1,q1\_n);

printQueue(q1,q1\_n);

}

int main(){

getInput();

int i=1;

while(n>0){

switch(i){

case 3:

RoundRobin();

break;

case 2:

PrioSorting();

break;

case 1:

FCFS();

break;

}

i++;

sleep(10);

}

printf("\n\n");

}

**Boundy Conditions :**

you can not enter any negative value of time and processes.