#include<iostream>

using namespace std;

/\*Ques. 17. Design a scheduler following non-preemptive scheduling approach to schedule the processes that arrives

at different units and having burst time double the arrival time. Scheduler selects the process with largest burst time from

the queue for the execution. Process is not being preempted until it finishes its service time. Compute the

average waiting time and average turnaround time. What should be the average waiting time if processes are executed

according to Shortest Job First scheduling approach with the same attribute values.

//23/03/18 14:50

\*/

float avg\_turnAroundtime(float a[],int len){

float avg=0;

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

avg+=a[i];

}

avg/=len;

return avg;

}

float avg\_waitingtime(float a[],int len){

float avg=0;

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

avg+=a[i];

}

avg/=len;

return avg;

}

int main(){

//mumber of process is 5(five)

//taking 1 as a start time of cpu

//their id is corresponding to index

float arrival\_time[5]={1,2,3,4,5} ;

int len =sizeof(arrival\_time)/sizeof(arrival\_time[0]);

float burst\_time[5]={0};

//Now we know burst time is double of arrival time so we are calulating it

int j;

for(j=0;j<5;j++){

//as burst time is double of arrival time

burst\_time[j]=arrival\_time[j]\*2;

}

//Now we are using Longest job first and drawing gantt chart

//assuming cpu startig from 1

cout<<"LONGEST JOB FIRST SCHEDULING:\n";

float completion\_time[5]={31,30,27,22,15};

//DRAWING TABLE OF PROCESS

for(int i=-1;i<5;i++){

if(i==-1){

cout<<"PROCESS ID | ARRIVAL TIME | BURST TIME |COMPLETION TIME|TURNAROUND TIME| WAITING TIME "<<endl;

}

else{

cout<<i<<" |"<<arrival\_time[i]<<" |"<<burst\_time[i]<<" |";

cout<<completion\_time[i]<<" |"<<completion\_time[i]-arrival\_time[i]<<" |"<<completion\_time[i]-arrival\_time[i]-burst\_time[i]<<endl;

}

}

float waiting\_time[5]={0};

float turnaround\_time[5]={0};

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

turnaround\_time[i]=completion\_time[i]-arrival\_time[i];

waiting\_time[i]= completion\_time[i]-arrival\_time[i]-burst\_time[i];

}

cout<<"\n\n\n";

cout<<"--------FOR LONGEST JOB FIRST SCHEDULING:---------\n";

cout<<"AVERAGE TURNAROUND TIME:"<<avg\_turnAroundtime(turnaround\_time,len)<<endl;

cout<<"AVERAGE WATING TIME:"<<avg\_waitingtime(waiting\_time,len);

cout<<"\n\n\n";

//for shortest job first

cout<<"SHORTEST JOB FIRST SCHEDULING:\n";

float completion\_time1[5]={0};

float counter\_time1=1;

for(int i=0;i<=4;i++){

cout<<"Process Id:"<<i<<" is running From :"<<counter\_time1<<" To:"<<counter\_time1+burst\_time[i]<<endl;

counter\_time1+=burst\_time[i];

completion\_time1[i]=counter\_time1;

}

cout<<"\n\n\n";

//DRAWING TABLE OF PROCESS

for(int i=-1;i<5;i++){

if(i==-1){

cout<<"PROCESS ID | ARRIVAL TIME | BURST TIME |COMPLETION TIME | TURNAROUND TIME| WAITING TIME "<<endl;

}

else{

cout<<i<<" |"<<arrival\_time[i]<<" |"<<burst\_time[i]<<" |";

cout<<completion\_time1[i]<<" |"<<completion\_time1[i]-arrival\_time[i]<<" |"<<completion\_time1[i]-arrival\_time[i]-burst\_time[i]<<endl;

}

}

//for storing waiting time and turnaround time

//formula is TURNAROUND TIME= COMPLETION TIME - ARRIVAL TIME

//formula is WAITING TIME= TURNAROUNDTIME - BURSTTIME

float waiting\_time1[5]={0};

float turnaround\_time1[5]={0};

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

turnaround\_time1[i]=completion\_time1[i]-arrival\_time[i];

waiting\_time1[i]= completion\_time1[i]-arrival\_time[i]-burst\_time[i];

}

cout<<"\n\n\n";

cout<<"--------FOR SHORTEST JOB FIRST SCHEDULING:---------\n";

cout<<"AVERAGE TURNAROUND TIME:"<<avg\_turnAroundtime(turnaround\_time1,len)<<endl;

cout<<"AVERAGE WATING TIME:"<<avg\_waitingtime(waiting\_time1,len);

cout<<"\n\n\n";

}