**Assignment 2b**

Code:

//ROUND ROBIN

#include <iostream>

using namespace std;

int main() {

int i, limit, total = 0, x, counter = 0, time\_quantum;

int wait\_time = 0, turnaround\_time = 0, arrival\_time[10], burst\_time[10], temp[10];

float average\_wait\_time, average\_turnaround\_time;

cout << "\nEnter Total Number of Processes:";

cin >> limit;

x = limit;

for (i = 0; i < limit; i++) {

cout << "\nEnter Details of Process[" << i + 1 << "]\n";

cout << "Arrival Time:";

cin >> arrival\_time[i];

cout << "Burst Time:";

cin >> burst\_time[i];

temp[i] = burst\_time[i];

}

cout << "\nEnter Time Quantum:";

cin >> time\_quantum;

cout << "\nProcess ID\t\tBurst Time\tTurnaround Time\t\tWaiting Time\n";

for (total = 0, i = 0; x != 0;) {

if (temp[i] <= time\_quantum && temp[i] > 0) {

total = total + temp[i];

temp[i] = 0;

counter = 1;

} else if (temp[i] > 0) {

temp[i] = temp[i] - time\_quantum;

total = total + time\_quantum;

}

if (temp[i] == 0 && counter == 1) {

x--;

cout << "\nProcess[" << i + 1 << "]\t\t" << burst\_time[i] << "\t\t" << total - arrival\_time[i]

<< "\t\t\t" << total - arrival\_time[i] - burst\_time[i];

wait\_time = wait\_time + total - arrival\_time[i] - burst\_time[i];

turnaround\_time = turnaround\_time + total - arrival\_time[i];

counter = 0;

}

if (i == limit - 1) {

i = 0;

} else if (arrival\_time[i + 1] <= total) {

i++;

} else {

i = 0;

}

}

average\_wait\_time = (float)wait\_time / limit;

average\_turnaround\_time = (float)turnaround\_time / limit;

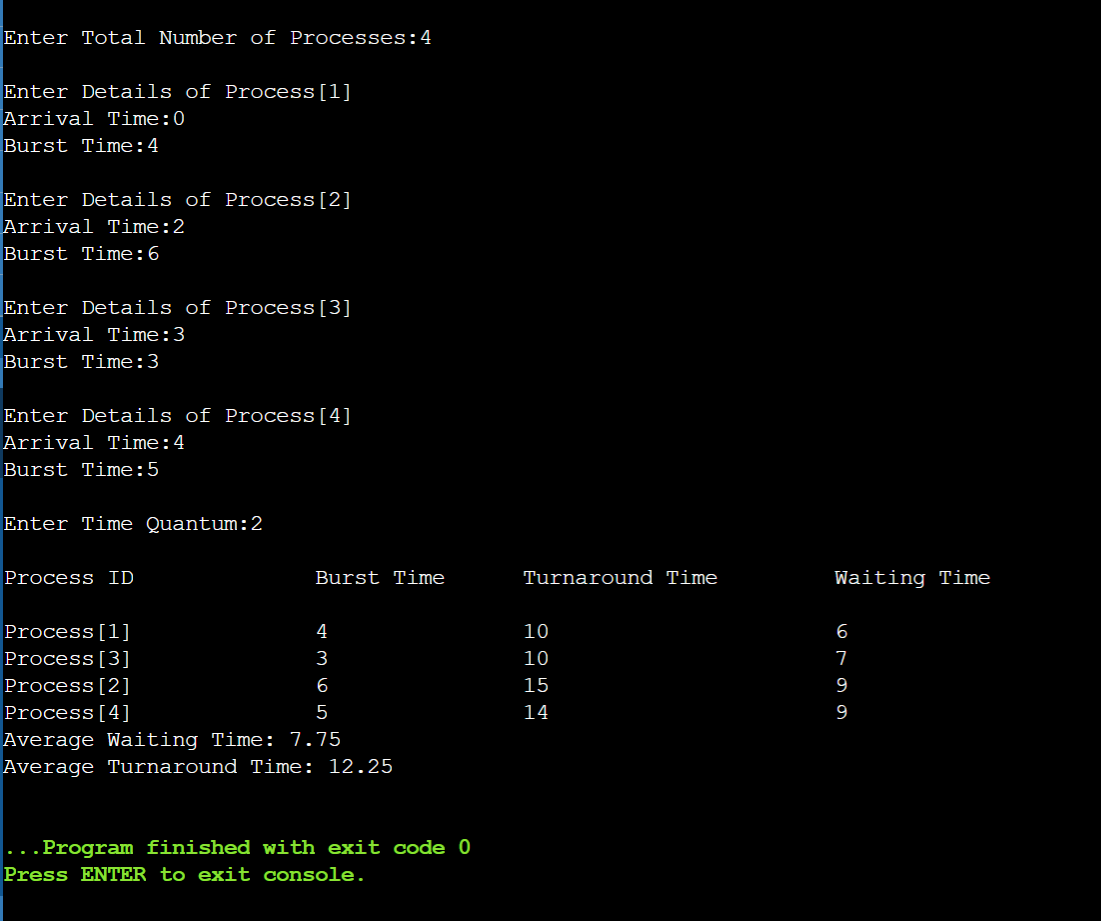
cout << "\nAverage Waiting Time: " << average\_wait\_time << endl;

cout << "Average Turnaround Time: " << average\_turnaround\_time << endl;

return 0;

}

Output:



//PRIORITY

CODE:

#include<iostream>

#include<algorithm>

using namespace std;

struct node{

char pname;

int btime;

int atime;

int priority;

int restime=0;

int ctime=0;

int wtime=0;

}a[1000],b[1000],c[1000];

void insert(int n){

int i;

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

cin>>a[i].pname;

cin>>a[i].priority;

cin>>a[i].atime;

cin>>a[i].btime;

a[i].wtime=-a[i].atime+1;

}

}

bool btimeSort(node a,node b){

return a.btime < b.btime;

}

bool atimeSort(node a,node b){

return a.atime < b.atime;

}

bool prioritySort(node a,node b){

return a.priority < b.priority;

}

int k=0,f=0,r=0;

void disp(int nop,int qt){

int n=nop,q;

sort(a,a+n,atimeSort);

int ttime=0,i;

int j,tArray[n];

int alltime=0;

bool moveLast=false;

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

alltime+=a[i].btime;

}

alltime+=a[0].atime;

for(i=0;ttime<=alltime;){

j=i;

while(a[j].atime<=ttime&&j!=n){

b[r]=a[j];

j++;

r++;

}

if(r==f){

c[k].pname='i';

c[k].btime=a[j].atime-ttime;

c[k].atime=ttime;

ttime+=c[k].btime;

k++;

continue;

}

i=j;

if(moveLast==true){

sort(b+f,b+r,prioritySort);

// b[r]=b[f];

// f++;

// r++;

}

j=f;

if(b[j].btime>qt){

c[k]=b[j];

c[k].btime=qt;

k++;

b[j].btime=b[j].btime-qt;

ttime+=qt;

moveLast=true;

for(q=0;q<n;q++){

if(b[j].pname!=a[q].pname){

a[q].wtime+=qt;

}

}

}

else{

c[k]=b[j];

k++;

f++;

ttime+=b[j].btime;

moveLast=false;

for(q=0;q<n;q++){

if(b[j].pname!=a[q].pname){

a[q].wtime+=b[j].btime;

}

}

}

if(f==r&&i>=n)

break;

}

tArray[i]=ttime;

ttime+=a[i].btime;

for(i=0;i<k-1;i++){

if(c[i].pname==c[i+1].pname){

c[i].btime+=c[i+1].btime;

for(j=i+1;j<k-1;j++)

c[j]=c[j+1];

k--;

i--;

}

}

int rtime=0;

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

rtime=0;

for(i=0;i<k;i++){

if(c[i].pname==a[j].pname){

a[j].restime=rtime;

break;

}

rtime+=c[i].btime;

}

}

float averageWaitingTime=0;

float averageResponseTime=0;

float averageTAT=0;

cout<<"\nGantt Chart\n";

rtime=0;

for (i=0; i<k; i++){

if(i!=k)

cout<<"| "<<'P'<< c[i].pname << " ";

rtime+=c[i].btime;

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

if(a[j].pname==c[i].pname)

a[j].ctime=rtime;

}

}

cout<<"\n";

rtime=0;

for (i=0; i<k+1; i++){

cout << rtime << "\t";

tArray[i]=rtime;

rtime+=c[i].btime;

}

cout<<"\n";

cout<<"\n";

cout<<"P.Name Priority AT\tBT\tCT\tTAT\tWT\n";

for (i=0; i<nop&&a[i].pname!='i'; i++){

if(a[i].pname=='\0')

break;

cout <<'P'<< a[i].pname << "\t";

cout << a[i].priority << "\t";

cout << a[i].atime << "\t";

cout << a[i].btime << "\t";

cout << a[i].ctime << "\t";

cout << a[i].wtime+a[i].ctime-rtime+a[i].btime << "\t";

averageTAT+=a[i].wtime+a[i].ctime-rtime+a[i].btime;

cout << a[i].wtime+a[i].ctime-rtime << "\t";

averageWaitingTime+=a[i].wtime+a[i].ctime-rtime;

cout <<"\n";

}

cout<<"Average Waiting time: "<<(float)averageWaitingTime/(float)n<<endl;

cout<<"Average TA time: "<<(float)averageTAT/(float)n<<endl;

}

int main(){

int nop,choice,i,qt;

cout<<"Enter number of processes\n";

cin>>nop;

cout<<"Enter process, priority, AT, BT\n";

insert(nop);

disp(nop,1);

return 0;

}

OUTPUT:

