**EXPERIMENT 5**

CPU SCHEDULING

**NITHIN JOSE(R4)**

**ROLL No: 46**

**FCFS**

**Code:-**

**FCFS.c**

# include <stdio.h>

# include <stdlib.h>

# include "process.h"

int main() {

int n;

process \*\*collection;

printf("Enter the number of process: ");

scanf("%d", &n);

collection = (process \*\*) malloc(n \* sizeof(process \*));

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

collection[i] = (process \*) malloc(sizeof(process));

printf("Process %d \n", i + 1);

collection[i]->pid = i + 1;

printf("Enter the Arrival Time: ");

scanf("%d", &((\*(collection + i))->at));

printf("Enter the Burst Time: ");

scanf("%d", &((\*(collection + i))->bt));

collection[i]->ct = 0;

collection[i]->tat = 0;

collection[i]->wt = 0;

}

sort(collection, 0, n - 1);

int curr = 0;

if (collection[0]->at != 0) {

curr = collection[0]->at;

}

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

if (collection[i]->at > curr) {

curr = collection[i]->at;

}

curr += collection[i]->bt;

collection[i]->ct = curr;

collection[i]->tat = collection[i]->ct - collection[i]->at;

collection[i]->wt = collection[i]->tat - collection[i]->bt;

}

printf("AT \t BT \t CT \t TAT \t WT\n");

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

printf("Process %d \n", collection[i]->pid);

printf("%d \t %d \t %d \t %d \t %d\n", collection[i]->at, collection[i]->bt, collection[i]->ct,

collection[i]->tat, collection[i]->wt);

}

int wt = 0;

int tat = 0;

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

wt += collection[i]->wt;

tat += collection[i]->tat;

}

float wt\_avg = (float) wt / n;

float tat\_avg = (float) tat / n;

printf("Average Waiting Time: %f\n", wt\_avg);

printf("Average Turn Around Time: %f", tat\_avg);

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

free(collection[i]);

}

free(collection);

return 0;

}

Process.h

# include <stdio.h>

# include <stdlib.h>

#include<string.h>

typedef struct {

int pid;

int bt;

int at;

int ct;

int tat;

int wt;

} process;

void sort(process \*\*collection,int start,int end){

process\* temp;

for(int i=start;i<end-1;i++)

{

for(int j=start;j<end-i-1;j++)

{

if(collection[j]->at>collection[j+1]->at)

{

temp = collection[j];

collection[j]=collection[j+1];

collection[j+1]=temp;

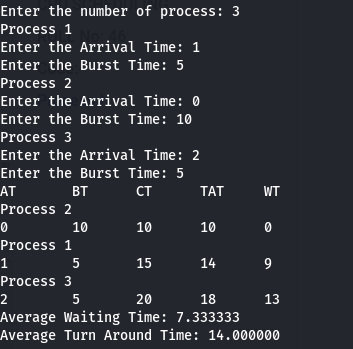
}

}

}

}

Output



**SJF**

**Code:-**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct process

{

int id,at,bt;

int wt,tat,ct;

}temp;

void calc(struct process \* startOfArray,int len)

{

float averagetat=0;

float averagewt=0;

struct process \*sjf,\*final;

sjf=(struct process\*)malloc(sizeof(struct process)\*len);

final=(struct process\*)malloc(sizeof(struct process)\*len);

//make a new sorted array based on burst time

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

{

sjf[i]=startOfArray[i];

}

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

{

for(int j=0;j<len-i-1;j++)

{

if(sjf[j].bt>sjf[j+1].bt)

{

temp=sjf[j+1];

sjf[j+1]=sjf[j];

sjf[j]=temp;

}

}

}

int timer=0;

int value=0;

int k=0;

int i=0;

while(i<len)

{

int flag=0;

int j=0;

if(startOfArray[i].at<=timer)

{

while(j<len)

{

if(sjf[j].ct==0&&startOfArray[i].bt>=sjf[j].bt&&sjf[j].at<=timer)

{

sjf[j].ct=1;

final[k].id=sjf[j].id;

final[k].at=sjf[j].at;

final[k].bt=sjf[j].bt;

if(k==0)

final[k].ct=sjf[j].bt+sjf[j].at;

else

final[k].ct=sjf[j].bt+final[k-1].ct;

final[k].tat=final[k].ct-sjf[j].at+value;

final[k].wt=final[k].tat-sjf[j].bt;

timer+=sjf[j].bt;

k++;

}

j++;

}

i++;

}

else

{

timer++;

value++;

}

}

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

printf("PROCESS AT BT TAT WT\n");

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

{

averagetat+=final[i].tat;

averagewt+=final[i].wt;

printf("%d %d %d %d %d\n",final[i].id,final[i].at,final[i].bt,final[i].tat,final[i].wt);

}

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

printf("Average Turn Around Time = %f\n",averagetat/len);

printf("Average Waiting Time = %f\n",averagewt/len);

}

int main()

{

int n;

printf("enter number of operations : ");

scanf("%d",&n);

struct process \* start;

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

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

{

start[i].ct=0;

printf("enter process name , arrival time and burst time : ");

scanf("%d %d %d",&start[i].id,&start[i].at,&start[i].bt);

}

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

printf("Process AT BT\n");

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

{

printf("Process%d %d %d\n",start[i].id,start[i].at,start[i].bt);

}

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

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

{

for(int j=0;j<n-i-1;j++)

{

if(start[j].at>start[j+1].at)

{

temp=start[j+1];

start[j+1]=start[j];

start[j]=temp;

}

}

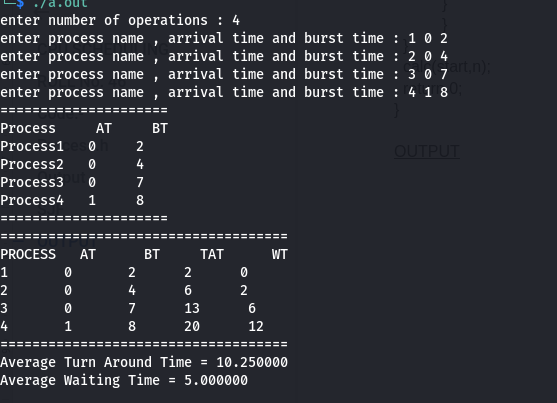
}

calc(start,n);

return 0;

}

OUTPUT



**PRIORITY**

**Code:-**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct process

{

int id,at,bt,pt;

int wt,tat,ct;

}temp;

void calc(struct process \* startOfArray,int len)

{

float averagetat=0;

float averagewt=0;

struct process \*sjf,\*final;

sjf=(struct process\*)malloc(sizeof(struct process)\*len);

final=(struct process\*)malloc(sizeof(struct process)\*len);

//make a new sorted array based on burst time

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

{

sjf[i]=startOfArray[i];

}

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

{

for(int j=0;j<len-i-1;j++)

{

if(sjf[j].pt>sjf[j+1].pt)

{

temp=sjf[j+1];

sjf[j+1]=sjf[j];

sjf[j]=temp;

}

}

}

printf("Sorted based on Burst Time\n");

printf("Processes AT BT PT\n");

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

{

printf("%d %d %d %d\n",sjf[i].id,sjf[i].at,sjf[i].bt,sjf[i].pt);

}

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

int timer=0;

int value=0;

int k=0;

int i=0;

while(i<len)

{

int flag=0;

int j=0;

if(startOfArray[i].at<=timer)

{

while(j<len)

{

if(sjf[j].ct==0&&startOfArray[i].pt>=sjf[j].pt&&sjf[j].at<=timer)

{

sjf[j].ct=1;

final[k].id=sjf[j].id;

final[k].at=sjf[j].at;

final[k].bt=sjf[j].bt;

if(k==0)

final[k].ct=sjf[j].bt+sjf[j].at;

else

final[k].ct=sjf[j].bt+final[k-1].ct;

final[k].tat=final[k].ct-sjf[j].at+value;

final[k].wt=final[k].tat-sjf[j].bt;

timer+=sjf[j].bt;

k++;

}

j++;

}

i++;

}

else

{

timer++;

value++;

}

}

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

printf("PROCESS AT BT TAT WT\n");

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

{

averagetat+=final[i].tat;

averagewt+=final[i].wt;

printf("%d %d %d %d %d\n",final[i].id,final[i].at,final[i].bt,final[i].tat,final[i].wt);

}

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

printf("Average Turn Around Time = %f\n",averagetat/len);

printf("Average Waiting Time = %f\n",averagewt/len);

}

int main()

{

int n;

printf("enter number of operations : ");

scanf("%d",&n);

struct process \* start;

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

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

{

start[i].ct=0;

printf("enter process name , arrival time,burst time and priority : ");

scanf("%d %d %d %d",&start[i].id,&start[i].at,&start[i].bt,&start[i].pt);

}

printf("Sorted based on Arrival Time\n");

printf("Processes AT BT PT\n");

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

{

printf("%d %d %d %d\n",start[i].id,start[i].at,start[i].bt,start[i].pt);

}

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

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

{

for(int j=0;j<n-i-1;j++)

{

if(start[j].at>start[j+1].at)

{

temp=start[j+1];

start[j+1]=start[j];

start[j]=temp;

}

}

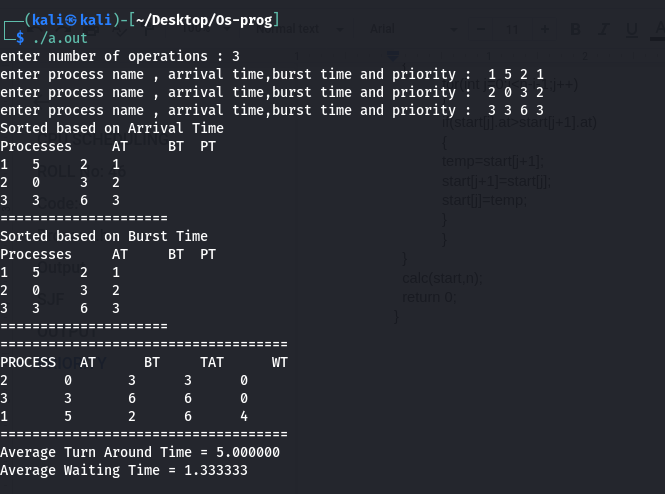
}

calc(start,n);

return 0;

}

OUTPUT



**ROUND ROBIN(PRE EMPTIVE)**

CODE:

#include <stdio.h>

#include <stdlib.h>

#include<string.h>

struct process

{

int num;

int at;

int bt;

int ct,tat,wt;

}temp;

void cal(struct process\* processArray,int len,int timeSlice)

{

int timer=0;

int i=0;

int value=0;

float averagetat=0;

float averagewt=0;

float tempArray[len];

int sum=0;

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

{

tempArray[i]=processArray[i].bt;

sum+=tempArray[i];

}

printf("sum is %d",sum);

int flag=0;

while(flag!=sum)

{

if(processArray[i].at<=timer)

{

if(tempArray[i]!=0)

{

if(tempArray[i]<=timeSlice)

{

processArray[i].ct=timer+tempArray[i];

timer=timer+tempArray[i];

flag+=tempArray[i];

tempArray[i]=0;

}

else

{

processArray[i].ct=timer+timeSlice;

tempArray[i]=tempArray[i]-timeSlice;

timer+=timeSlice;

flag+=timeSlice;

}

}

}

else

{

timer++;

value++;

}

i++;

if(i==len)

{

i=0;

}

}

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

printf("Process Arrival Time Burst Time T.A.T W.T\n");

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

{

processArray[i].tat=processArray[i].ct-processArray[i].at+value;

processArray[i].wt=processArray[i].tat-processArray[i].bt;

printf("Process %d %d %d %d %d\n",processArray[i].num,processArray[i].at,processArray[i].bt,processArray[i].tat,processArray[i].wt);

averagetat+=processArray[i].tat;

averagewt+=processArray[i].wt;

}

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

float floattat=averagetat/len;

float floatwt=averagewt/len;

printf("Average Turn Around Time = %f\n",floattat);

printf("Average Waiting Time = %f\n",floatwt);

}

int main()

{

int n,ts;

printf("enter number of processes : ");

scanf("%d",&n);

printf("enter time quantum : ");

scanf("%d",&ts);

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

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

{

start[i].ct=0;

start[i].wt=0;

printf("enter process number,arrival time,burst time of process[%d]: ",i);

scanf("%d %d %d",&start[i].num,&start[i].at,&start[i].bt);

}

for(int i=0;i<n-1;i++)

{

for(int j=0;j<n-i-1;j++)

{

if(start[j].at>start[j+1].at)

{

temp=start[j+1];

start[j+1]=start[j];

start[j]=temp;

}

}

}

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

printf("Process Arrival Time Burst Time\n");

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

{

printf("Process %d %d %d \n",start[i].num,start[i].at,start[i].bt);

}

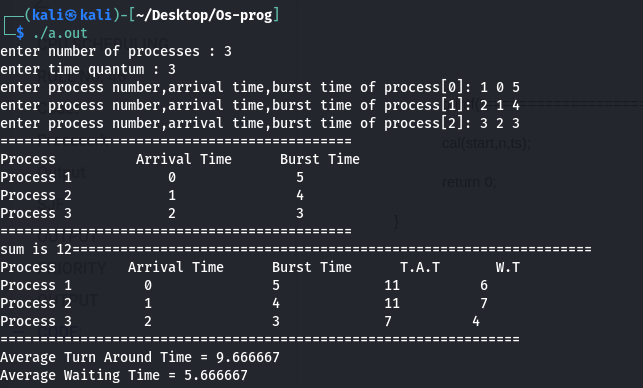
printf("============================================\n");

cal(start,n,ts);

return 0;

}

**OUTPUT**



-------------------------------------------------------------------------------------------------------------------------------