

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Programme | : | **BTech. CSE Core** | Semester | : | **Win 2021-22** |
| Course | : | **Operating Systems** | Code | : | **CSE2005** |
| Faculty | : | **Dr. Shyamala L** | Slot | : | **L25+L26** |
| Name | : | **Hariket Sukesh Kumar Sheth** | Register No. | : | **20BCE1975** |

**Lab 5:**Scheduling Algorithms in OS

**1**

|  |  |  |  |
| --- | --- | --- | --- |
| Date: 25-02-2022 | LAB 05 | Scheduling Algorithms |  |

**LAB 5**

Consider the ready queue of OS, the process are present and maintained with their arrival time and expected burst time for execution. Some processes have priority which is also given. Consider the required data to run different scheduling algorithms and analyse the result with respect to average waiting time and turnaround time.

* In priority, lowest number has highest priority
* For time quantum, t = 2 ms.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.no** | **Process ID** | **Arrival time** | **Expected Burst time** | **Priority** |
| 1 | P1 | 0 | 15 | 3 |
| 2 | P2 | 2 | 2 | 5 |
| 3 | P3 | 4 | 5 | 2 |
| 4 | P4 | 7 | 1 | 1 |
| 5 | P5 | 3 | 7 | 4 |

1. Write one single C /C++ program to simulate different scheduling algorithm in OS (functions)
   1. First Come First Serve

**#include<iostream>**

**#include<algorithm>**

**using namespace std;**

**struct process {**

**int id = 0;**

**int wait = 0;**

**int burst = 0;**

**int turnaround = 0;**

**int arrival = 0;**

**};**

**bool compareArrival(process *p1*, process *p2*) {**

**return (p1.arrival < p2.arrival);**

**}**

**int main() {**

**cout << "Enter number of processes for FCFS: ";**

**int n;**

**int ttlwait = 0, ttlturnaround = 0;**

**cin >> n;**

**process arr[n];**

**2**

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

**cout << "\nEnter PiD: ";**

**cin >> arr[i].id;**

**cout << "\nEnter Arrival time: ";**

**cin >> arr[i].arrival;**

**cout << "\nEnter Burst time: ";**

**cin >> arr[i].burst;**

**}**

**sort(arr, arr + n, compareArrival);**

**arr[0].wait = 0;**

**arr[0].turnaround = arr[0].burst;**

**for (int i = 1; i < n; i++) {**

**for (int j = 0; j < i; j++) {**

**arr[i].wait = arr[i].wait + arr[j].burst;**

**}**

**arr[i].turnaround = arr[i].wait + arr[i].burst;**

**ttlturnaround = ttlturnaround + arr[i].turnaround;**

**ttlwait = ttlwait + arr[i].wait;**

**}**

**int avgturnaround, avgwait;**

**avgturnaround = ttlturnaround / n;**

**avgwait = avgturnaround / n;**

**cout << "Avg waiting time is : " << avgwait << endl;**

**cout << "Avg turnaround is : " << avgturnaround << endl;**

**cout << "\n\nStats for each process: \n";**

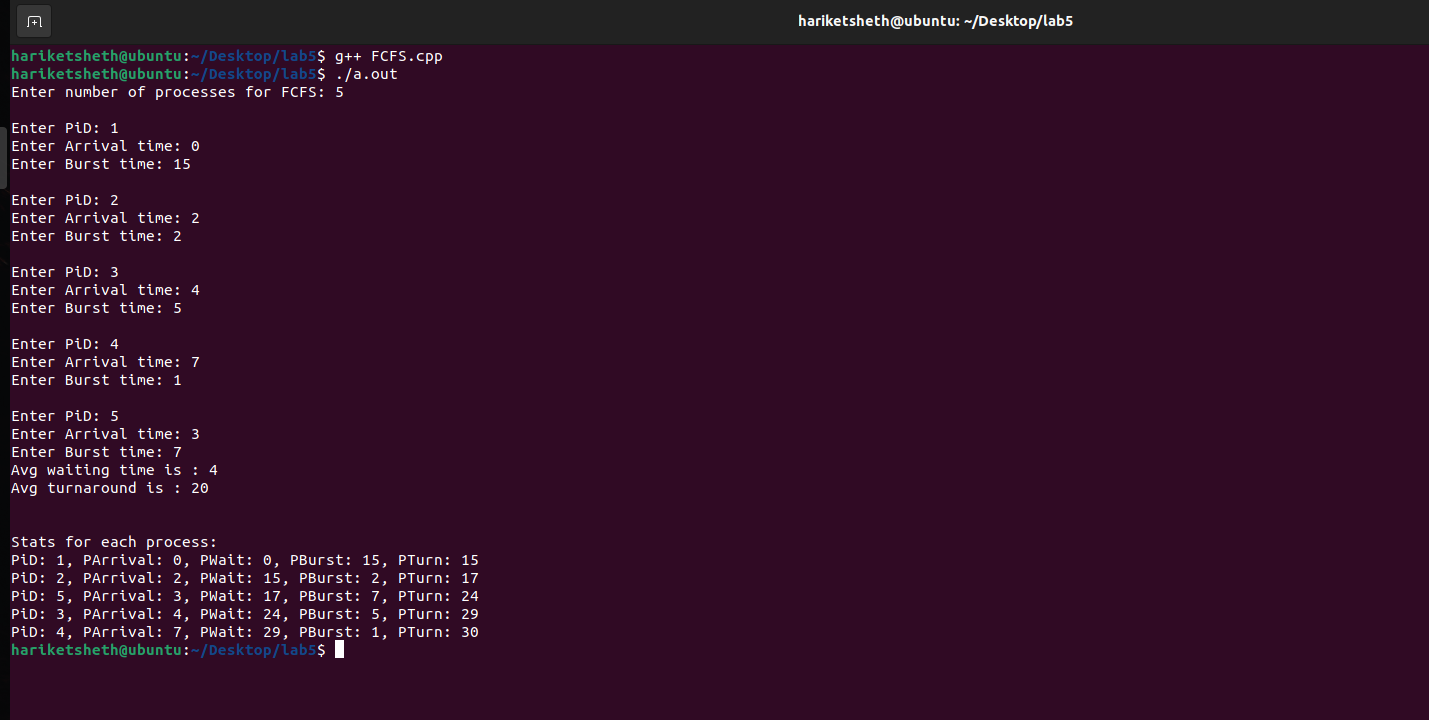
**for (int i = 0; i < n; i++) {**

**cout << "PiD: " << arr[i].id << ", PArrival: " << arr[i].arrival << ", PWait: " << arr[i].wait << ", PBurst: " << arr[i].burst << ", PTurn: " << arr[i].turnaround << endl;**

**}**

**return 0;**

**}**

**OUTPUT:**

**3**

**Average Waiting Time = 4  
Average Turnaround Time = 20**

* 1. Shortest Job First

**#include<stdio.h>**

**int main()**

**{**

**int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;**

**float avg\_wt,avg\_tat;**

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

**scanf("%d",&n);**

**printf("\nEnter Burst Time:\n");**

**for(i=0;i<n;i++)    {**

**printf("p%d:",i+1);**

**scanf("%d",&bt[i]);**

**p[i]=i+1;**

**}**

**for(i=0;i<n;i++)        {**

**pos=i;**

**for(j=i+1;j<n;j++)**

**{**

**if(bt[j]<bt[pos])**

**pos=j;**

**}**

**temp=bt[i];**

**bt[i]=bt[pos];**

**bt[pos]=temp;**

**temp=p[i];**

**p[i]=p[pos];**

**p[pos]=temp;**

**}**

**wt[0]=0;**

**for(i=1;i<n;i++)        {**

**wt[i]=0;**

**for(j=0;j<i;j++)**

**wt[i]+=bt[j];**

**total+=wt[i];**

**}**

**avg\_wt=(float)total/n;**

**total=0;**

**printf("\nProcessBurst\tTime\tWaiting Time\tTurnaround Time");**

**for(i=0;i<n;i++)        {**

**tat[i]=bt[i]+wt[i];**

**total+=tat[i];**

**printf("np%d\t\t%d\t\t%d\t\t\t%d\n",p[i],bt[i],wt[i],tat[i]);**

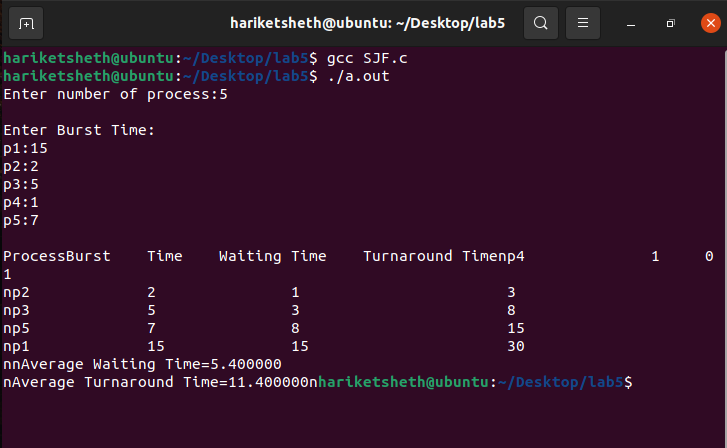
**}**

**avg\_tat=(float)total/n;**

**printf("nnAverage Waiting Time=%f\n",avg\_wt);**

**printf("nAverage Turnaround Time=%fn",avg\_tat);**

**}**

****

**OUTPUT:**

**4**

* 1. Shortest Remaining Time First

**#include <stdio.h>**

**int main()**

**{**

**int a[10],b[10],x[10],i,smallest,count=0,time,n;**

**double avg=0,tt=0,end;**

**printf("enter the number of Processes:\n");**

**scanf("%d",&n);**

**printf("enter arrival time\n");**

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

**scanf("%d",&a[i]);**

**printf("enter burst time\n");**

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

**scanf("%d",&b[i]);**

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

**x[i]=b[i];**

**b[9]=9999;**

**for(time=0;count!=n;time++)**

**{**

**smallest=9;**

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

**{**

**if(a[i]<=time && b[i]<b[smallest] && b[i]>0 )**

**smallest=i;**

**}**

**b[smallest]--;**

**if(b[smallest]==0)**

**{**

**count++;**

**end=time+1;**

**avg=avg+end-a[smallest]-x[smallest];**

**tt= tt+end-a[smallest];**

**}**

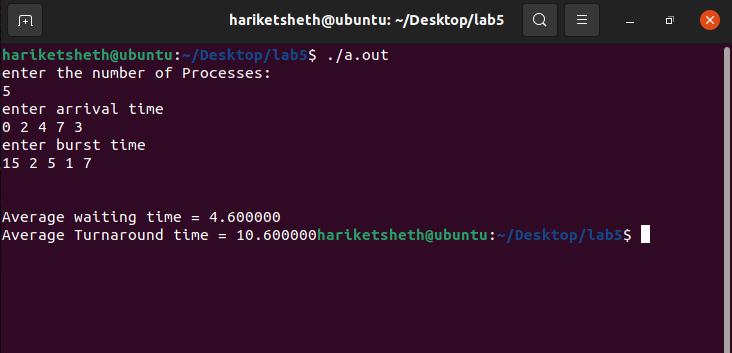
**}**

**printf("\n\nAverage waiting time = %lf\n",avg/n);**

**printf("Average Turnaround time = %lf",tt/n);**

**return 0;**

**}**

OUTPUT:

**5**

* 1. Priority
     1. Non-Pre-emptive

**#include<iostream>**

**#include<vector>**

**#include <algorithm>**

**using namespace std;**

**struct process {**

**int pid;**

**int arrtime;**

**int burstime;**

**int remaintime;**

**int priority; *//*higher is more urgent**

**int wtime = 999;**

**int turnaroundtime;**

**int exitime;**

**};**

**bool compareArr(process *p1*, process *p2*) {**

**return (p1.arrtime > p2.arrtime);**

**}**

**bool compareRem(process *p1*, process *p2*) {**

**return (p1.remaintime > p2.remaintime);**

**}**

**bool comparePri(process *p1*, process *p2*) {**

**return (p1.priority < p2.priority);**

**}**

**int main() {**

**int n;**

**int avgturnaround = 0, avgwait = 0;**

**cout << "Enter no. of processes: ";**

**cin >> n;**

**int curtime = 0, ttlexec = 0;**

**vector < process > notinqueue, waitqueue, inqueue, completed;**

**for (int i = 1; i < n + 1; i++) {**

**process a;**

**a.pid = i;**

**cout << "Enter the arrival time for pid " << i << ": ";**

**cin >> a.arrtime;**

**cout << "Enter the burst time for pid " << i << ": ";**

**cin >> a.burstime;**

**a.remaintime = a.burstime;**

**cout << "Enter priorty (higher is more urgent): ";**

**cin >> a.priority;**

**ttlexec = ttlexec + a.burstime;**

**notinqueue.push\_back(a);**

**}**

**cout << endl << endl;**

**sort(notinqueue.begin(), notinqueue.end(), compareArr);**

**for (curtime; curtime <= ttlexec; curtime++) {**

**int k1 = 999;**

**if (!notinqueue.empty())**

**k1 = notinqueue.back().arrtime;**

**if (inqueue.empty()) {**

**6**

**if (k1 == curtime) {**

**process a = notinqueue.back();**

**notinqueue.pop\_back();**

**inqueue.push\_back(a);**

**inqueue.back().remaintime--;**

**cout << "'" << inqueue.back().pid << "' ";**

**inqueue.back().wtime = 0;**

**avgwait = avgwait + curtime;**

**if (inqueue.back().remaintime == 0) {**

**inqueue.back().exitime = curtime + 1;**

**inqueue.back().turnaroundtime = inqueue.back().exitime - inqueue.back().arrtime;**

**avgturnaround = avgturnaround + inqueue.back().turnaroundtime;**

**completed.push\_back(inqueue.back());**

**inqueue.pop\_back();**

**}**

**}**

**} else {**

**if (k1 == curtime) {**

**process a = notinqueue.back();**

**notinqueue.pop\_back();**

**waitqueue.push\_back(a);**

**}**

**if (!inqueue.empty()) {**

**inqueue.back().remaintime--;**

**cout << "'" << inqueue.back().pid << "' ";**

**}**

**if (inqueue.back().wtime != 999) {**

**inqueue.back().wtime = curtime;**

**avgwait = avgwait + curtime;**

**}**

**if (inqueue.back().remaintime == 0) {**

**inqueue.back().exitime = curtime + 1;**

**inqueue.back().turnaroundtime = inqueue.back().exitime - inqueue.back().arrtime;**

**avgturnaround = avgturnaround + inqueue.back().turnaroundtime;**

**process a = inqueue.back();**

**completed.push\_back(a);**

**inqueue.pop\_back();**

**sort(waitqueue.begin(), waitqueue.end(), comparePri); *//*for non-primitive**

**inqueue.push\_back(waitqueue.back());**

**waitqueue.pop\_back();**

**}**

**}**

**}**

**avgturnaround = avgturnaround / n;**

**avgwait = avgwait / n;**

**cout << "\navg turnaround is: " << avgturnaround;**

**cout << "\navg wait is: " << avgwait << endl;**

**cout << "final results: " << endl;**

**sort(completed.begin(), completed.end(), compareArr);**

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

**cout << "Pid\tArrival Time\tBurst Time\tTurnaround Time\t\tExit Time\tPriority\n";**

**cout << completed.back().pid << "\t\t" << completed.back().arrtime << "\t\t" <<**

**completed.back().burstime << "\t\t" << completed.back().turnaroundtime << "\t\t" <<**

**completed.back().exitime << "\t\t" << completed.back().priority << "\n";**

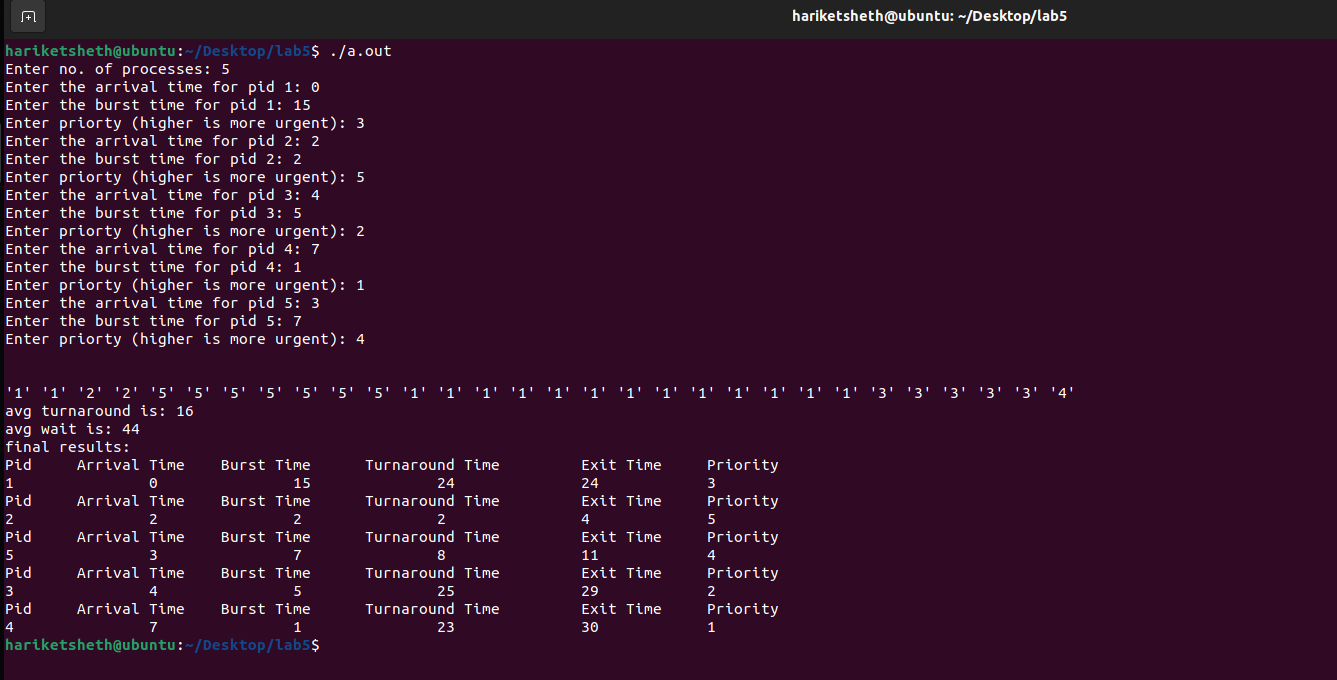
**completed.pop\_back();**

**}**

**return 0;**

**}**

**7**

OUTPUT:

* + 1. Pre-emptive

**#include<iostream>**

**#include<vector>**

**#include<algorithm>**

**using namespace std;**

**struct process {**

**int pid;**

**int arrtime;**

**int burstime;**

**int remaintime;**

**int priority; *//*higher is more urgent**

**int wtime = 999;**

**int turnaroundtime;**

**int exitime;**

**};**

**bool compareArr(process *p1*, process *p2*) {**

**return (p1.arrtime > p2.arrtime);**

**}**

**bool compareRem(process *p1*, process *p2*) {**

**return (p1.remaintime > p2.remaintime);**

**}**

**bool comparePri(process *p1*, process *p2*) {**

**return (p1.priority < p2.priority);**

**}**

**int main() {**

**8**

**int n;**

**int avgturnaround = 0, avgwait = 0;**

**cout << "Enter no. of processes: ";**

**cin >> n;**

**int curtime = 0, ttlexec = 0;**

**vector < process > notinqueue, inqueue, completed;**

**for (int i = 1; i < n + 1; i++) {**

**process a;**

**a.pid = i;**

**cout << "Enter the arrival time for pid " << i << ": ";**

**cin >> a.arrtime;**

**cout << "Enter the burst time for pid " << i << ": ";**

**cin >> a.burstime;**

**a.remaintime = a.burstime;**

**cout << "Enter priorty (higher is more urgent): ";**

**cin >> a.priority;**

**ttlexec = ttlexec + a.burstime;**

**notinqueue.push\_back(a);**

**}**

**cout << endl << endl;**

**sort(notinqueue.begin(), notinqueue.end(), compareArr);**

**for (curtime; curtime <= ttlexec; curtime++) {**

**int k1 = 999;**

**if (!notinqueue.empty())**

**k1 = notinqueue.back().arrtime;**

**if (inqueue.empty()) {**

**if (k1 == curtime) {**

**process a = notinqueue.back();**

**notinqueue.pop\_back();**

**inqueue.push\_back(a);**

**inqueue.back().remaintime--;**

**cout << "'" << inqueue.back().pid << "' ";**

**inqueue.back().wtime = 0;**

**avgwait = avgwait + curtime;**

**if (inqueue.back().remaintime == 0) {**

**inqueue.back().exitime = curtime + 1;**

**inqueue.back().turnaroundtime = inqueue.back().exitime - inqueue.back().arrtime;**

**avgturnaround = avgturnaround + inqueue.back().turnaroundtime;**

**process a = inqueue.back();**

**completed.push\_back(a);**

**inqueue.pop\_back();**

**}**

**}**

**} else {**

**if (k1 == curtime) {**

**process a = notinqueue.back();**

**notinqueue.pop\_back();**

**inqueue.push\_back(a);**

**}**

**sort(inqueue.begin(), inqueue.end(), comparePri); *//*for primitive**

**if (!inqueue.empty()) {**

**inqueue.back().remaintime--;**

**cout << "'" << inqueue.back().pid << "' ";**

**}**

**if (inqueue.back().wtime != 999) {**

**inqueue.back().wtime = curtime;**

**avgwait = avgwait + curtime;**

**}**

**if (inqueue.back().remaintime == 0) {**

**inqueue.back().exitime = curtime + 1;**

**inqueue.back().turnaroundtime = inqueue.back().exitime - inqueue.back().arrtime;**

**9**

**avgturnaround = avgturnaround + inqueue.back().turnaroundtime;**

**process a = inqueue.back();**

**completed.push\_back(a);**

**inqueue.pop\_back();**

**sort(inqueue.begin(), inqueue.end(), comparePri); *//*for primitive**

**}**

**}**

**}**

**avgturnaround = avgturnaround / n;**

**avgwait = avgwait / n;**

**cout << "\navg turnaround is: " << avgturnaround;**

**cout << "\navg wait is: " << avgwait << endl;**

**cout << "final results: " << endl;**

**sort(completed.begin(), completed.end(), compareArr);**

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

**cout << "Pid\tArrival Time\tBurst Time\tTurnaround Time\t\tExit Time\tPriority\n";**

**cout << completed.back().pid << "\t\t" << completed.back().arrtime << "\t\t" <<**

**completed.back().burstime << "\t\t" << completed.back().turnaroundtime << "\t\t" <<**

**completed.back().exitime << "\t\t" << completed.back().priority << "\n";**

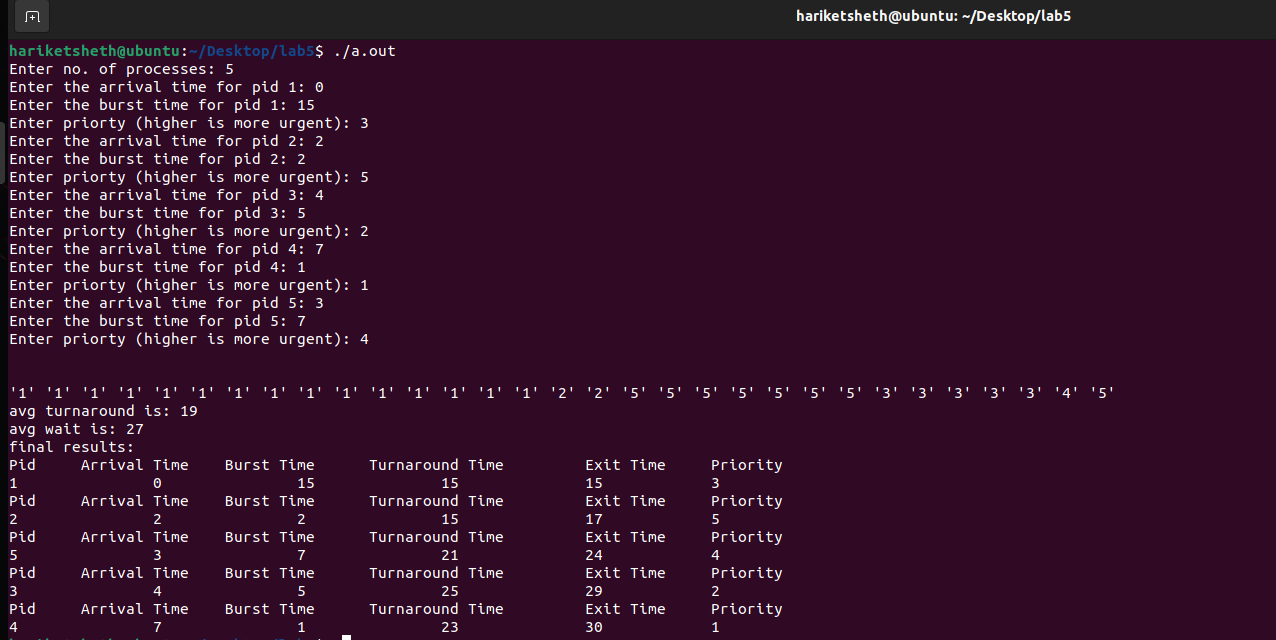
**completed.pop\_back();**

**}**

**return 0;**

**}**

OUTPUT:



* 1. Round Robin

**#include<stdio.h>**

**int main()**

**{**

**10**

**int count,j,n,time,remain,flag=0,time\_quantum;**

**int wait\_time=0,turnaround\_time=0,at[10],bt[10],rt[10];**

**printf("Enter Total Process:\t ");**

**scanf("%d",&n);**

**remain=n;**

**for(count=0;count<n;count++)**

**{**

**printf("Enter Arrival Time and Burst Time for Process Process Number %d :",count+1);**

**scanf("%d",&at[count]);**

**scanf("%d",&bt[count]);**

**rt[count]=bt[count];**

**}**

**printf("Enter Time Quantum:\t");**

**scanf("%d",&time\_quantum);**

**printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");**

**for(time=0,count=0;remain!=0;)**

**{**

**if(rt[count]<=time\_quantum && rt[count]>0)**

**{**

**time+=rt[count];**

**rt[count]=0;**

**flag=1;**

**}**

**else if(rt[count]>0)**

**{**

**rt[count]-=time\_quantum;**

**time+=time\_quantum;**

**}**

**if(rt[count]==0 && flag==1)**

**{**

**remain--;**

**printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-at[count],time-at[count]-bt[count]);**

**wait\_time+=time-at[count]-bt[count];**

**turnaround\_time+=time-at[count];**

**flag=0;**

**}**

**if(count==n-1)**

**count=0;**

**else if(at[count+1]<=time)**

**count++;**

**else**

**count=0;**

**}**

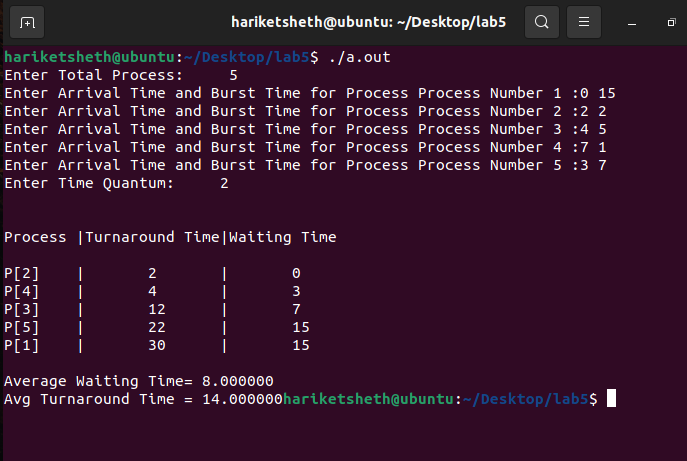
**printf("\nAverage Waiting Time= %f\n",wait\_time\*1.0/n);**

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

**return 0;**

**}**

**11**

OUTPUT: 

FCFS, Shortest Remaining Time First and Round Robin Scheduling algorithms have performed well in this scenario with **less average waiting time** and **more turnaround time**

**12**