Daffodil International University

Assignment on

Operating Systems

Course Title: Operating Systems

Course Code: CIS323

Submitted to:

Mr. Md. Aktaruzzaman Pramanik

Lecturer

Department of Computing & Information System (CIS)

Daffodil International University.

Submitted by:

Prosenjit Chowdhury

ID: 183-16-346

Department of Computing & Information System (CIS)

Daffodil International University.

Date of Submission: 07/04/2020

Task-1 (Theory)

Sub Task-1

CPU Scheduling algorithm (Round Robin)
Here, Time slice = 2

Prosenjit Chowdhury. ID: 183-16-346

CPU Scheduling Algorithm: (Round-Robin)

Process	Arival Time	Burst Time	Remaining Time	Waiting Time	Turn-around Time
P1	0	5	3,1,0	13	18
P2	1	3	1,0	6	9
P3	2	4	2,0	9	13
P4	3	7	5,3,1,0	21	28
P5	6	6	4,2,0	16	22
P6	8	10	8,6,4,2,0	17	27

Gantt chart:

P1	P2	P3	P1	P4	P2	P5	P3	P6

0 2 4 6 8 10 11 13 15 17

P1	P4	P5	P6	P4	P5	P6	P4	P6	P6

17 18 20 22 24 26 28 30 31 33 36

Queue: P1 P2 P3 P1 P4 P2 P5 P3 P6 P1 P4 P5 P6 P4 P5 P6 P4 P6 P6.

Average Waiting Time: $\left(\frac{13+6+9+21+16+17}{6}\right) = \frac{82}{6} = 13.67$

Average Turn-Around Time: $\left(\frac{18+9+13+28+22+27}{6}\right) = \frac{117}{6} = 19.5$ (Ans).

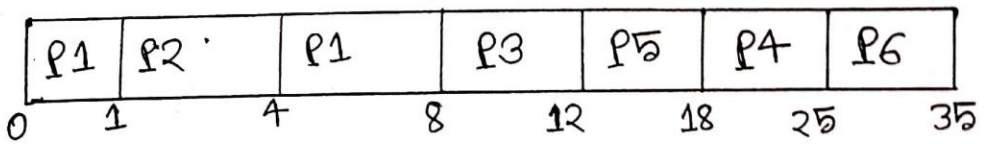
CPU Scheduling algorithm SJF (preemptive)

Prosenjit Chowdhury. ID: 183-16-346

CPU Scheduling Algorithm: (SJF-Preemptive)

Process	Arrival Time	Burst Time	Remaining Time	Waiting Time	Turn-around Time
P1	0	5	4,0	3	8
P2	1	3	0	0	3
P3	2	4	0	6	10
P4	3	7	0	15	22
P5	6	6	0	6	12
P6	8	10	0	17	27

Gantt chart:



∴ Average waiting Time: $\left(\frac{3+0+6+15+6+17}{6}\right) = \frac{47}{6} = 7.83$

∴ Average Turn Around Time: $\left(\frac{8+3+10+22+12+27}{6}\right)$

$= \frac{82}{6} = 13.67$

(Ans)

Sub Task – 2

Solaris Operating System

Introduction: Solaris is known for its scalability, especially on SPARC systems, and many innovative features emerging such as D-Trace, ZFS and Time Slider. Solaris supports SPARC and x86-64 workstations and servers from Oracle and other vendors. Solaris is registered as Compliant with the Single UNIX Specification. It is use Unix kernel.

History: Solaris is a Unix operating system developed by Sun Microsystems. The Sun Microsystems is now acquired by Oracle Corporation. It was founded in 1982 by Bill Joy. The Solaris was famous for its features like Scalability, Dtrace, ZFS and Time Slider. The Solaris operating system is written in C, C ++. Earlier Solaris was developed as proprietary software. Solaris Software is a perfect platform for network computing developed by Sun microsystems. Solaris OS enables the ability to support multiple-terabyte data warehouses and millions of users. From 2005 to March 2010 it was open source code but from March 2010 to present it is post-oracle closed source.

Hardware Support: The classic CPU for Sun systems is the SPARC chip. Many systems in deployment today, including SPARC 5, 10, and 20, use different versions of the SPARC chip, with processor speeds of around 40–60 MHz. Later systems, which use the UltraSPARC chipset, have processor speeds (as of this writing) of up to 900 MHz. Although this may not seem fast, the bus architectures of the Sun systems are much faster than their PC counterparts, making for much slower chip speeds. Many SPARC systems are still supported in Solaris 9, though it is advisable to check with Sun to determine whether older machines such as IPCs and IPXs will be supported in future releases.

Cross-Platform Interoperability: Solaris supports several different types of cross-platform interoperability. For example, Sun recently released a product called Lxrun, which allows Linux binaries to run under Solaris for Intel. This is very handy, as many database vendors, for example, have given away free versions of their database management products for Linux, but not for Solaris. Being able to exploit a free offer for a platform and make use of it on Solaris is a very handy cost saver indeed. Sun also includes a binary compatibility package in Solaris that allows Solaris 1.x applications to run without modification. However, success can depend on whether the application is statically or dynamically linked. It is not clear whether binary compatibility will continue to be supported in future releases of Solaris.

Solaris Operating System Versions: Solaris 1, Solaris 1.0.1, Solaris 1.1c, Solaris 1.1.1, Solaris 1.1.2, Solaris 2.0, Solaris 2.1, Solaris 2.2, Solaris 2.3, Solaris 2.4, Solaris 2.5, Solaris 2.5.1, Solaris 2.6, Solaris 7, Solaris 8, Solaris 9, Solaris 10, Solaris 11.
Out of these versions are the successful ones - Solaris 10, Solaris 11 express 2010.11, Solaris 11, Solaris 11.1, Solaris 11.2, Solaris 11.3.

Solaris Operating System Features - The Solaris OS is embedded in a distributed networking environment with a high-performance client server application. It provides features like transparent access to systems, servers, printers, remote databases and other resources. Transmission control protocol TCP / IP is the network protocol in the core of the Solaris networking environment. Some of the most important features include Solaris operating systems

1. Security - It provides the administrator with the ability to limit and enable applications to gain sufficient system resources to perform their functions only. This capability reduces the possibility of a hacker who can manipulate the written programming code. Even if hackers gain access to the server, they are unable to inject malicious code or damage data.

2. Networking - It releases more efficient routing and improved network availability, protocols to support telecommunications applications such as Voice over IP (VoIP).

3. Multiprocessing - It increases the productivity, database queries, provides remote file service, and accelerates computation intensive applications.

4. Multithreading - It is a software technique that divides program code into segments that can be executed in parallel on multiple processors for the overall fast execution of the main program. It is used in distributed client server applications.

5. NFS (Network File System) - It is used in distributed file system which facilitates transparent access to remote files and directories across the network.

6. Interoperability - It includes sub-feature like - Common desktop environment with Sun Java Desktop System desktops, easy portability with Java technology, Built-in binary and source code compatibility, free, high-quality porting tools.

7. Support and Services - An enhanced set of Support and Services duties are available during the entire life cycle of a Solaris operating system milestone versions.

8. Other features may include - Visualization, Platform choice, Observability or Dynamic Tracing, Data Management.

Advantages of Solaris:

- Solaris is free. Both Regular Solaris and Open Solaris
- Solaris 10 is secure and certified as secure.
- Containers are of interest to database people. Instead of needing to buy separate systems for their production, test, and development database environments they can put them in separate containers on one system.
- Open Solaris is a clear reason for Linux ahead, Linux is developed by lots of [smart] people on commodity hardware. Very powerful features in Solaris for databases and application servers

- IPS make installation / maintain / update very easy.

ZFS snapshot / rollback your config / data / tables, add / remove your storage space.

Ones Zones isolate your apps. clone your apps.

- SMF make your apps very reliable.
- DTrace observe everything, from the app to the kernel.

Disadvantages of Solaris:

Hardware support is not as good as many Linux or Windows operating systems. It is refined.

- You can run Solaris for free, but you can't get updates for free.
- It crashes, is insecure, buggy and expensive.
- It is less stable than Linux

Conclusions: A design goal for Solaris 10 was significantly improved performance for all applications. Users immediately benefit from an enhanced network stack, radically improved kernel, advanced trading technology, and special optimizations for memory allocation and chip multithreading. Solaris customers can improve performance on their existing applications by simply upgrading to Solaris 10.

Sub Task-3

I use Laptop model- hp ProBook 4440S from 2013. Though I live in Bangladesh So, I prefer Windows for my uses operating system. Because in Bangladesh most of the users used to with windows for their personal computer. Now I am using Windows 10 latest version. The second and main reason is windows is way more graphically well organized and user friendly than others. But I feel windows need some improvement for their next version. My recommendations are given below:

1) **Terminal is not user friendly-** Windows Command prompt is not user friendly as Unix base command prompt. In windows version windows add a new command prompt name power sell. But power sell is not too much well for write a command. MAC or Linux command prompt is a very useful tool for administration and daily task but windows command line can’t use as Linux or MAC command line. We need to go to run and enter cmd then command line will open and its written code is also difficult than others.

2) **Need Lite version-** Windows latest version 8,8.1 & 10 are graphically very high. Its feathers are required huge number of recourses to run well. It also takes much ram and CPU for its run time. It is not major issue for recent time computer but it creates problem when user use old model computer. Old model users face lot of

problem. It is not possible for every user to change their device after a fixed time. So, I think windows need to make lite version for them. Like as Lite Facebook.

3) **Security issue-** Windows malware and virus protection is good. Here I mention security issue for another reason. When you install an application in Linux or MAC it takes password then allow for installation. But windows always give permission for application if your computer is open. But in Linux you can't install any application in an open computer. It always takes password for install application. It is very dangerous. For example- Suppose you use PC in your office. You left your desk for any reason but your PC open. That time anyone can install any types of application in your PC.

4) **Extensibility issue-** Windows 10 is not enough customizable. Linux and MAC functions are well customizable. Users are can change their graphical interface easily. All the functions manage easily. But windows 10 is not follow user customizable format. It is not good decision for user. Example- Windows my computer GUI looks almost same in last 15 years.

Task-2(Lab)

Sub Task-1

Program's output screenshot (Using C Language):

```
                              --------------------------------------------------
                                   ### OPERATING SYSTEMS LAB ASSIGNMENT ###
                              --------------------------------------------------
                                     **** MADE BY - Prosenjit Chowdhury ****
                                             ID-> 183-16-346
                              --------------------------------------------------


Enter number of processes: 4
Enter arrival time for process p1: 0
Enter arrival time for process p2: 3
Enter arrival time for process p3: 2
Enter arrival time for process p4: 4
Enter burst time for process p1: 5
Enter burst time for process p2: 6
Enter burst time for process p3: 4
Enter burst time for process p4: 8

Choose any one:
1. Show Gantt Chart
2. Show comparison table
```

Figure:1. Program Interface & taking input from user.

```
                              --------------------------------------------------
                                   ### OPERATING SYSTEMS LAB ASSIGNMENT ###
                              --------------------------------------------------
                                     **** MADE BY - Prosenjit Chowdhury ****
                                             ID-> 183-16-346
                              --------------------------------------------------


Enter number of processes: 4
Enter arrival time for process p1: 0
Enter arrival time for process p2: 3
Enter arrival time for process p3: 2
Enter arrival time for process p4: 4
Enter burst time for process p1: 5
Enter burst time for process p2: 6
Enter burst time for process p3: 4
Enter burst time for process p4: 8

Choose any one:
1. Show Gantt Chart
2. Show comparison table
1

Choose any one:
1. FCFS
2. SJF preemptive
3. Round-Robin
```

Figure:2. Choose option 1 for show Gantt chart.

```
Enter arrival time for process p2: 3
Enter arrival time for process p3: 2
Enter arrival time for process p4: 4
Enter burst time for process p1: 5
Enter burst time for process p2: 6
Enter burst time for process p3: 4
Enter burst time for process p4: 8

Choose any one:
1. Show Gantt Chart
2. Show comparison table
1

Choose any one:
1. FCFS
2. SJF preemptive
3. Round-Robin
1



 Gantt chart of FCFS
 ---------- ------------ -------- ----------------
|    P0    |     P1     |   P2   |       P3       |
 ---------- ------------ -------- ----------------
0          5            11       15               23

Process returned 0 (0x0)   execution time : 202.782 s
Press any key to continue.
```

Figure:3. Choose option 1 FCFS for show FCFS Gantt chart.

```
                              ------------------------------------------------



Enter number of processes: 4
Enter arrival time for process p1: 0
Enter arrival time for process p2: 3
Enter arrival time for process p3: 2
Enter arrival time for process p4: 4
Enter burst time for process p1: 5
Enter burst time for process p2: 6
Enter burst time for process p3: 4
Enter burst time for process p4: 8

Choose any one:
1. Show Gantt Chart
2. Show comparison table
1

Choose any one:
1. FCFS
2. SJF preemptive
3. Round-Robin
2

Gantt chart:SJF preemptive
   p[1]    p[3]    p[2]    p[4]
0       5       9       15      23
Process returned 0 (0x0)   execution time : 36.102 s
Press any key to continue.
```

Figure:4. Choose option 2 SJF preemptive for show SJF preemptive Gantt chart.

```
Enter number of processes: 4
Enter arrival time for process p1: 0
Enter arrival time for process p2: 3
Enter arrival time for process p3: 2
Enter arrival time for process p4: 4
Enter burst time for process p1: 5
Enter burst time for process p2: 6
Enter burst time for process p3: 4
Enter burst time for process p4: 8

Choose any one:
1. Show Gantt Chart
2. Show comparison table
1

Choose any one:
1. FCFS
2. SJF preemptive
3. Round-Robin
3

Enter the time quantum:2

Gantt chart in Round-Robin
  p[1]    p[3]    p[1]    p[2]    p[4]    p[3]    p[1]    p[2]    p[4]    p[2]    p[4]    p[4]
0       2       4       6       8       10      12      13      15      17      19      21      23
Process returned 0 (0x0)   execution time : 33.550 s
Press any key to continue.
```

Figure:5. Choose option 3 Round Robin then Time quantum=2 for show Round Robin Gantt chart.

```
                                --------------------------------------------------


Enter number of processes: 4
Enter arrival time for process p1: 0
Enter arrival time for process p2: 3
Enter arrival time for process p3: 2
Enter arrival time for process p4: 4
Enter burst time for process p1: 5
Enter burst time for process p2: 6
Enter burst time for process p3: 4
Enter burst time for process p4: 8

Choose any one:
1. Show Gantt Chart
2. Show comparison table
2

Enter the time quantum for Round robin:2

Sample Comparison Table:

Algorithm       Avg. Waiting Time(ms)   Avg. Turnaround Time(ms)
FCFS                    5.500            11.250
SJF (Preemptive)        5.000            10.750
Round-Robin             8.750            14.500

Process returned 0 (0x0)   execution time : 46.709 s
Press any key to continue.
```

Figure:6. Choose option 2 Show comparison table then select Round Robin Time quantum=2 for show comparison table.

Raw Code: I use C language by Code Blocks platform for make the program.

```
/*Program  - LAB ASSIGNMENT Operating Systems
Author   - Md. Prosenjit Chowdhury
ID - 183-16-346
Dept. of Computing & Information System
Daffodil International University
Language - C Language
Date     - 07/03/2020*/
#include <stdio.h>

void sjfchar(int grtp[],int time);
struct process{
int name;
int at,bt,wt,tt,rt;
int completed;
float ntt;
}p[50];

int n;
int q[50];  //queue
int front=-1,rear=-1;
// enqueue function for round robin
void enqueue(int i){
if(rear==50)
printf("Queue is overflow");
rear++;
q[rear]=i;
if(front==-1)
front=0;
}

int dequeue(){
if(front==-1)
```

```
printf("Queue is empty");
int temp=q[front];
if(front==rear)
front=rear=-1;
else
front++;
return temp;
}
//check weather the value is in queue or not
int isInQueue(int i){
int k;
for(k=front;k<=rear;k++){
if(q[k]==i)
return 1;
}
return 0;
}
void sortByArrival(){
struct process temp;
int i,j;
//for(i=0;i<n-1;i++)
for(j=i+1;j<n;j++){
if(p[i].at>p[j].at){
temp=p[i];
p[i]=p[j];
p[j]=temp;
}
}
}

// SJF Gantt chart
void sjfchart(int grtp[],int time){
int old=-1,processtime[50],bttime=0;
```

```
int process[30];
int i,m=0;

for(i=0;i<time;i++){
if(grtp[i] == old){
processtime[m-1] = processtime[m-1]+1;
}else{
process[m] =grtp[i];
processtime[m] =1;
m++;
old = grtp[i];
}
}
printf("\nGantt chart:SJF preemptive\n");
for(i=0;i<m;i++){
printf("   p[%d] ",process[i]+1);
}
printf("\n0\t");
for(i=0;i<m;i++){
bttime = bttime+processtime[i];
printf("%d\t",bttime);
}
}


int main(){
//Program Design
printf("\t\t\t\t----------------------------------------------------\n");
printf("\t\t\t\t     ### OPERATING SYSTEMS LAB ASSIGNMENT ###       \n");
printf("\t\t\t\t----------------------------------------------------\n");
printf("\t\t\t\t       **** MADE BY - Prosenjit Chowdhury ****          \n");
printf("\t\t\t\t               ID-> 183-16-346                  \n");
printf("\t\t\t\t----------------------------------------------------\n\n\n");
```

```
int n,atime[10],btime[10],i,j,rtq;
printf("Enter number of processes: ");
scanf("%d",&n);
for(i=0;i<n;i++){
printf("Enter arrival time for process p%d: ",i+1);
scanf("%d",&atime[i]);
}

for(i=0;i<n;i++){
printf("Enter burst time for process p%d: ",i+1);
scanf("%d",&btime[i]);
}

printf("\nChoose any one: \n1. Show Gantt Chart\n2. Show comparison table\n");

int number;
scanf("%d",&number);
if(number==1){
//gantt chart portion
int chartNo;
printf("\nChoose any one:\n1. FCFS\n2. SJF preemptive\n3. Round-Robin\n");
scanf("%d",&chartNo);
if(chartNo == 1){
//FCFS portion
//FCFS code
int at[10]={0},bt[10]={0},tat[10]={0},wt[10]={0},ct[10]={0},ctn=0;
int sum=0,j,k;
float totalTAT=0,totalWT=0;

for(i=0;i<n;i++){
at[i] = atime[i];
bt[i] = btime[i];
}
```

```
for(j=0;j<n;j++){
ct[j+1] = ct[j] + bt[j]; //calculate completion time

tat[j]=ct[j]-at[j]; //calculate Turnaround time TAT = CT-AT
wt[j]=tat[j]-bt[j]; //calculate Waiting time WT=  TAT+BT

totalTAT = totalTAT+tat[j];
totalWT = totalWT+wt[j];
}

printf("\n\n");

//generating gantt chart
printf("\n Gantt chart of FCFS\n");
printf(" ");
for(i=0; i<n; i++) {
for(j=0; j<bt[i]; j++) printf("--");
printf(" ");
}
printf("\n|");
for(i=0; i<n; i++) {
for(j=0; j<bt[i] - 1; j++) printf(" ");
printf("P%d", i);
for(j=0; j<bt[i] - 1; j++) printf(" ");
printf("|");
}
printf("\n ");

// printing bottom bar
for(i=0; i<n; i++) {
for(j=0; j<bt[i]; j++) printf("--");
printf(" ");
```

```
}
printf("\n");

// printing the time line
printf("0");
for(i=0; i<n; i++) {
for(j=0; j<bt[i]; j++) {
printf("  ");
}
if(tat[i] > 9) printf("\b"); // backspace : remove 1 space

ctn = ctn + bt[i];
printf("%d",ctn);
}
printf("\n");

//end of FCFS gantt chart code
}
else if(chartNo == 2){
//SJF preemptive
//SJF code
int time,bt[10],at[10],sum_bt=0,smallest,rem,rt[10],grtp[50];
int sumt=0,sumw=0;

for(i=0;i<n;i++){
at[i] = atime[i];
bt[i] = btime[i];rt[i]=bt[i];
}

rt[9]=bt[9]=9999;
rem=n;

for(time=0;rem!=0;time++){
```

```
smallest=9;
for(i=0;i<n;i++){
if(at[i]<=time && rt[i]>0 && rt[i]<rt[smallest])
smallest=i;
}
rt[smallest]--;
grtp[time] = smallest;

if(rt[smallest]==0){
rem--;
sumt+=time+1-bt[smallest]-at[smallest];
sumw+=time+1-at[smallest];
}
}

sjfchart(grtp,time);
//end of SJF code
}
else if(chartNo == 3){
//Round-Robin portion

int j,time=0,sum_bt=0,tq,sqt[50],s=0,sql[50],sl=0;
float avgwt=0,avgtat=0;

for(i=0;i<n;i++){
p[i].name = i;
p[i].at = atime[i];
p[i].bt = btime[i];

p[i].rt=p[i].bt;
p[i].completed=0;
sum_bt+=p[i].bt;
}
```

```
printf("\nEnter the time quantum:");
scanf("%d",&tq);

sortByArrival();
enqueue(0);          // enqueue the first process

for(time=p[0].at;time<sum_bt;){       // run until the total burst time reached
i=dequeue();
if(p[i].rt<=tq){
// for processes having remaining time with less than
time+=p[i].rt;//From here need to take remaining time
sqt[s] = p[i].rt;s++;
p[i].rt=0;
p[i].completed=1;
sql[sl] = p[i].name;sl++;
p[i].wt=time-p[i].at-p[i].bt;
p[i].tt=time-p[i].at;
p[i].ntt=((float)p[i].tt/p[i].bt);
for(j=0;j<n;j++) {              //enqueue the processes which have come
if(p[j].at<=time && p[j].completed!=1&& isInQueue(j)!=1){
enqueue(j);
}
}
}else{              // more than time quantum
time+=tq;
p[i].rt-=tq;
sqt[s] = tq;s++;
sql[sl] = p[i].name;sl++;
for(j=0;j<n;j++){    //first enqueue the processes which have come while
if(p[j].at<=time && p[j].completed!=1&&i!=j&& isInQueue(j)!=1){
enqueue(j);
}
```

```
}
enqueue(i);   // then enqueue the uncompleted process
}
}

for(i=0;i<n;i++){
avgwt+=p[i].wt;
avgtat = avgtat+p[i].tt;
}

printf("\nGantt chart in Round-Robin\n");
for(i=0;i<sl;i++){
printf("  p[%d]\t",sql[i]+1);
}
printf("\n0\t");
int charttime = 0;
for(i=0;i<s;i++){ //sqt = sequence time,sl = sequence list,s =sequence,sql = sequence list
charttime=charttime+sqt[i];
printf("%d\t",charttime);
}
//end of Round robin code

}//if number not be within gantt chart
else{

printf("\nThe number is invalid. Try again\n");

}
}else if(number == 2){
//comparison portion
printf("\nEnter the time quantum for Round robin:");
scanf("%d",&rtq);
```

Prosenjit_Chowdhury_183-16-346

```
//FCFS code
int fbt[10]={0},fat[10]={0},ftat[10]={0},fwt[10]={0},fct[10]={0};
int fsum=0,fk;
float ftotalTAT=0,ftotalWT=0,fcfstat=0,fcfswt=0;

for( i=0;i<n;i++){
fat[i] = atime[i];
fbt[i] = btime[i];
}

for(j=0;j<n;j++){
//calculate completion time
fsum+=fbt[j];
fct[j]+=fsum;

ftat[j]=fct[j]-fat[j]; //calculate Turnaround time TAT = CT-AT
fwt[j]=ftat[j]-fbt[j]; //calculate Waiting time WT=  TAT+BT

ftotalTAT = ftotalTAT+ftat[j];
ftotalWT = ftotalWT+fwt[j];
}

fcfstat = ftotalTAT/n;
fcfswt = ftotalWT/n;

//end of FCFS code

//SJF code
int sjftime,sjfbt[10],sjfat[10],sjfsum_bt=0,sjfsmallest,sjfrem,sjfrt[10],grtp[50];
int sjfsumtat=0,sjfsumwat=0;

for(i=0;i<n;i++){
sjfat[i] = atime[i];
```

```
sjfbt[i] = btime[i];
sjfrt[i]=sjfbt[i];
}

sjfrt[9]=sjfbt[9]=9999;
sjfrem=n;

for(sjftime=0;sjfrem!=0;sjftime++){

sjfsmallest=9;
for(i=0;i<n;i++){

if(sjfat[i]<=sjftime && sjfrt[i]>0 && sjfrt[i]<sjfrt[sjfsmallest])
sjfsmallest=i;
}
sjfrt[sjfsmallest]--;
grtp[sjftime] = sjfsmallest;

if(sjfrt[sjfsmallest]==0){

sjfrem--;
sjfsumtat+=sjftime+1-sjfbt[sjfsmallest]-sjfat[sjfsmallest];
sjfsumwat+=sjftime+1-sjfat[sjfsmallest];
}
}//end of SJF code

//Round robin code
int rtime=0,rsum_bt=0,sqt[50],s=0,sql[50],sl=0;
float ravgwt=0,ravgtat=0;

for(i=0;i<n;i++){
p[i].name=i;
p[i].at = atime[i];
```

```
p[i].bt = btime[i];

p[i].rt=p[i].bt;
p[i].completed=0;
rsum_bt+=p[i].bt;
}



sortByArrival();
enqueue(0);          // enqueue the first process

for(rtime=p[0].at;rtime<rsum_bt;){ // loop until until reached the total bust time
i=dequeue();

if(p[i].rt<=rtq){
rtime+=p[i].rt;//From here remaining time need to take
sqt[s] = p[i].rt;s++;
p[i].rt=0;
p[i].completed=1;
sql[sl] = p[i].name;sl++;
p[i].wt=rtime-p[i].at-p[i].bt;
p[i].tt=rtime-p[i].at;
p[i].ntt=((float)p[i].tt/p[i].bt);
for(j=0;j<n;j++) {
//enqueue the processes which have come while scheduling
if(p[j].at<=rtime && p[j].completed!=1&& isInQueue(j)!=1){
enqueue(j);
}
}
}
else{            // if more than time quantum
rtime+=rtq;
p[i].rt-=rtq;
```

```
sqt[s] = rtq;s++;
sql[sl] = p[i].name;sl++;
for(j=0;j<n;j++){
if(p[j].at<=rtime && p[j].completed!=1&&i!=j&& isInQueue(j)!=1){
enqueue(j);
}
}
enqueue(i);   //enqueue the uncompleted process
}
}
for(i=0;i<n;i++){
ravgwt+=p[i].wt;
ravgtat = ravgtat+p[i].tt;
}

float roundravgwt,roundravgtat;
roundravgwt = ravgwt/n;
roundravgtat = ravgtat/n;

//end of Round robin code

//display option
printf("\nSample Comparison Table: \n\n");
printf("Algorithm\tAvg. Waiting Time(ms)\tAvg. Turnaround Time(ms)\n");
printf("FCFS\t\t\t%0.3f\t\t %0.3f\n",fcfswt,fcfstat);
printf("SJF (Preemptive)\t%0.3f\t\t %0.3f\n",sjfsumtat*1.0/n,sjfsumwat*1.0/n);
printf("Round-Robin\t\t%0.3f\t\t %0.3f\n",roundravgwt,roundravgtat);

//comparison code
}else{
printf("\nThe number is invalid. Try again\n");
}
}
```

Sub Task-2

Description of what challenges I faced while coding:

1. **Create Gantt Chart:** After taking when create Gantt chart it is hard. Because, three algorithms follow three rules. So, their Gantt chart is totally different from each other. When I code in code blocks, I follow different types of formula for different algorithm.

2. **Different algorithm in one program:** It is very hard that different types function write code in one program.

3. **Calculation challenge:** In program all the programs take same input but their outputs are different. Each algorithm follows different types of formula. When I make comparison table that time is very hard. Make one program with their waiting time & turn-around time is very challenging.

4. **Round Robin and its time slice:** For round robin algorithm calculation, I have taken an extra called time slice. Round robin calculation is depending on it.

5. **Take same input but output difference:** Here, three algorithms in one program. It is very hard to manage them. Because, their input same but their output, mechanism, Gantt chart, table are totally different from each other.

------: The End: -----