4. Simulate the following CPU scheduling algorithms

a) Round Robin b) SJF c) FCFS d) Priority

**a. Round Robin:**

**AIM:**

Implementation of round robin (RR) CPU scheduling algorithms.

**DESCRIPTION:**

Schedulers are the special software in system which handle process scheduling in various types. Their main task is to select the jobs to be submitted into the system and decide which process to run.

Round Robin is one of the CPU scheduling algorithms.

ADVANTAGES:

* It is simple and easy.
* Each process get equal chance to execute.
* Handles all processes without priority.

DISADVANTAGES:

* Depends on length of time slice.
* Same as FCFS, if time slice is indefinitely large.
* Small time slice will deteriorates due to frequent context switching.

**ALGORITHM:**

**Step 1:** Start

**Step 2:** Declare the variables

int st[10], bt[10], wt[10], ttat[10], n, i, tq, count <- 0, swt <- 0, stat <- 0,temp,sq <- 0

float awt <- 0.0, attat <- 0.0

**Step 3:** Read number of processes (n)

**Step 4:** Read the time slice (tq)

**Step 5:** Read the burst time

for I <- 0 repeat upto n

begin

display “ enter burst time of process %d”

Read b[i]

st[i] = bt[i]

end

**Step 6:** while(1)

begin

for i <- 0, count <- 0 repeat upto n

begin

temp <- tq

if(st[i] == 0)

begin

count ++

continue

end

if(st[i] > tq)

begin

st[i] <- st[i] – tq

end

else

begin

temp <- st[i]

st[i] <- 0

end

display (“ \n process no %d from time %d” ,i+1,sq)

sq <- sq + temp

display( “ to %d “,sq)

ttat[i] <- sq

end for

if(n == count)

break

end while

**Step 7:** for i <- 0 upto n

begin

w[i] <- ttat[i] – bt[i]

swt <- swt + wt[i]

stat <- stat + ttat[i]

end

awt <- (float)swt/n

attat <- (float)stat/n

**Step 8:** display “\n process no burst time \t waiting time \t turn aroud time \t”

for i <- 0 upto n

begin

display(i+1, bt[i], wt[i], ttat[i])

end

**Step 9:** display(awt & attat)

**Step 10:** Stop

**SOURCE CODE:**

#include<stdio.h>

void main()

{

int st[10],bt[10],wt[10],ttat[10],n,i,tq,count=0,swt=0,stat=0,temp,sq=0;

float awt=0.0,attat=0.0;

printf(“enter value of processes\n”);

scanf(“%d”,&n);

printf(“enter timeslice value\n”);

scanf(“%d”,&tq);

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

{

printf(“enter the burst time of the process %d\n”,i+1);

scanf(“%d”,&bt[i]);

st[i]=bt[i];

}

while(1)

{

for(i=0,count=0;i<n;i++)

{

temp=tq;

if(st[i]==0)

{

count++;

continue;

}

if(st[i]>tq)

{

st[i]=st[i]-tq;

}

else

{

temp=st[i];

st[i]=0;

}

printf(“\n process %d from time %d”,i+1,sq);

sq=sq+temp;

printf(“ to %d\n”,sq);

ttat[i]=sq;

}

if(n==count)

break;

}

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

{

wt[i]=ttat[i]-bt[i];

swt=swt+wt[i];

stat=stat+ttat[i];

}

awt=(float)swt/n;

attat=(float)stat/n;

printf(“\nprocess no\tburst time\twaiting time\tturnaroundtime”);

printf(“\n------------------------------------------------------------------\n”);

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

{

printf(“\n \t%d\t\t %d\t\t %d\t\t %d\n”,i+1,bt[i],wt[i],ttat[i]);

}

printf(“\n------------------------------------------------------------------\n”);

printf(“average waiting time of process is %f\n”,awt);

printf(“average turnaround time of process is %f\n”,attat);

}

**OUTPUT:**

enter value of processes

5

enter timeslice value

1

enter the burst time of the process 1

10

enter the burst time of the process 2

1

enter the burst time of the process 3

2

enter the burst time of the process 4

1

enter the burst time of the process 5

5

process 1 from time 0 to 1

process 2 from time 1 to 2

process 3 from time 2 to 3

process 4 from time 3 to 4

process 5 from time 4 to 5

process 1 from time 5 to 6

process 3 from time 6 to 7

process 5 from time 7 to 8

process 1 from time 8 to 9

process 5 from time 9 to 10

process 1 from time 10 to 11

process 5 from time 11 to 12

process 1 from time 12 to 13

process 5 from time 13 to 14

process 1 from time 14 to 15

process 1 from time 15 to 16

process 1 from time 16 to 17

process 1 from time 17 to 18

process 1 from time 18 to 19

process no burst time waiting time turnaroundtime

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

1 10 9 19

2 1 1 2

3 2 5 7

4 1 3 4

5 5 9 14

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

average waiting time of process is 5.400000

average turnaround time of process is 9.200000

**b. Shortest Job First:**

**AIM:**

Implementation of shortest job first (SJF) cpu scheduling algorithm.

**DESCRIPTION :**

A scheduler is a software which transfers the processes from job pool to main memory and from main memory to processor for execution based on cpu scheduling algorithms.

**SHORTEST JOB FIRST:**

* It is also called as shortest job next.
* It is the best approach to minimize waiting time.
* It is easy to implement in batch systems where required cpu time is known in advance.
* The job which has the shortest burst time will be get into the processor for execution.
* It is impossible to implement in interactive systems where required cpu time is not known.
* The processor should know in advance how much time process will take.

**ALGORITHM:**

**Step 1:** Start

**Step 2:** Declare necessary variables

temp,j,n,b[10],w[10],t[10],i, p[10] of int type

twt🡨 0.0,ttat🡨 0.0 of float type

**Step-3:** Read the number of process (n)

**Step-4:** Read the burst time for the process

for i🡨0 repeat to n

begin

Read b[i]

P[i]=i+1;

End

**Step-5:** for i🡨0 repeat to n

begin

for j🡨 i+1 repeat to n

begin

if(b[i]>b[j])

begin

temp🡨b[i]

b[i] 🡨b[j]

b[j] 🡨temp

temp🡨p[i]

p[i] 🡨p[j]

p[j] 🡨temp

end if

**Step-6:** w[0] 🡨 0

**Step-7:** for i🡨0 repeat to n

Begin

w[i+1] 🡨w[i]+b[i]

twt🡨twt+ w[i]

end

**Step-8:** t[0]=b[0], ttat=t[0]

**Step-9:** for i🡨1 repeat to n

begin

t[i]= w[i] + b[i]

ttat=t[i]

end

**Step-10:** for i🡨0 repeat to n

write(“waiting time of the process %d is %d”, p[i],w[i]);

write(“turn around time of the process%d is %d”,p[i],t[i]);

**Step-11:** write(“average waiting time of all process is %f\n”,twt/n) ;

write(“average turn around time of all process is %f”,ttat/n);

**Step-12:**stop

**SOURCE CODE:**

#include<stdio.h>

void main()

{

int temp,j, n, b[10],w[10],t[10],i, p[10];

float twt= 0.0,ttat= 0.0 ;

printf(“enter the number of processes\n”);

scanf(“%d”,&n);

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

{

printf(“enter the burst time of process%d:”,i+1);

scanf(“%d”,&b[i]);

p[i]=i+1;

}

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

{

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

{

if(b[i]>b[j])

{

temp=b[i];

b[i] =b[j];

b[j] =temp;

temp=p[i];

p[i] =p[j];

p[j] =temp;

}

}

}

w[0] = 0;

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

{

w[i+1] =w[i]+b[i];

twt=twt+ w[i];

}

t[0]=b[0];

ttat=t[0];

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

{

t[i]= w[i] + b[i];

ttat=t[i];

}

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

{

printf(“waiting time of the process %d is %d”, p[i],w[i]);

printf(“turn around time of the process%d is %d”,p[i],t[i]);

}

printf(“average waiting time of all process is %f\n”,twt/n) ;

printf(“average turn around time of all process is %f”,ttat/n);

}

**OUTPUT:**

Enter number of processes 5

Enter the burst time of process1: 15

Enter the burst time of process2: 22

Enter the burst time of process3: 27

Enter the burst time of process4: 12

Enter the burst time of process5: 8

Waiting time of the process5 is 0

Turn around time of the process5 is 8

Waiting time of the process4 is 8

Turn around time of the process4 is 20

Waiting time of the process1 is 20

Turn around time of the process1 is 35

Waiting time of the process2 is 35

Turn around time of the process2 is 57

Waiting time of the process3 is 57

Turn around time of the process3 is 84

Average Waiting time of all the process is 24.000000

Average Turn around time of all the process is 40.799999

**c. First Come First Served:**

**AIM:**

Implementation of first come first served (FCFS) CPU scheduling algorithms.

**DESCRIPTION:**

Schedulers are the special software in system which handle process scheduling in various types. Their main task is to select the jobs to be submitted into the system and decide which process to run.

First Come First Served is one of the CPU scheduling algorithms.

ADVANTAGES:

* It is simple.
* It is easy and useful.

DISADVANTAGES:

* This scheduling method is non-preemptive, that is, the process will run until it finishes.
* Throughput is not efficient.
* It is used in a small system only where I/O efficiency is not very important.

**ALGORITHM:**

**Step 1:** Start

**Step 2:** Declare necessary variables

n, b[10], w[10], t[10],i of int type; awt <- 0.0, att <- 0.0 of float type

**Step 3:** Read the number of processes

**Step 4:** Read the burst time for the process

for i <- 0 repeat to n

read(b[i])

**Step 5:** w[0] <- 0 , t[0] <- b[0]

**Step 6:** for i <- 1 repeat to n

begin

w[i] = w[i-1] + b[i-1]

t[i] = w[i] + b[i]

end

**Step 7:** for i <- 0 repeat to n

begin

write (“waiting time of process of %d is %d”, w[i])

write (“turnaround time of process of %d is %d”, t[i])

end

**Step 8:** for i <- 0 repeat to n

begin

awt <- awt + w[i]

att <- att + t[i]

end

**Step 9:** write (“average waiting time”,awt/n)

write (“average turnaround time”,att/n)

**Step 10:** stop.

**SOURCE CODE:**

#include<stdio.h>

void main()

{

int n, b[10], w[10], t[10], i;

float awt=0.0, att=0.0;

printf(“enter n value\n”);

scanf(“%d”, &n);

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

{

printf(“the burst time of process %d\n”,i+1);

scanf(“%d”, &b[i]);

}

w[0]=0;

t[0]=b[0];

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

{

w[i] = w[i-1] + b[i-1];

t[i] = w[i] + b[i];

}

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

{

printf(“waiting time of process %d is %d \n”, i+1, w[i]);

printf(“turn around time of process %d is %d \n”, i+1, t[i]);

}

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

{

awt = awt + w[i];

att = att + t[i];

}

printf(“average waiting time of process %f\n”, awt/n);

printf(“average turnaround time of process %f\n”, att/n);

}

**OUTPUT:**

enter n value

5

the burst time of process 1

10

the burst time of process 2

5

the burst time of process 3

15

the burst time of process 4

20

the burst time of process 5

7

waiting time of process 1 is 0

turn around time of process 1 is 10

waiting time of process 2 is 10

turn around time of process 2 is 15

waiting time of process 3 is 15

turn around time of process 3 is 30

waiting time of process 4 is 30

turn around time of process 4 is 50

waiting time of process 5 is 50

turn around time of process 5 is 57

average waiting time of process 21.000000

average turnaround time of process 32.400002

**d. Priority:**

**AIM:**

Implementation of priority cpu scheduling algorithm

**DESCRIPTION:**

A scheduler is a software that transfers the processes from job pool to main memory

And from main memory to processor for execution.

**PRIORITY:**

* Priority scheduling is one of the most common scheduling algorithms in batch systems.
* Each process is assigned a priority.
* Process with highest priority is to be executed first and so on.
* Process with same priority are executed on FCFS basis.
* Priority can be decided based on memory requirements, time requirements or any other resource requirements.

**ALGORITHM:**

**Step 1:** Start

**Step 2:** Declare necessary variables

temp,j, n, b[10],w[10],t[10],i, p[10],pr[10] of int type

twt=0.0,ttat=0.0 of float type

**Step 3:** Read the number of process (n)

**Step 4:** Read the burst time and priorities

for i🡨0 repeat to n

begin

write “enter burst time and priority”

Read b[i],pr[i]

P[i] 🡨i+1

End

**Step 5:** for i🡨0 repeat to n

begin

for j🡨 i+1 repeat to n

begin

if(pr[i]>pr[j])

begin

temp🡨pr[i]

pr[i] 🡨pr[j]

pr[j] 🡨temp

temp🡨pr[i]

pr[i] 🡨pr[j]

pr[j] 🡨temp

temp🡨p[i]

p[i] 🡨p[j]

p[j] 🡨temp

end if

end for

end for

**Step 6:** w[0] 🡨 0

t[0] 🡨b[0]

**Step 7:** for i🡨1 repeat to n

begin

w[i] 🡨w[i-1]+b[i-1]

t[i]🡨+w[i]+b[i]

end for

**Step 8:** for i🡨0 repeat to n

begin

twt🡨twt+ w[i]

ttat🡨ttat+t[i]

end for

**Step 9:** for i🡨0 repeat to n

Write “waiting time of the process is “w[i]

Write “turn around time of the process is %d” t[i]

end

**Step 10:** write”average waiting time of all process is” twt/n

write”average turn around time of all process is” ttat/n

**Step 11:**stop

**SOURCE CODE:**

#include<stdio.h>

void main()

{

int temp,j, n, b[10],w[10],t[10],i, p[10],pr[10];

float twt= 0.0,ttat= 0.0 ;

printf(“enter the number of processes\n”);

scanf(“%d”,&n);

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

{

printf(“enter the burst time of process%d:”,i+1);

scanf(“%d”,&b[i]);

printf(“enter the priority of process%d:”,i+1);

scanf(“%d”,&pr[i]);

p[i]=i+1;

}

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

{

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

{

if(pr[i]>pr[j])

{

temp=pr[i];

pr[i] =pr[j];

pr[j] =temp;

temp=b[i];

b[i] =b[j];

b[j] =temp;

temp=p[i];

p[i] =p[j];

p[j] =temp;

}

}

}

w[0] = 0;

t[0]=b[0];

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

{

w[i] =w[i-1]+b[i-1];

t[i]= w[i]+b[i];

}

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

{

twt= twt+w[i];

ttat=ttat+t[i];

}

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

{

printf(“waiting time of the process %d is %d”, p[i],w[i]);

printf(“turn around time of the process%d is %d”,p[i],t[i]);

}

printf(“average waiting time of all process is %f\n”,twt/n) ;

printf(“average turn around time of all process is %f”,ttat/n);

}

**OUTPUT:**

Enter number of processes 5

Enter the burst time of process1: 10

Enter the priority of process1:1

Enter the burst time of process2: 5

Enter the priority of process2:3

Enter the burst time of process3: 15

Enter the priority of process3:4

Enter the burst time of process4: 20

Enter the priority of process4:0

Enter the burst time of process5: 7

Enter the priority of process5:2

Waiting time of the process4 is 0

Turn around time of the process4 is 20

Waiting time of the process1 is 20

Turn around time of the process1 is 30

Waiting time of the process5 is 30

Turn around time of the process5 is 37

Waiting time of the process2 is 37

Turn around time of the process2 is 42

Waiting time of the process3 is 42

Turn around time of the process3 is 57

Average Waiting time of all the process is 25.799999

Average turn around time of all the process is 37.200001