



GATE INSTITUTE OF TECHNOLOGY
& MANAGEMENT SCIENCES

(Approved by AICTE, Affiliated to Sri Venkateswara University, TIRUPATI)

ICET CODE : GTMT



Master of Computer Applications

SEMESTER - _____

Hall Ticket No: 3124167

Name of the Student: Kondapalli Sai Teja

Subject: operating system

Year: 2 Semester: 1st Batch 2024-2026



GATE EDUCATIONAL INSTITUTIONS

DEPARTMENT OF COMPUTERS

YEAR 2 SEMESTER 1st

Reg No. 3124167

CERTIFICATE

Certified that this is the bonafide record of practical work done in the laboratory
by the candidate Kondapalli Sai Teja during the
academic year 2024-2026 subject operating system.

No. of Practical's Conducted 12

No. of Practical's Attended 12


LECTURER / HEAD
Department of Computer Applications
GATE Institute of Technology
And Management Sciences
TIRUPATI


LECTURER / HEAD
HEAD OF THE DEPARTMENT
GATE Institute of Technology
And Management Sciences
TIRUPATI

Submitted for the Practical Examination held on _____

Examiners

1.

2.

INDEX

S.No.	Date	Name of the Experiment	Page No.	Marks Awarded	Remarks
1		producer consumer using threads			
2		first come first serve scheduling algorithm			
3		shortest Job first scheduling algorithm			
4		Round Robin scheduling algorithm			
5		First fit			
6		Best fit			
7		Worst fit			
8		Bankers Algorithm			
9		First in first out page replacement algorithm			
10		optimal page replacement algorithm			

INDEX

S.No.	Date	Name of the Experiment	Page No.	Marks Awarded	Remarks
		1. To find the value of π using a circular disc	100	10	1
		2. To find the value of π using a circular disc	101	10	2
		3. To find the value of π using a circular disc	102	10	
		4. To find the value of π using a circular disc	103	10	3
		5. To find the value of π using a circular disc	104	10	4
		6. To find the value of π using a circular disc	105	10	5
		7. To find the value of π using a circular disc	106	10	6
		8. To find the value of π using a circular disc	107	10	7
		9. To find the value of π using a circular disc	108	10	8
		10. To find the value of π using a circular disc	109	10	9

INDEX

S.No.	Date	Name of the Experiment	Page No.	Marks Awarded	Remarks
11		Least Recently used page Replacement Algorithm			
12		First come first serve Disk scheduling Algorithm			
		Kishor Kumar			

INDEX

[illegible]

Date :	GATE Educational Institutions	Page No.:
	producer consumer using threads	Practical No. :

Program Definition:—

Design and develop a java program for implementing produces consumes concepts for using threads.

Descriptions:—

- * A thread is a single sequential flow of execution of task of process. so it is known as thread of execution (or) thread of control.
- * Thread is often referred to as a light weight process.
- * In java there are two types of threads
 - * kernel level thread
 - * user level thread

Algorithm:—

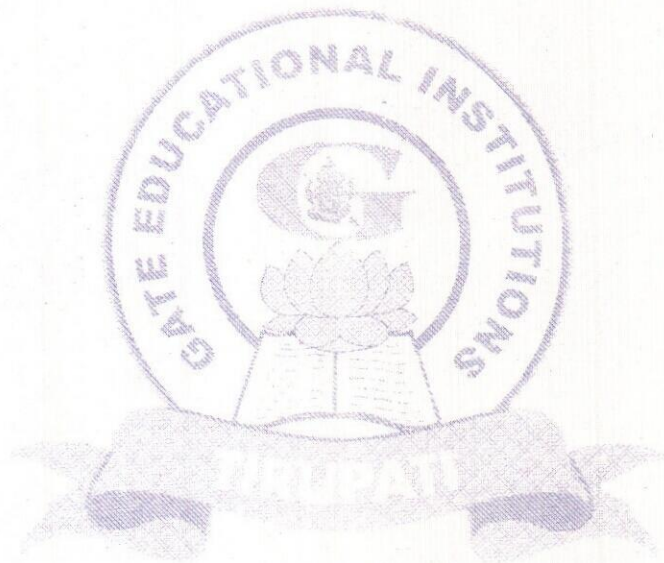
- step 1: start the program
- step 2: create class thread example
- step 3: create instance for thread
- step 4: declare an method with try and catch blocks.
- step 5: to override the run method from class pc
- step 6: creation exception

Date :	GATE Educational Institutions	Page No.:
		Practical No. :

Step 7: create linked list in class PC

Step 8: thread methods, start, join, sleep is used

Step 9: stop the program



Date :	GATE Educational Institutions	Page No. :
		Practical No. :

1. Design and Develop a Java Program to implement Reader Writers Problem

```

import java.util.concurrent.Semaphore;
class ReaderWritersProblem {
static Semaphore readLock = new Semaphore(1);
static Semaphore writeLock = new Semaphore(1);
static int readCount = 0;
static class Read implements Runnable {
public void run() {
try {
readLock.acquire();
readCount++;
if (readCount == 1) {
writeLock.acquire();
}
readLock.release();
System.out.println("Thread "+Thread.currentThread().getName() + " is READING");
Thread.sleep(1000);
System.out.println("Thread "+Thread.currentThread().getName() + " has FINISHED
READING");
readLock.acquire();
readCount--;
if(readCount == 0) {
writeLock.release();
}
readLock.release();
} catch (InterruptedException e) {
System.out.println(e.getMessage());
}
}
}
static class Write implements Runnable {
public void run() {
try {
writeLock.acquire();
System.out.println("Thread "+Thread.currentThread().getName() + " is WRITING");
Thread.sleep(2500);
System.out.println("Thread "+Thread.currentThread().getName() + " has finished
WRITING");
writeLock.release();
} catch (InterruptedException e) {
System.out.println(e.getMessage());
}
}
}
}

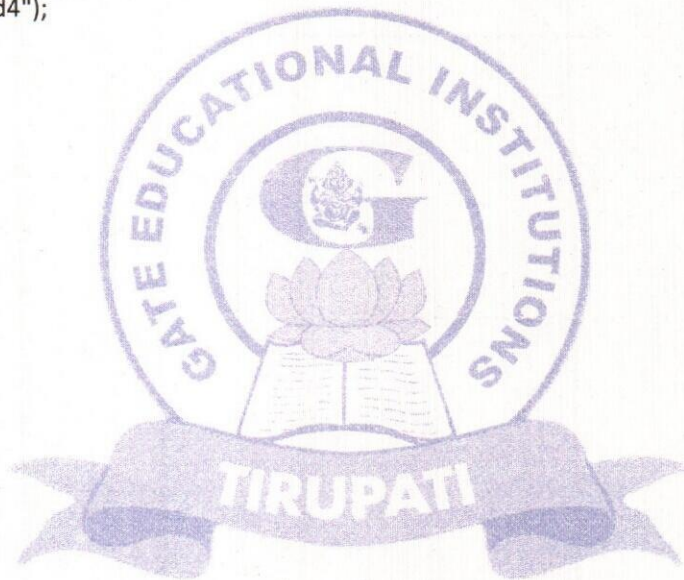
```


Date :	GATE Educational Institutions	Page No. :
		Practical No. :

```

    }
}
public static void main(String[] args) throws Exception {
    Read read = new Read();
    Write write = new Write();
    Thread t1 = new Thread(read);
    t1.setName("thread1");
    Thread t2 = new Thread(read);
    t2.setName("thread2");
    Thread t3 = new Thread(write);
    t3.setName("thread3");
    Thread t4 = new Thread(read);
    t4.setName("thread4");
    t1.start();
    t3.start();
    t2.start();
    t4.start();
}
}

```



Date :

GATE Educational Institutions

Page No.:

Practical No. :

output :-

producer produced - 0
producer produced - 1
consumer consumed - 0
consumer consumed - 1
producer produced - 2
producer produced - 2
consumer consumed - 2
consumer consumed - 3
producer produced - 4
producer produced - 5
consumer consumed - 4
consumer consumed - 5

conclusion :-

The above java program executed successfully

Date :	GATE Educational Institutions	Page No.:
	First come first serve scheduling algorithm	Practical No. :

Program definition :-

TO design and develop a java program to illustrate the concept of first serve algorithm for cpu scheduling.

Description :-

- * FCFS - first come first serve
- * FCFS is a non-preemption algorithm for cpu scheduling
- * it executes the process which is first in the ready queue
- * it also assign the process based on the arrival time.

Algorithm :-

step 1 : start the process

step 2 : accept the number of processors into the ready queue

step 3 : for each process in the ready queue assign the process arrival time and burst time.

step 4 : sort all the processors according to the arrival time, by declaring tempt

$ct[i] = ar[i] + bt[i]$, for the first process and
if $cat[i] > ct[i-1]$ the calalable
 $ct[i] = at[i] + bt[i]$

else

$ct[i] = ct[i-1] + bt[i]$

Date :	GATE Educational Institutions	Page No. :
		Practical No. :

2. Design and Develop a Java Program to implement FCFS CPU Scheduling Algorithm

```

import java.util.*;
public class FCFS {
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("enter no of process: ");
        int n = sc.nextInt();
        int pid[] = new int[n]; // process ids
        int ar[] = new int[n]; // arrival times
        int bt[] = new int[n]; // burst or execution times
        int ct[] = new int[n]; // completion times
        int ta[] = new int[n]; // turn around times
        int wt[] = new int[n]; // waiting times
        int temp;
        float avgwt=0,avgta=0;
        for(int i = 0; i < n; i++)
        {
            System.out.println("enter process " + (i+1) + " arrival time: ");
            ar[i] = sc.nextInt();
            System.out.println("enter process " + (i+1) + " brust time: ");
            bt[i] = sc.nextInt();
            pid[i] = i+1;
        }
        //sorting according to arrival times
        for(int i = 0 ; i < n; i++)
        {
            for(int j=0; j < n-(i+1) ; j++)
            {
                if( ar[j] > ar[j+1] )
                {
                    temp = ar[j];
                    ar[j] = ar[j+1];
                    ar[j+1] = temp;
                    temp = bt[j];
                    bt[j] = bt[j+1];
                    bt[j+1] = temp;
                    temp = pid[j];
                    pid[j] = pid[j+1];
                    pid[j+1] = temp;
                }
            }
        }
    }
}

```


Date :	GATE Educational Institutions	Page No. :
		Practical No. :

```

    }
    // finding completion times
    for(int i = 0 ; i < n; i++)
    {
        if( i == 0)
        {
            ct[i] = ar[i] + bt[i];
        }
        else
        {
            if( ar[i] > ct[i-1])
            {
                ct[i] = ar[i] + bt[i];
            }
            else
            {
                ct[i] = ct[i-1] + bt[i];
            }
        }
        ta[i] = ct[i] - ar[i] ;    // turnaround time= completion time- arrival time
        wt[i] = ta[i] - bt[i] ;    // waiting time= turnaround time- burst time
        avgwt += wt[i] ;          // total waiting time
        avgta += ta[i] ;          // total turnaround time
    }
    System.out.println("\npid arrival burst complete turn waiting");
    for(int i = 0 ; i < n; i++)
    {
        System.out.println(pid[i] + " \t " + ar[i] + "\t" + bt[i] + "\t" + ct[i] + "\t" + ta[i] + "\t" + wt[i] ) ;
    }
    sc.close();
    System.out.println("\naverage waiting time: "+ (avgwt/n));    // printing average waiting time.
    System.out.println("average turnaround time: "+(avgta/n));    // printing average turnaround
time.
    }
}

```


Date :

GATE Educational Institutions

Page No.:

Practical No. :

output :-

enter no. of process :-

3

enter process 1 arrival time

1

enter process 1 burst time

12

enter process 2 arrival time

2

enter process 2 burst time

2

enter process 3 arrival time

3

enter process 3 arrival time

5

Pid	arrival	burst	complete	turn	waiting
1	1	12	13	12	0
2	2	2	15	13	11
3	3	5	20	17	12

average waiting time : 7.66

average turnaround

time : 14.0

conclusion :-

The above java program executed successfully

Date :	GATE Educational Institutions	Page No.:
	shortest job first scheduling	Practical No. :

problem definition:- design and develop a java program for implementing SJF cpu scheduling algorithm.

Description:-

SJR stands for shortest job first

- * SJF algorithm can be preemptive or non-preemptive
- * In preemptive SJF, the process are put into the ready queue as they come and process with shortest burst time will first begin its execution
- * In non-preemptive SJF once the cpu is allocated the process the process holds it until it reaches a waiting state can be terminated

Algorithm:-

Step 1 : start the process

Step 2 : accept the no of process into ready queue.

Step 3 : for each process in the ready queue assign the arrival time and burst time

Step 4 : sort all the process according to the arrival time by taking a minimum value

Step 5 : complete time for all the processes if $at[i] \leq st \& \& (F[i] == 0) \& \&$
 $cbt[i] < minimum$ then $minimum = bt[i]$ $c=i$

Date :

GATE Educational Institutions

Page No.:

Practical No. :

step 6:- And if $cc == n$ then

$st++$ else

calculate the turn around time

$$ta[c] = ct[c] - at[c]$$

calculate the waiting time

$$wt[c] = ta[c] - bt[c]$$

calculate the total turn around time

$$avg\ tat = ta[i]$$

calculate the total waiting time

$$avg\ wtt = wt[i]$$

step 7:- print the average waiting time $avg\ wtt/n$

print the average turn around time $avg\ tat/n$

step 8:- stop

Date :	GATE Educational Institutions	Page No. :
		Practical No. :

3. Design and Develop a Java Program to implement SJF CPU Scheduling Algorithm

```
import java.util.*;

public class SJF {
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println ("enter no of process:");
        int n = sc.nextInt();
        int pid[] = new int[n];
        int at[] = new int[n]; // at means arrival time
        int bt[] = new int[n]; // bt means burst time
        int ct[] = new int[n]; // ct means complete time
        int ta[] = new int[n]; // ta means turn around time
        int wt[] = new int[n]; //wt means waiting time
        int f[] = new int[n]; // f means it is flag it checks process is completed or not
        int st=0, tot=0;
        float avgwt=0, avgta=0;

        for(int i=0;i<n;i++)
        {
            System.out.println ("enter process " + (i+1) + " arrival time:");
            at[i] = sc.nextInt();
            System.out.println ("enter process " + (i+1) + " burst time:");
            bt[i] = sc.nextInt();
            pid[i] = i+1;
            f[i] = 0;
        }
        boolean a = true;
        while(true)
        {
            int c=n, min=999;
            if (tot == n) // total no of process = completed process loop will be terminated
                break;
            for (int i=0; i<n; i++)
            {
                /*
                 * If i'th process arrival time <= system time and its flag=0 and burst<min
                 * That process will be executed first
                 */
                if ((at[i] <= st) && (f[i] == 0) && (bt[i]<min))
```


Date :

GATE Educational Institutions

Page No. :

Practical No. :

```
{
    min=bt[i];
    c=i;
}
}
/* If c==n means c value can not updated because no process arrival time< system time so we
increase the system time */
if (c==n)
    st++;
else
{
    ct[c]=st+bt[c];
    st+=bt[c];
    ta[c]=ct[c]-at[c];
    wt[c]=ta[c]-bt[c];
    f[c]=1;
    tot++;
}
}
System.out.println("\npid arrival burst complete turn waiting");
for(int i=0;i<n;i++)
{
    avgwt+= wt[i];
    avgta+= ta[i];
    System.out.println(pid[i]+"\\t"+at[i]+"\\t"+bt[i]+"\\t"+ct[i]+"\\t"+ta[i]+"\\t"+wt[i]);
}
System.out.println ("\naverage tat is "+ (float)(avgta/n));
System.out.println ("average wt is "+ (float)(avgwt/n));
sc.close();
}
}
```


Date :

GATE Educational Institutions

Page No.:

Practical No. :

output :-

enter no of process

3

enter process 1 arrival time

1

enter process 1 burst time

5

enter process 2 arrival time

2

enter process 2 burst time

4

enter process 3 arrival time

3

enter process 3 burst time

6

pid	arrival	burst	complete	turn	waiting
1	1	5	6	5	0
2	2	4	10	8	4
3	3	6	16	13	17

average turn around time is : 8.66

average waiting time is :- 3.66

conclusion :-

The above java program executed successfully

Date :	GATE Educational Institutions	Page No.:
	Round Robin scheduling	Practical No. :

Problem definition:- design and develop a java program for implementing Round Robin scheduling algorithm

Description:-

- * Round Robin is a cpu scheduling algorithm where each process is assigned a fixed time slot in a cyclic way.
- * it gives the best performance in terms of average response time
- * it is best suited for the time sharing system client server architecture and interaction system.
- * The disadvantages of Round Robin is the performance may, heavily depend on time quantum.

Algorithm:-

step 1 : Start the process

step 2 : Accept the time quantum of sharing system.

step 3 : Accept the no. of process into the ready queue.

step 4 : for each process assign the arrival time and the burst time.

step 5 : sort all the process are completed arrival time and make sure it follows time quanta

Date :	GATE Educational Institutions	Page No.:
		Practical No. :

step 6: check whether the all process are or not

step 7: calculate the turn around time.

$$\text{Turn}[i] = \text{turn}[i] - \text{arrival}[i]$$

calculate the waiting time

$$\text{wait}[i] = \text{turn}[i] - \text{burst}[i]$$

calculate the total turn around time

$$\text{avg tat} = \text{turn}[i]$$

calculate the total waiting time

$$\text{avg wait t} = \text{wait}[i]$$

Step 8: print the average waiting time

$$(\text{avg wait}/n)$$

print the average waiting time

$$(\text{avg wait}/n)$$

print the average turn around time

$$(\text{avg tt}/n)$$

step 9: stop the process

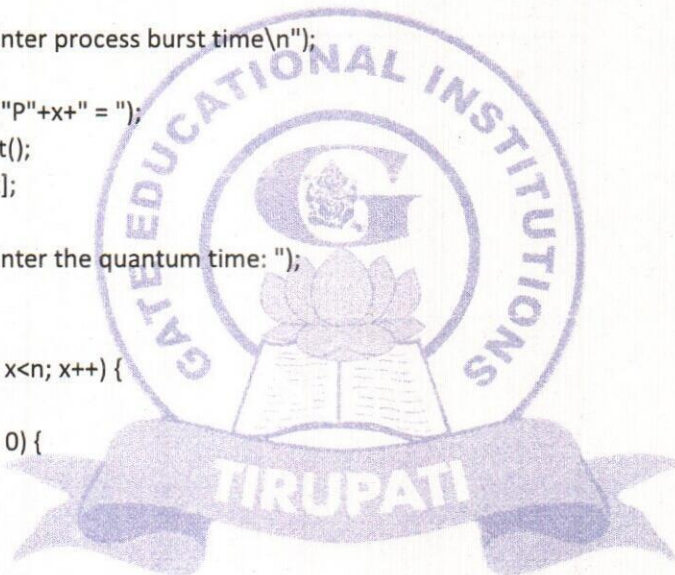
Date :	GATE Educational Institutions	Page No. :
		Practical No. :

4. Design and Develop a Java Program to implement ROUND ROBIN CPU Scheduling Algorithm

```

import java.util.Scanner;
public class RoundRobinExample {
    public static void main(String args[]) {
        int n,x,q_t,count=0,temp,sq=0,b_t[],w_t[],ta_t[],rem_bt[];
        float aw_t=0,ata_t=0;
        b_t = new int[10];
        w_t = new int[10];
        ta_t = new int[10];
        rem_bt = new int[10];
        Scanner sc=new Scanner(System.in);
        System.out.print("Number of processes(maximum 10) = ");
        n = sc.nextInt();
        System.out.print("Enter process burst time\n");
        for (x=0; x<n; x++) {
            System.out.print("P"+x+" = ");
            b_t[x]= sc.nextInt();
            rem_bt[x] = b_t[x];
        }
        System.out.print("Enter the quantum time: ");
        q_t = sc.nextInt();
        while(true) {
            for (x=0,count=0; x<n; x++) {
                temp = q_t;
                if(rem_bt[x] == 0) {
                    count++;
                    continue;
                }
                if(rem_bt[x]>q_t)
                    rem_bt[x]= rem_bt[x] - q_t;
                else if(rem_bt[x]>=0) {
                    temp = rem_bt[x];
                    rem_bt[x] = 0;
                }
                sq = sq + temp;
                ta_t[x] = sq;
            }
            if(n == count)
                break;
        }
    }
}

```



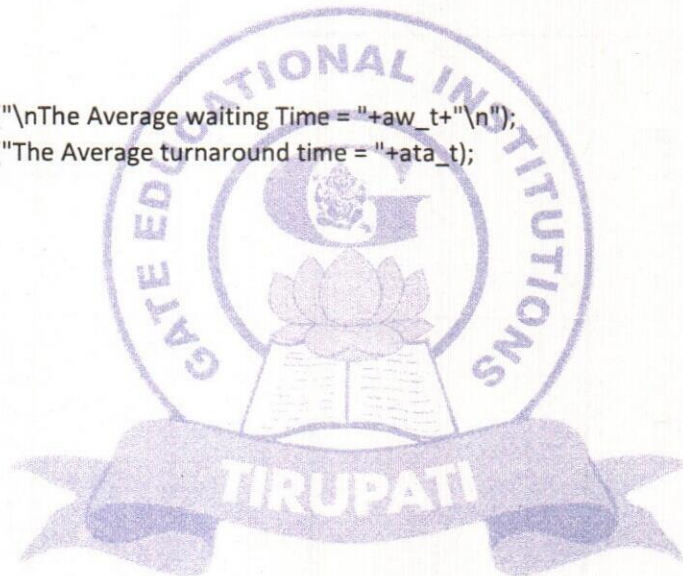
Date :	GATE Educational Institutions	Page No. :
		Practical No. :

```

System.out.print("*****
*****");
    System.out.print("\nProcess\t Burst Time\t Turnaround Time\t Waiting Time\n");

System.out.print("*****
*****");
    for(x=0; x<n; x++) {
        w_t[x]=ta_t[x]-b_t[x];
        aw_t=aw_t+w_t[x];
        ata_t=ata_t+ta_t[x];
        System.out.println("\n "+(x+1)+"\t "+b_t[x]+\t\t "+ta_t[x]+\t\t "+w_t[x]+\n");
    }
    aw_t=aw_t/n;
    ata_t=ata_t/n;
    System.out.println("\nThe Average waiting Time = "+aw_t+"\n");
    System.out.println("The Average turnaround time = "+ata_t);
}
}

```



Date :

GATE Educational Institutions

Page No.:

Practical No. :

output :-

enter the time quantum

2

enter the no. of process

3

enter the arrival time of the process

1

2

3

enter the burst time of the process

4

5

6

program
noArrival
timeBurst
timeWait
timeTurn around
time

1

1

4

4

8

2

2

5

7

12

3

3

6

7

13

Average wait time : 6.0

Average turn around time : 11.0

conclusion :-

The above java program executed successfully

Date :	GATE Educational Institutions	Page No.:
	First Fit	Practical No. :

program Definition :- design and develop a first time memory allocation program in java

Description :-

- * first fit memory allocation keeps the free busy list of jobs organized by memory allocation low ordered to high ordered memory.
- * In this method first job claims the first job claims the first available memory with space more than and equal to its size
- * its is fast in processing in As the allocation the nearest available memory position to the job it is very fast in execution.

Algorithm :-

step 1 : start the process

step 2 : create a class first fit

step 3 : declare jobsize [], memory size [], blocks, jobs, the counter, jobIndex Robin queue variables.

step 4 : create sets jobs, set memory first fit method and show data method

Date :	GATE Educational Institutions	Page No.:
		Practical No. :

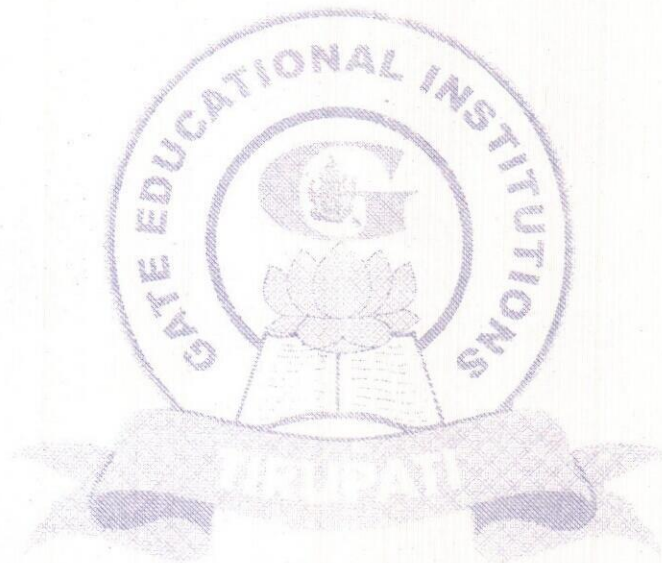
step 5: create main method

step 6: create object "ob" for first fit class

step 7: pass the parameter's to the methods throw objects

step 8: pass the parameters to the methods throw object

step 9: stop the program



Date :	GATE Educational Institutions	Page No. :
		Practical No. :

5. Design and Develop a Java Program to implement FIFO Memory Allocations (FIRSTFIT)

```

class FirstFit
{
    static void firstFit(int blockSize[], int m, int processSize[], int n)
    {
        int allocation[] = new int[n];
        for (int i = 0; i < allocation.length; i++)
            allocation[i] = -1;
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < m; j++)
            {
                if (blockSize[j] >= processSize[i])
                {
                    allocation[i] = j;
                    blockSize[j] -= processSize[i];
                    break;
                }
            }
        }
        System.out.println("\nProcess No.\tProcess Size\tBlock no.");
        for (int i = 0; i < n; i++)
        {
            System.out.print(" " + (i+1) + "\t\t" +
                processSize[i] + "\t\t");
            if (allocation[i] != -1)
                System.out.print(allocation[i] + 1);
            else
                System.out.print("Not Allocated");
            System.out.println();
        }
    }
    public static void main(String[] args)
    {
        int blockSize[] = {100, 500, 200, 300, 600};
        int processSize[] = {212, 417, 112, 426};
        int m = blockSize.length;
        int n = processSize.length;
        firstFit(blockSize, m, processSize, n);
    }
}

```


Date :	GATE Educational Institutions	Page No.:
		Practical No. :

output :-

=====

Available memory blocks are : (29.0), (19.0)(10.0), (40.0), (35.0)

And jobs to allocate are :- (45.0), (20.0), (40.0) (42.0) (18.0)

=====

START.

=====

Job1 of size 45.0 is sent to waiting queue

Job2 of size of the memory block 1 is now 9.0

Job3 of size 40.0 has been loaded into memory block:4

The size of the memory block 4 is now 0.0

Job4 of size 42.0 is sent to waiting queue

conclusion :-

The above java program executed successfully

Date :	GATE Educational Institutions	Page No.:
	Best Fit	Practical No. :

Program definition :- design and develop a best fit memory program in java.

Description :-

- * Best fit keeps the free/busy list in order size smallest to largest
- * In this method, as first search the whole of memory
- * According to the size job and allocates it to the closest fitting free partition in the memory.
- * memory efficient the OS allocates the job minimum possible space in the memory
- * memory efficient the OS allocates the job minimum possible space in the memory making memory management very efficient.

Algorithm :-

- step 1 : start the process
- step 2 : create a class Best fit
- step 3 : create Best fit static method in a class
- step 4 : create a main method in a class
- step 5 : declare block size array with values

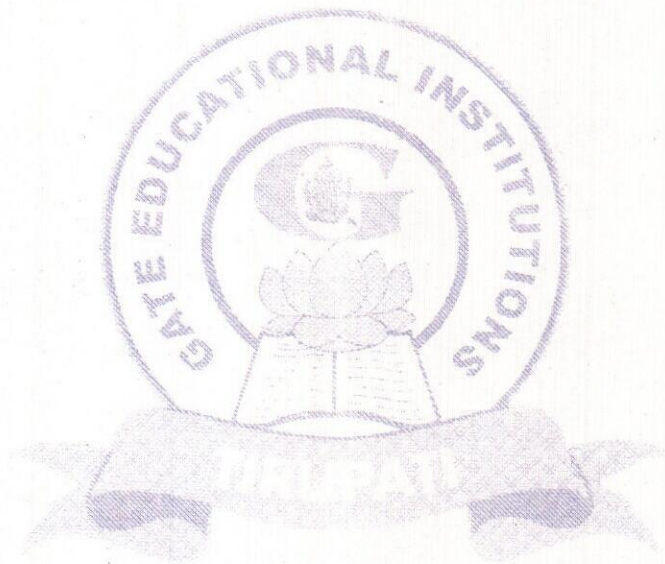
Date :	GATE Educational Institutions	Page No.:
		Practical No. :

step 6 : declare a process size with value

step 7 : declare a process size with values

step 8 : call the static method best-fit and passing parameters

step 9 : stop the program.



Date :

GATE Educational Institutions

Page No. :

Practical No. :

6. Design and Develop a Java Program to implement Best Fit Memory Allocations

```

public class BestFit
{
    static void bestFit(int blockSize[], int m, int processSize[],int n)
    {
        int allocation[] = new int[n];
        for (int i = 0; i < allocation.length; i++)
            allocation[i] = -1;
        for (int i=0; i<n; i++)
        {
            int bestIdx = -1;
            for (int j=0; j<m; j++)
            {
                if (blockSize[j] >= processSize[i])
                {
                    if (bestIdx == -1)
                        bestIdx = j;
                    else if (blockSize[bestIdx] > blockSize[j])
                        bestIdx = j;
                }
            }
            if (bestIdx != -1)
            {
                allocation[i] = bestIdx;
                blockSize[bestIdx] -= processSize[i];
            }
        }
        System.out.println("\nProcess No.\tProcess Size\tBlock no.");
        for (int i = 0; i < n; i++)
        {
            System.out.print(" " + (i+1) + "\t\t" + processSize[i] + "\t\t");
            if (allocation[i] != -1)
                System.out.print(allocation[i] + 1);
            else
                System.out.print("Not Allocated");
            System.out.println();
        }
    }
}

public static void main(String[] args)
{
    int blockSize[] = {100, 500, 200, 300, 600};

```


Date :	GATE Educational Institutions	Page No. :
		Practical No. :

```

int processSize[] = {212, 417, 112, 426};
int m = blockSize.length;
int n = processSize.length;
bestFit(blockSize, m, processSize, n);
}
}

```



Date :

GATE Educational Institutions

Page No.:

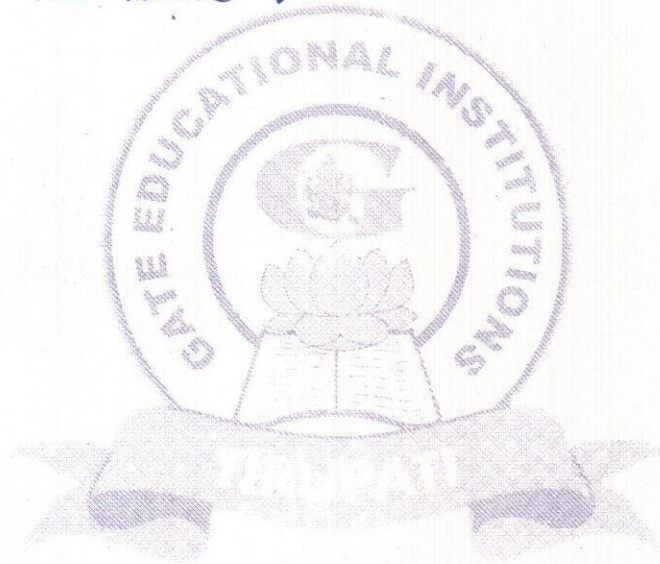
Practical No. :

output :-

process no	process size	Block No
1	412	2
2	317	4
3	434	1
4	542	Not allocated

conclusion :-

The above java program executed successfully



Date :	GATE Educational Institutions	Page No.:
	Worst Fit	Practical No. :

program definition :- design and develop a worst fit memory allocation program in java

Description :-

- * Best fit keeps the free/busy list in order by size by smallest to largest.
- * In this method as first searches the whole of memory according to the size of the give job and allocation it to the closed fitting free partition in the memory.
- * Memory :- efficient the os allocator the job minimum possible space in the memory making memory management very efficient

Algorithm :-

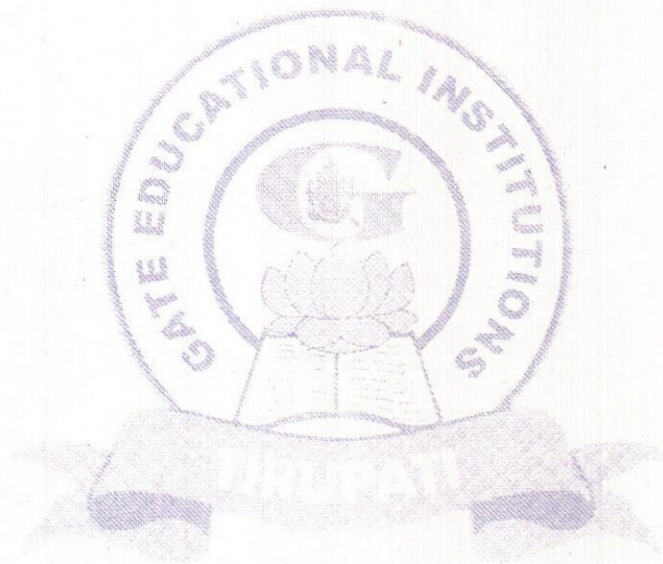
- step 1 : start the program
- step 2 : create a class Best fit
- step 3 : create Best fit static method in a class
- step 4 : create a main method
- step 5 : declare 'a' block size with integer values
- step 6 : declare a block size process size with inter values

Date :	GATE Educational Institutions	Page No.:
		Practical No. :

Step 7 : declare $m = \text{block size length}$ and
 $n = \text{process size length}$

Step 8 : call the static method worst fit and passing parameters

Step 9 : stop the process.



7. Design and Develop a Java Program to implement Worst Fit Memory Allocations

```
public class WorstFit
```

```
{
```

```
    static void worstFit(int blockSize[], int m, int processSize[],int n)
```

```
    {
```

```
        int allocation[] = new int[n];
```

```
        for (int i = 0; i < allocation.length; i++)
```

```
            allocation[i] = -1;
```

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

```
        {
```

```
            int wstIdx = -1;
```

```
            for (int j=0; j<m; j++)
```

```
            {
```

```
                if (blockSize[j] >= processSize[i])
```

```
                {
```

```
                    if (wstIdx == -1)
```

```
                        wstIdx = j;
```

```
                    else if (blockSize[wstIdx] < blockSize[j])
```

```
                        wstIdx = j;
```

```
                }
```

```
            }
```

```
            if (wstIdx != -1)
```

```
            {
```

```
                allocation[i] = wstIdx;
```

```
                blockSize[wstIdx] -= processSize[i];
```

```
            }
```

```
        }
```

```
        System.out.println("\nProcess No.\tProcess Size\tBlock no.");
```

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

```
        {
```

```
            System.out.print(" " + (i+1) + "\t\t" + processSize[i] + "\t\t");
```

```
            if (allocation[i] != -1)
```

```
                System.out.print(allocation[i] + 1);
```

```
            else
```

```
                System.out.print("Not Allocated");
```

```
            System.out.println();
```

```
        }
```

```
    }
```

```
    public static void main(String[] args)
```

```
    {
```

```
        int blockSize[] = {100, 500, 200, 300, 600};
```

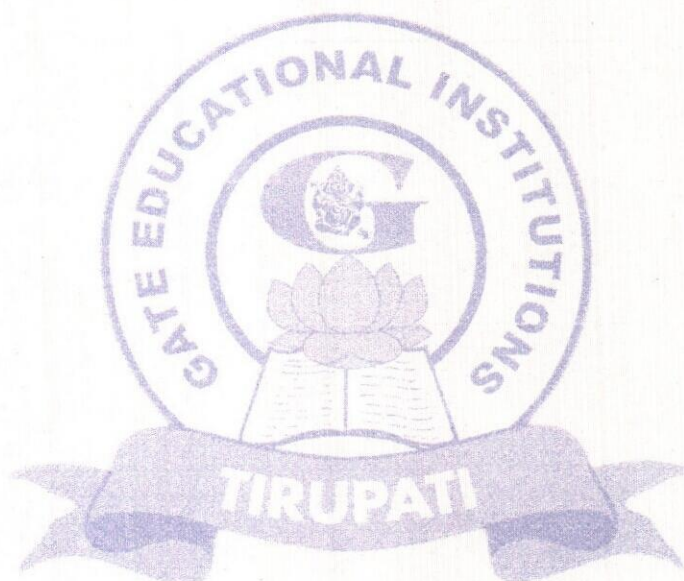
```
        int processSize[] = {212, 417, 112, 426};
```


Date :	GATE Educational Institutions	Page No. :
		Practical No. :

```

    int m = blockSize.length;
    int n = processSize.length;
    worstFit(blockSize, m, processSize, n);
}
}

```



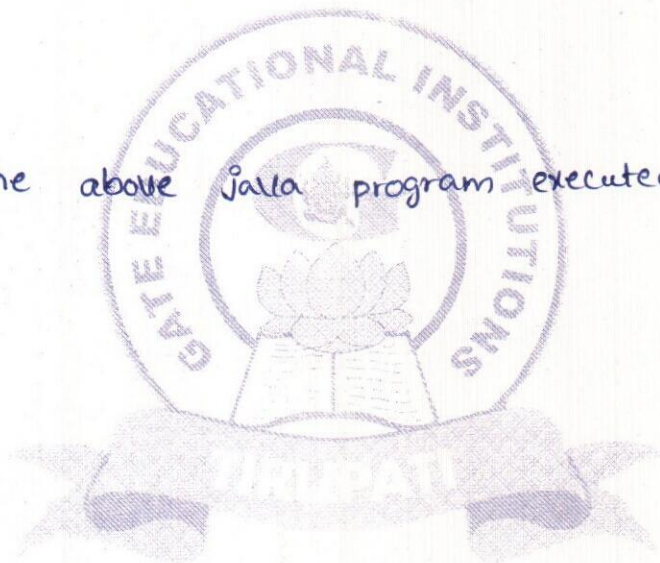
Date :	GATE Educational Institutions	Page No.:
		Practical No. :

Output :-

process no	process size	Block no
1	212	3
2	111	1
3	356	Not allocated
4	265	Not allocated

Conclusion :-

The above Java program executed successfully



Date :	GATE Educational Institutions	Page No.:
	Bankers Algorithm	Practical No. :

Program definition :- Design and develop the Banker's Algorithm to implement dead lock avoidance in java.

Description :-

- * Banker's algorithm is used to avoid deadlock and allocates resources safely to each process in the computer system.
- * The Banker's algorithm mainly work on three attributes is max requested allocated resources available resources.
- * It helps the operating system to manage and control process request for each type of resource s in the computer system.

Algorithm :-

Step 1 :- Start the process

Step 2 :- Accept no. of resource and total no. of instance to resources assign maximum and allocation string.

Step 3 :- To calculate the available resource by $cur-avail[i] = cur[j] - alloc[i][j];$

Step 4 :- To calculate the need resource by $need[j][i] = Max[j][i] - alloc[j][i];$

Step 5 :- To check the system in safe state boolean state check state (need, alloc, resn) pros-n)

Date :	GATE Educational Institutions	Page No.:
		Practical No. :

Step 6: if system is in safe state the print safe sequence.

Step 7: if any process requesting more resources the display command to user please enter the request matrix

Step 8: Again check the safe state then check the available resources are great than requesting resource to grant other wise donot grant to it.

output:—

please enter the total no. of resources

3

enter total number of instances for resource 1.

2

enter total number of instances for resource 2.

4

enter total number of instances for resource 3

3

enter no. of process:—

Enter the maximum string for process 1

321

Enter the maximum string for process 2

456

enter the maximum string for process 3

987

Date :	GATE Educational Institutions	Page No.:
		Practical No. :

Enter the Allocation string for process 1

123

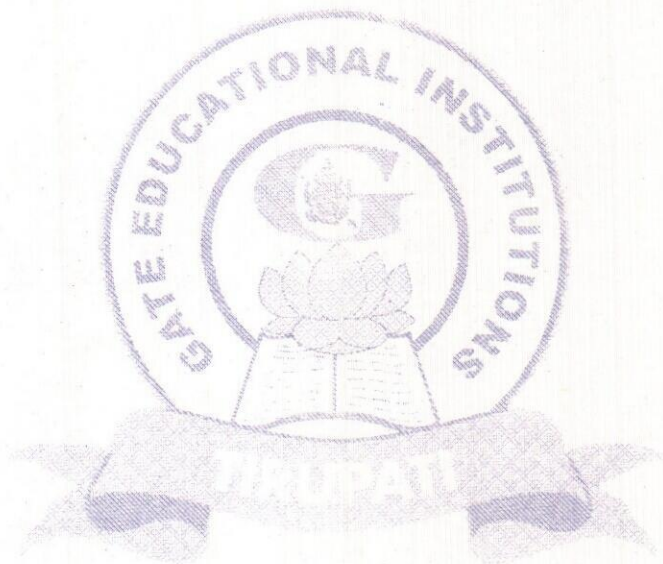
Enter the Allocation string for process 2

654

Enter the Allocation string for process 3

784

The system is not safe state



Date :	GATE Educational Institutions	Page No. :
		Practical No. :

8. Design and Develop a Java Program to implement Bankers Algorithm Example

```

import java.util.*;
import java.io.*;
import java.util.Scanner;
class BankersAlgoExample
{
    static void findNeedValue(int needArray[][], int maxArray[][], int allocationArray[][], int
totalProcess, int totalResources)
    {
        for (int i = 0 ; i < totalProcess ; i++){ // for each process
            for (int j = 0 ; j < totalResources ; j++){ //for each resource
                needArray[i][j] = maxArray[i][j] - allocationArray[i][j];
            }
        }
    }

    static boolean checkSafeSystem(int processes[], int availableArray[], int maxArray[][], int
allocationArray[][], int totalProcess, int totalResources)
    {
        int [][]needArray = new int[totalProcess][totalResources];
        findNeedValue(needArray, maxArray, allocationArray, totalProcess, totalResources);
        boolean []finishProcesses = new boolean[totalProcess];
        int []safeSequenceArray = new int[totalProcess];
        int []workArray = new int[totalResources];
        for (int i = 0; i < totalResources ; i++)
            workArray[i] = availableArray[i];
        int counter = 0;
        while (counter < totalProcess)
        {
            boolean foundSafeSystem = false;
            for (int m = 0; m < totalProcess; m++)
            {
                if (finishProcesses[m] == false) // when process is not finished
                {
                    int j;
                    for (j = 0; j < totalResources; j++)
                        if (needArray[m][j] > workArray[j])
                            break;
                    if (j == totalResources)
                    {
                        for (int k = 0 ; k < totalResources ; k++)
                            workArray[k] += allocationArray[m][k];
                        safeSequenceArray[counter++] = m;
                    }
                }
            }
        }
    }
}

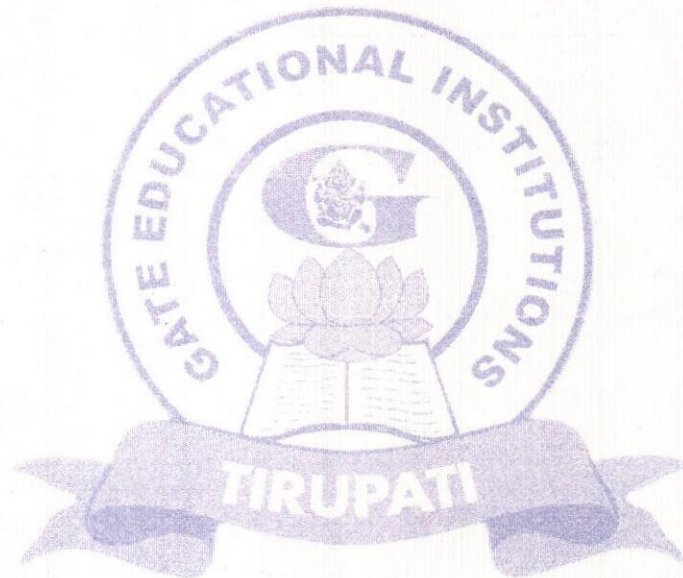
```


Date :	GATE Educational Institutions	Page No. :
		Practical No. :

```

for( int i = 0; i < numberOfProcesses; i++){
    for( int j = 0; j < numberOfResources; j++){
        System.out.println("How many instances of resource"+ j +" are allocated to process"+ i +"?
");
        allocationArray[i][j] = sc.nextInt();
    }
}
checkSafeSystem(processes, availableArray, maxArray, allocationArray, numberOfProcesses,
numberOfResources);
}
}

```



Date :	GATE Educational Institutions	Page No.:
	First in first out page replacement	Practical No. :

TO Design and develop a java program to illustrate the concept of FIFO page replacement algorithm.

Description :-

This is the simplest page replacement algorithm the operating system keeps track of all pages in the memory in a queue. The oldest page needs to be replaced first page in the queue will be removed.

Algorithm :-

- step 1 : start the process
- step 2 : declare the size with respect to page length
- step 3 : check the need of replacement from the page 'memory'
- step 4 : check the need of replacement from old page to new page in a memory.
- step 5 : from a queue to hold all pages
- step 6 : Insert the page require memory into queue
- step 7 : check the replacement and page fault
- step 8 : Get the number of process inserted
- step 9 : display values
- step 10 : stop

Date :	GATE Educational Institutions	Page No. :
		Practical No. :

9. Design and Develop a Java Program to implement FIFO page replacement algorithm

```

import java.io.*;
class FIFO
{
    public static void main(String args[]) throws IOException
    {
        int n;
        int f;

        float rat;
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter the number of FRAMES :");
        f=Integer.parseInt(br.readLine());
        int fifo[]=new int[f];
        System.out.println("Enter the number of INPUTS :");
        n=Integer.parseInt(br.readLine());
        int inp[]=new int[n];
        System.out.println("Enter INPUT:");
        for(int i=0;i<n;i++)
            inp[i]=Integer.parseInt(br.readLine());
        System.out.println("-----");
        for(int i=0;i<f;i++)
            fifo[i]=-1;
        int Hit=0;
        int Fault=0;
        int j=0;
        boolean check;
        for(int i=0;i<n;i++)
        {
            check=false;

            for(int k=0;k<f;k++)
                if(fifo[k]==inp[i])
                {
                    check=true;
                    Hit=Hit+1;
                }
            if(check==false)

```

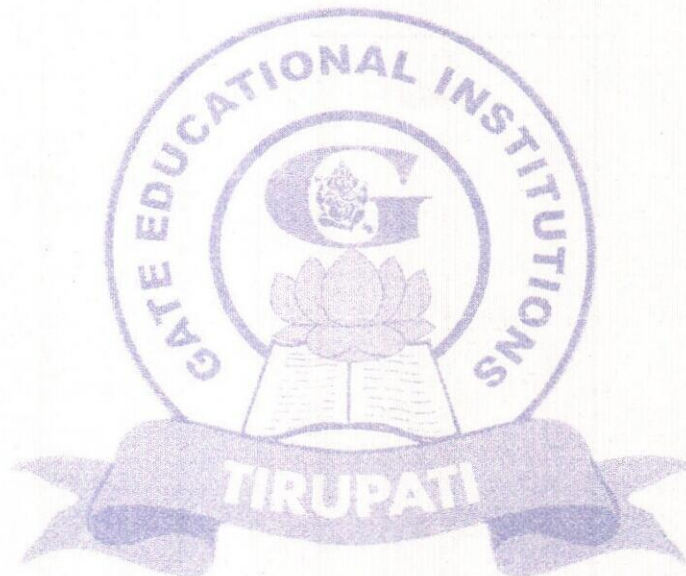

Date :	GATE Educational Institutions	Page No. :
		Practical No. :

```

        {
            fifo[j]=inp[i];
            j++;
            if(j>=f)
            j=0;
            Fault=Fault+1;
        }

    }
    rat = (float)Hit/(float)n;
    System.out.println("HIT:"+Hit+" FAULT:"+Fault+" HIT RATIO:"+rat);
}

```



Date :	GATE Educational Institutions	Page No.:
		Practical No. :

output :-

please enter the number of frames :- 3

please enter the length of the Reference string :- 8

please enter the reference string

2

3

4

3

6

7

9

2 2 2 2 5 5 5 9

7 3 3 3 3 6 6 6

7 7 4 4 4 4 7 7

the number of hits :- 1

hit ratio :- 0.125

the number of fault :- 7

conclusion :-

The above java program executed successfully

Date :	GATE Educational Institutions	Page No.:
	optimal page replacement	Practical No. :

program definition :- TO design and develop a JAVA program to optimal page replacement algorithm

Description :-

In operating system whenever a new page is referred and not presented in memory page found fault occurs and operating system replaces one of the existing pages with newly needed page different page replacement algorithms suggest different ways to decide which page to replace.

Algorithm :-

step 1 : start

step 2 : import class Buffered Reader and io exception

step 3 : import input stream reader

step 4 : declare class name

step 5 : declare main class method using throws key board

step 6 : declare int - reference and new - layout [][] variable

step 7 : enter number of frames

step 8 : enter length of the reference string

step 9 : using "for" loop

print number of hits

number of faults

ratio

step 10 : stop

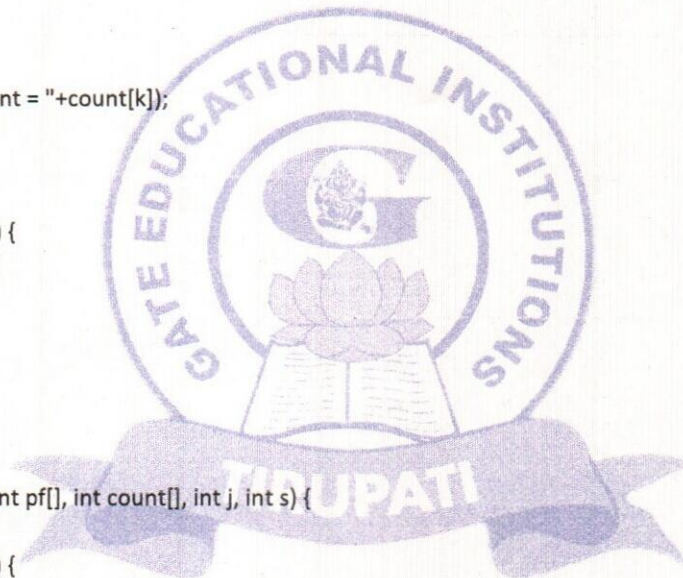
Date :	GATE Educational Institutions	Page No. :
		Practical No. :

10. Design and Develop a Java Program to implement Optimal page replacement algorithm

```

import java.util.*;
class PR3 {
static int check2(int p[], int pf[], int count[], int j, int s) {
    int k;
    for (int i = 0; i < s; ++i) {
        count[i] = 0;
    }
    for (k = 0; k < s; ++k) {
        loop:
        for (int i = j - 1; --i) {
            ++count[k];
            if (p[i] == pf[k]) {
                break loop;
            }
        }
    }
    //System.out.println("count = "+count[k]);
    }
    int max = count[0];
    int pos = 0;
    for (int l = 0; l < s; ++l) {
        if (count[l] > max) {
            max = count[l];
            pos = l;
        }
    }
    return pos;
}
static int check(int p[], int pf[], int count[], int j, int s) {
    int k;
    for (int i = 0; i < s; ++i) {
        count[i] = 0;
    }
    int found = 0;
    for (k = 0; k < s; ++k) {
        loop:
        for (int i = j + 1; i < p.length; ++i) {
            ++count[k];
            if (p[i] == pf[k]) {
                ++found;
                break loop;
            }
        }
    }
    int pos;
    if (found < (s - 1)) {

```



Date :	GATE Educational Institutions	Page No. :
		Practical No. :

```

        System.out.println();
        System.out.println("More than 1 used pages not required for further operation.\nSwapping LRU page.");
        pos = check2(p, pf, count, j, s);
    } else {
        int max = count[0];
        pos = 0;
        for (int l = 0; l < s; ++l) {
            if (count[l] > max) {
                max = count[l];
                pos = l;
            }
        }
    }
    return pos;
}

static boolean check1(int p[], int x, int n) {
    for (int i = 0; i < n; ++i) {
        if (p[i] == x) {
            return true;
        }
    }
    return false;
}

public static void main(String args[]) {
    Scanner scr = new Scanner(System.in);
    System.out.println("Enter the number of page entries");
    int n = scr.nextInt();
    int fault = 0, hit = 0;
    int p[] = new int[n];
    System.out.println("Enter the pages ");
    for (int i = 0; i < n; ++i) {
        p[i] = scr.nextInt();
    }
    System.out.print("The page table entries are ");
    for (int i = 0; i < n; ++i) {
        System.out.print(" " + p[i]);
    }
    System.out.println();
    int s = 3;
    System.out.println("Do you wish to set a the number of page frames?(1=yes/0=no)");
    int ch = scr.nextInt();

    if (ch == 1) {
        System.out.println("Enter the number of page frames ");
        s = scr.nextInt();
        System.out.println("The number of page frames set to " + s);
    }
}

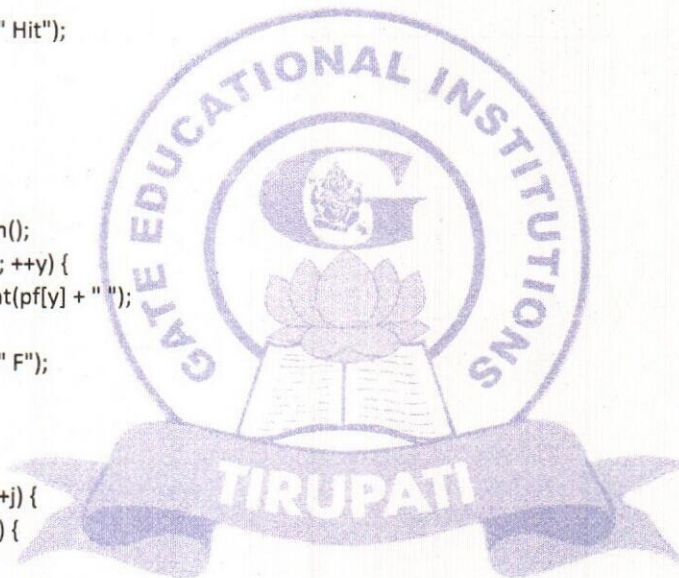
```


Date :	GATE Educational Institutions	Page No. :
		Practical No. :

```

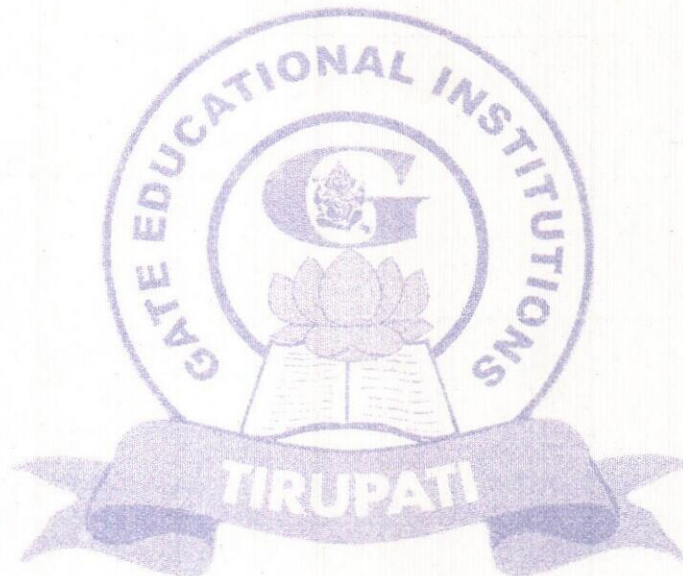
} else {
    System.out.println("The default number of page frames are " + s);
}
int count[] = new int[s];
int pf[] = new int[s];
int t = 0;
int incr = 0;
for (int i = 0; i < s || (t != s); ++i) {
    if (check1(pf, p[i], t)) {
        ++incr;
        System.out.println();
        for (int y = 0; y < t; ++y) {
            System.out.print(pf[y] + " ");
        }
        System.out.print(" Hit");
        ++hit;
    } else {
        ++incr;
        ++t;
        pf[t - 1] = p[i];
        System.out.println();
        for (int y = 0; y < t; ++y) {
            System.out.print(pf[y] + " ");
        }
        System.out.print(" F");
        ++fault;
    }
}
for (int j = incr; j < n; ++j) {
    if (check1(pf, p[j], s)) {
        ++hit;
        System.out.println();
        for (int k = 0; k < s; ++k) {
            System.out.print(pf[k] + " ");
        }
        System.out.print(" H");
    } else {
        int choice = check(p, pf, count, j, s);
        ++fault;
        pf[choice] = p[j];
        System.out.println();
        for (int k = 0; k < s; ++k) {
            System.out.print(pf[k] + " ");
        }
        System.out.print(" F");
    }
}

```



Date :	GATE Educational Institutions	Page No. :
		Practical No. :

```
}  
System.out.println();  
System.out.println("Page faults= " + fault + " Page hits= " + hit);  
}  
}
```



Date :

GATE Educational Institutions

Page No.:

Practical No. :

output :-

please enter the number of frames :- 3

please enter the length of reference string :- 10

please enter the reference string :-

4, 7, 6, 1, 7, 6, 1, 2, 7, 2

4 7 6 1 7 6 1 2 7 2

4 4 1 1 1 1 1 1 1 1

1 7 7 7 7 7 7 7 7 7

1 1 6 6 6 6 2 2 2 2

the number of hits :- 5

hit ratio :- 0.5

the number of faults :- 5

conclusion :-

The above program executed successfully

Date :	GATE Educational Institutions	Page No.:
	Least Recently used page replacement	Practical No. :

Program definition :-

Design and develop a java program to implement LRU (least recently used) page replacement algorithm

Description :- least recently used algorithm is a page replacement algorithm used for memory management according to this method the page which is recently used is replaced therefore in memory any page that has been unused for a longer period of time then the other is replaced therefore in memory any page that has been unused for a longer period of time then the other is replaced

Algorithm :-

step 1 : start

step 2 : Import utility package

step 3 : declare class name

step 4 : declare main method using throws keyword

step 5 : initially frames, pointer, fault refer variable

step 6 : initialize frames pointer, with fault and refer variable

step 7 : print number of frame

Date :	GATE Educational Institutions	Page No.:
		Practical No. :

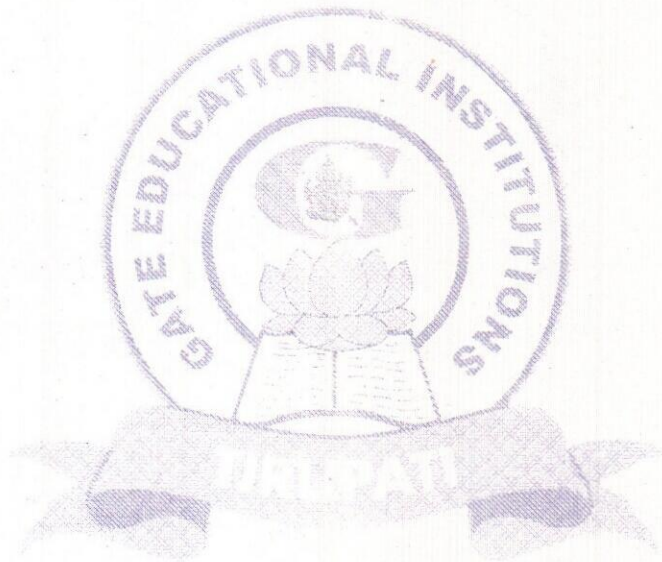
step 8 : frame = Integer parse Int (br. read lines)

step 9 : print length of the reference string

step 10 : enter reference string

step 11 : print number of page faults

step 12 : stop



Date :	GATE Educational Institutions	Page No. :
		Practical No. :

11. Design and Develop a Java Program to implement LRU page replacement algorithm

```

import java.io.*;
import java.util.*;
public class LRU {
    public static void main(String[] args) throws IOException
    {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        int frames,pointer = 0, hit = 0, fault = 0,ref_len;
        Boolean isFull = false;
        int buffer[];
        ArrayList<Integer> stack = new ArrayList<Integer>();
        int reference[];
        int mem_layout[][];

        System.out.println("Please enter the number of Frames: ");
        frames = Integer.parseInt(br.readLine());

        System.out.println("Please enter the length of the Reference string:");
        ref_len = Integer.parseInt(br.readLine());

        reference = new int[ref_len];
        mem_layout = new int[ref_len][frames];
        buffer = new int[frames];
        for(int j = 0; j < frames; j++)
            buffer[j] = -1;

        System.out.println("Please enter the reference string: ");
        for(int i = 0; i < ref_len; i++)
        {
            reference[i] = Integer.parseInt(br.readLine());
        }
        System.out.println();
        for(int i = 0; i < ref_len; i++)
        {
            if(stack.contains(reference[i]))
            {
                stack.remove(stack.indexOf(reference[i]));
            }
            stack.add(reference[i]);
            int search = -1;
            for(int j = 0; j < frames; j++)
            {

```


Date :

GATE Educational Institutions

Page No. :

Practical No. :

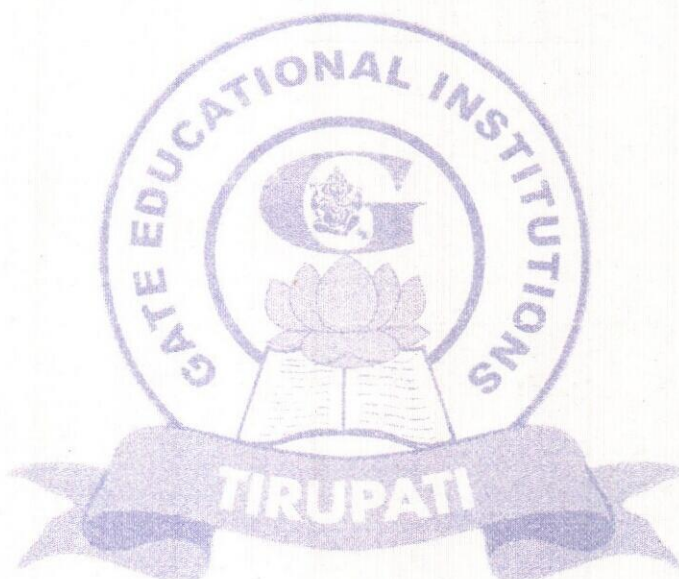
```
        if(buffer[j] == reference[i])
        {
            search = j;
            hit++;
            break;
        }
    }
    if(search == -1)
    {
        if(isFull)
        {
            int min_loc = ref_len;
            for(int j = 0; j < frames; j++)
            {
                if(stack.contains(buffer[j]))
                {
                    int temp = stack.indexOf(buffer[j]);
                    if(temp < min_loc)
                    {
                        min_loc = temp;
                        pointer = j;
                    }
                }
            }
        }
        buffer[pointer] = reference[i];
        fault++;
        pointer++;
        if(pointer == frames)
        {
            pointer = 0;
            isFull = true;
        }
    }
    for(int j = 0; j < frames; j++)
        mem_layout[i][j] = buffer[j];
}

for(int i = 0; i < frames; i++)
{
    for(int j = 0; j < ref_len; j++)
        System.out.printf("%3d ", mem_layout[j][i]);
```


Date :	GATE Educational Institutions	Page No. :
		Practical No. :

```
        System.out.println();
    }

    System.out.println("The number of Hits: " + hit);
    System.out.println("Hit Ratio: " + (float)((float)hit/ref_len));
    System.out.println("The number of Faults: " + fault);
}
}
```



Date :

GATE Educational Institutions

Page No.:

Practical No. :

output :-

please enter the number of frames :- 4

please enter the length of the reference
string :- 14

please enter reference string :-

7,0,1,2,0,3,0,4,2,3,0,3,2,3

7	7	7	7	7	7	3	3	3	3	3	3
-1	0	0	0	0	0	0	0	0	0	0	0
-1	-1	1	1	1	1	1	4	4	4	4	4
-1	-1	-1	2	2	3	2	2	2	2	2	2

the number of hits :- 8

hit ratio :- 0.571428

the number of faults :- 6

conclusion :-

The above program executed successfully

Date :	GATE Educational Institutions	Page No.:
	First come first serve disk scheduling	Practical No. :

program definition?—

Design and develop a java program to implement fcfs disk scheduling algorithm.

Description?—

- * FCFS is the simplest disk scheduling algorithm
- * At the name suggests this algorithm extends requests is in the order then arrive in the disk queue.
- * the algorithm looks very fair and there is no starvation but generally it does not provides the fastest service

Algorithm?—

- step 1: start the process
- step 2: Define FCFS function and initialize parameters
- step 3: check the condition, start for loop
- step 4: print total number of operations
- step 5: print the sequence
- step 6: declare main method
- step 7: initialize the elements in the array
- step 8: call the parameters in the function
- step 9: stop the program.

Date :	GATE Educational Institutions	Page No. :
		Practical No. :

12. Design and Develop a Java Program to implement FCFS Disk Scheduling Algorithm

```

class FCFS
{
    static int size = 8;
    static void FCFS(int arr[], int head)
    {
        int seek_count = 0;
        int distance, cur_track;
        for (int i = 0; i < size; i++)
        {
            cur_track = arr[i];
            distance = Math.abs(cur_track - head);
            seek_count += distance;
            head = cur_track;
        }
        System.out.println("Total number of " + "seek operations = " + seek_count);
        System.out.println("Seek Sequence is");

        for (int i = 0; i < size; i++)
        {
            System.out.println(arr[i]);
        }
    }
    public static void main(String[] args)
    {
        // request array
        int arr[] = { 176, 79, 34, 60,
                     92, 11, 41, 114 };

        int head = 50;

        FCFS(arr, head);
    }
}

```


Date :

GATE Educational Institutions

03
Page No. :

import java.util.LinkedList;

public class Threadexample {

Practical No. :

```
    public static void main(String[] args) throws  
        InterruptedException
```

```
    {
```

```
        final PC pc = new PC();
```

```
        Thread t1 = new Thread(new Runnable() {  
            @Override
```

```
            public void run()
```

```
            {
```

```
                try {
```

```
                    pc.produce();
```

```
                }
```

```
                catch (InterruptedException e) {  
                    e.printStackTrace();
```

```
                }
```

```
            }
```

```
        });
```

```
        Thread t2 = new Thread(new Runnable() {  
            @Override
```

```
            public void run()
```

```
            {
```

```
                try {
```

```
                    pc.consume();
```

```
                }
```

```
                catch (InterruptedException e) {  
                    e.printStackTrace();
```

```
                }
```

```
            }
```

```
        });
```

```
        t1.start();
```

```
        t2.start();
```

```
        t1.join();
```

```
        t2.join();
```

```
    }
```

```
public static class PC {
```

```
    LinkedList<Integer> list = new LinkedList<>();
```

```
    int capacity = 2;
```


public void produce() throws InterruptedException

Date :	{ GATE Educational Institutions	Page No. :
	<pre>int value = 0; while (true) { synchronized (this) { while (list.size() == capacity) wait(); System.out.println("Producer produced-" + value); list.add(value++); notify(); Thread.sleep(1000); }} public void consume() throws InterruptedException { while (true) { synchronized (this) { while (list.size() == 0) wait(); int val = list.removeFirst(); System.out.println("Consumer consumed-" + val); notify(); Thread.sleep(1000); }} }</pre>	Practical No. :

Output:

Producer produced:0
Producer produced :1
Consumer consumed :0
Consumer consumed:1
Producer produced :2
Producer produced:3
Consumer consumed:2
Consumer consumed:3

052

Date :	GATE Educational Institutions	Page No.:
		Practical No. :

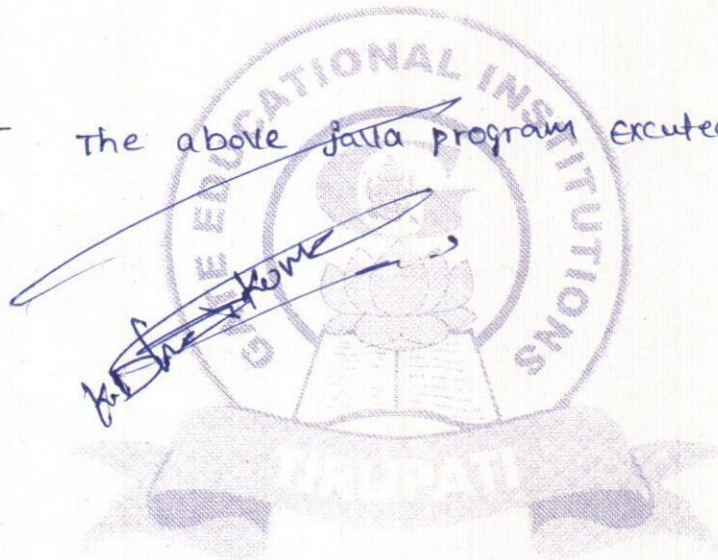
output :-

total number of seek operations = 53%

sequence is

176
79
34
60
92
11
70
140

conclusion :- The above java program executed successfully.



Verified by

~~for the purpose~~