INDEX

SL NO.	Date	Name of Experiment	Page No.	Remarks
1	11/3/22	producer Consumer using threads	1-5	
2	19/5/22	first come first Serve scheduling Algorithm	6-11	
3	25/3/22	shortest job first scheduling Algorithm	12-17	
4	26/3/22	Round Robin scheduling Algorithm	18-24	
5	8/4/22	first fit	25-30	
6	9/4/22	Best fit	31-35	
7	16/4/22	Worst fit	36-39	
8	22/4/22	Bankers Algorithm	40-47	
9	29/4/22	first In first out page replacement Algorithm	48-52	
10	30/4/22	Optimal page replacement Algorithm	53-58	

INDEX

SL NO.	Date	Name of Experiment	Page No.	Remarks
11	6/5/22	Least Recently used page Replacement Algorithm	59-64	
12	7/5/22	First Come First Serve Disk scheduling Algorithm	65-68	

Name of the Experiment: producer Consumer Using threads | Page no. 1

Program Definition :- Design and develop a Java program for Implementing producer Consumer Concepts for using threads.

Descriptions :-

* A thread is a single sequential flow of Exception of task of process. So it is known of thread of Exaustion (or) thread of control.

* Thread is often refered to as a light weight process.

* In os there are two types of threads
  * Kernel level thread
  * User level thread

Algorithm :-

step 1 :- start the program

step 2 :- Create class Thread Exauple

step 3 :- Create instayce for thread

step 4 :- Declare on method with try and catch stocks.

step 5 :- To override the rcn method from class pc

step 6 :- Creation exception

step 7 :- Create linked list in class pc

step 8 :- thread methods, start, join, sleep is used

step 9:- stop the program

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.

Name of the Experiment: First Come first Serve scheduling Algorithm

Page no. 5

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 temp t

ct [i] = ar[i] + bt[i], for the first process

and if Cat [i] > ct [i-1] the Calculable

ct[i] = at [i] + bt[i]

Else

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

oct-put

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	brust	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 turn aground time : 14.0

Conclusion :- the above java program Executed sucessfully

Name of the Experiment: Shortest Job first scheduling | Page no. 7

Problem definition :- Design and develop a Java Program for Implementing SJF cpu scheduling algorithm.

Description :- SJF stands for shortest Job first

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

Name of the Experiment:

Page no.

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] <= st && (f[i] == 0) &&

bt[i] < minimum then minimum = bt[i]

c = i

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 wt/n

print the average turn around time avg ta/n

step 8 :- stop

Out-put

```
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	wait Eng
1	1	5	6	5	0
2	2	4	10	8	4
3	3	6	16	13	7

average turn around time is : 8.66

average waiting time is : 3.66

Conclusion :- the above java program executed successfully

Name of the Experiment: Round Robin scheduling | Page no. 13

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 processes are completed arrival time and make sure it follows time quanta

step 6 :- Check whether the all process are Completed or not

step 7 :- Calculate the turn around time.

Turn [i] = turn [i] - arrival [i]

Calculate the waiting time

wait [i] = turn [i] - burst [i]

Calculate the total turn around time

avg tat = turn [i]

Calculate the total waiting time

avgwait t = wait [i]

Step8 :- print the average waiting time

(avg wait/n)

print the average waiting time

(avg wait/n)

print the average turn around time

(avgtt/n)

Step9 :- Stop the process

output

Enter the time quantam
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 no	Arrival time	Burst time	Wait time	Turn around time
1	1	4	4	8
2	2	5	7	12
3	3	6	7	13

Average Wait time: 6.0
Average Turnaround time: 11.0

the Coyclusion :- the above java program Executed successfully

}

Name of the Experiment: first fit

Program Definition :- Design and develop a first first time memory allocation program in Java

Description

* first fit memory allocation keeps the free busy list of jobs organised by memory allocation low ordered to high ordered memory.
* In this method first job claims the first job claims the first available memory with spam more than and Equal to its size
* Its is fast in processing in As the allocation the nearest available memory portion 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, Jobsinder Robin queue, variables.

step 4:- Create sets Jobs, set memory first fit method and show data method

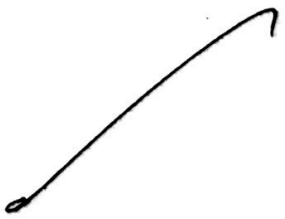
step 5:- Create main method

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

step 7:- pass the parameters to the methods throw objects

step 7 :- pass the parameters to the methods
throw object

step 8 :- stop the program.



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

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

Jobs 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

Name of the Experiment: Best fit | Page no. 21

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

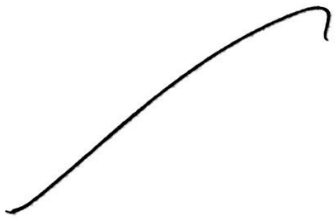
step 3:- Create Best fit static method in a class

step 4:- Create a main method in a class

step 5:- Declare block size array with values

step 6:- Declare a process size with values

step 7:- Decleare a process size with values



Step 8:- Call the static method Best fit and passing parameters

Step 9:- Stop the program.

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.

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 closest fitting free partition 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 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 Enter values

step 7:- Declare m = block size length and n = process size length

step 8:- Call the static method worst fit and passing parameters

step 9:- stop the process.

out-put

Process no	processize	Block no
1	212	3
2	111	1
3	356	Not allocated
4	285	Not allocated

Conclusion :- the above java program executed success fully

Name of the Experiment: Bankers Algorithm | Page no. 29

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 is in the Computer System.

Algorithm :-

step 1 :- Start the process

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

step 3 :- To calculate the available resource by cur-avai[i] = cur[i] - 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 stale check state [need, alloc, resn) pros-n)

step 6 :- if system is in safe state the point safe sequence.

step 7 :- if any process requesting more resources the display commond 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 dont grant to it.

out-put :-

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
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

Name of the Experiment: First In first out page replacement | Page no. 33

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 keeper 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.

Name of the Experiment:

Page no.

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.

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
	3	3	3	3	6	6	6
-1							

-1	-1	4	4	4	4	7	7

the number of hits :- 1
hit ratio :- 0.125
the number of faults :- 7

Conclusion :- the above java program executed successfully.

Name of the Experiment: Optimal page replacement | Page no. 34

Program definition :- To design and develop a Java Program to optimal page replacement algorithm

Description :-

In operating system whenever a newpage is reffered 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 Bubbered Reader and Io Exception

step3 :- Import Input Stream reader

step4 :- Declare class name

step5 :- Declare main class method using throws key board

step6 :- Declare int - reference and mem - layout [ ][ ] variable

step7 :- Enter number of frames

step8 :- Enter length of the reference string

step9 :- Using "per" loop

print number of hits

number of faults

ratio

step10 :- Stop

Name of the Experiment:

Page no. 39

out-put

Please Enter the number of frams:-3

please Enter the length of reference string :-10

please Enter the 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.

Name of the Experiment: Least Recently used page replacement | Page no. 40

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.

Name of the Experiment:

Algorithm :-

step 1 :- start

step 2 :- Import utility package

step 3 :- Declare class name

step 4 :- Declare main method using thrown Keyword

step 5 :- initially frame, pointer =0, fact refer variable

step 6 :- initialize frames pointer, wit fact and refbern variable

step 7 :- print number of frame

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

step 9 :- print length of the reference string

step 10 :- Enter reference string

step 11 :- print number of page faults

step 12 :- stop

Name of the Experiment:

Page no.

Out-put

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

Name of the Experiment: first come first serve disk scheduling | Page no. 44

Program definition :-

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

Description :-

* fcfs is the simplest disk scheduling algorithm
* As the name suggests this algorithm entertains requests is in the order they 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 initialise 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.

Name of the Experiment:

Output :-

Total number of seek operations = 539

Sequence is

176
79
34
60
92
11
70
140

Conclusion :- The above java program executed successfully.

122.