

## LABORATORY WORK BOOK

Name of the Student: KACHERLA SANTHOSH								ě,			_		1 .				
Class IT-B Semester 03							Roll Number										
Cou	rse Cod	e ALSD 10 Course	e Name	25 Labor	atory	2	3	9	5	1	A	1	2	G	3		
Nan	ne of the	Course Faculty	Raghou	a Rao	0				Fa	culty	/ ID :	IF	IRE.	109	124		
Exe	rcise Nu	mber06					Date 25 10 2024										
				KS AWARDED													
S. No.	Exercise Number	EXERCISE NAME	Aim/ Preparation	Algorithm	Source Code Calculations and Graphs			Program Execution Results and Error Analysis			→ v	Viva -	1	Total			
				Performano							r \	Voce		, otal			
			4	4			4 4			+	4	:	20				
1	6.1	Cloud Tech	G		4				4			(	e	()	20		
2	6.2	Edutech University	(		,					'							
3	63	Game Tech Studios	T.														
4	6.4	Mediane Hospital										3.					
5	6.5	ShopMart.	-		*1												
6																	
7																	
8																	
9			,	14	1							,					
10																	
11		ų.		<b>V</b> <sub>0</sub> -	74				6)								
12		`															

Signature of the Student

Signature of the Faculty

- 6. Paging Memory Management: -
- 6.1 Cloud Tech:

AIM: - Write a program for Cloud Tech Using Paging Memory Management.

PROGRAM: -

class Gloud Server:

def \_ init \_ ( Øelf, total \_ page, page \_ Øize):

Delf. total - pages = total - pages

Øelf. pages - Øizes = page- Dizes

Delf. memory = [None] \* total - pages

Delf. cworent\_page = 0

Delf. Processes

def allocate\_ nomony ( Delf, process\_ name,

requestéel - pages):

if Delf. Gwownt\_ pages + requested\_ pages >

Delf. total \_ pages:

Pount (f " Not enough memory for & Process\_ namey")

Veturn False

allocated\_ pages - list ( range ( Delf. < werent\_ page, Delf. current\_ page + requested\_ pages)) Delf. processes [ process\_ name ] = allocated\_pages for Page in allocated\_ Pages: Delf. memory [page ] = process \_ name Delf. Guvrent - page + = requested\_ Pages Print (f" { Process \_ name } allocated pages: fallented\_ pages &) return True. def memory\_ utilization (delf): utilized\_ pages = Sum (1 for pages in Self. memory if page is not None) total\_data = utilized\_pages \* Belf. page\_ Dize print (f " Total Memory utilized: { utilized\_Pages } Pages, of total-data y units of data") def fragmentation \_ analysis (self); tree\_pages = self. total \_ pages \_ self. current\_page Print (f " Free Pages remaining: (free\_pagesy")

```
if free_pages > 0:
      Print ( " potential fragmentation: There are
      free Pages left but Processes comnot be
       allocated more than this number")
     else:
      Print (" No fragmentation: All Memory 25
                allocated ")
TOTAL - PAGES = 100
PAGE _ SIZE = 4
Process _ requests = {
   " PYOCES A": 25.
   " Process B"; 15,
  "Process G"; 30,
Server = < lowd Server (TOTAL - PAGES, PAGE _ SIZE)
for Process, pages in Process - reguests, items ():
    Server. allowto_ memory (process, pages)
OUTPUT: - The Program you been executed Successfully.
```

6.2

Fidu Tech University:

AIM: - Write a Program for Paging Memory Management for Edutech University.

PROGRAM: -

Class University Server:

def \_\_ init \_ ( self, total \_ pages, Page \_ size):

Delf. total \_ Pages = total \_ Pages

Welf. Page - Size = Page - Size

Delf. memory = [None] \* total \_ Pages

Delf. = wovient \_ Page = 0

Delf. Dystems = of z

def allocate \_ memory ( Delf, Dystem\_name,

Yequested - Pages):

f Delf. = wovent - Page + requested - pages >

Delf. total - Pages:

Print (f" Not enough memory for

```
( Dystem _ name y . Allocation foiled ")
return False
 allocated - Pages = list (range ( delf. current - page,
   Delf. auvent_ Page + requested_ Pages).)
  Delf. Dystems [ Dystem-name] = allocated_Pages
 for page in allocated - pages:
  Delf. memory [page] = Dystem_name
Delf. current_page += Veguested_ Pages
Print (f" ( System_name) allocated Pages:
          ( allocated _ Pages y")
return True
def fragmentation_analysis (Delf):
 free_ pages = Delf. total_pages/- Delf. Ewwent_
 Prunt (f" Free pages Vernaining: (free pages j")
 if free_pages > 0:
```

```
Print (" Potential fragmentation: There are free
         Pages left, but they may be too few
         to fulfill large Vequests")
 else:
    Print ("No fragmentation: All memory
            has been allocated")
TOTAL_ PAGES = 200
 PAGE - SIZE = 8
Dystem - respuests . = 1
  "Student Records System": 40,
  "Faculty Management System": 25,
  "Library Enformation System": 30,
  "Online Lewining Platform": 35,
   Research Dolubase": 50
Berver = University. Server (TOTAL_PAGES,
                            PAGE - SIZE)
```

6.3

for Dystem, Pages in Dystem \_ requests. "Hems(): Derver. allerate - memory ( System, Pages) Derver. memory\_utilization () Derver. fragmentation\_amblysis() OUTPUT: -The OUTPUT has been Executed Successfully. Game Tech Studios: -AIM: - Write a Program for Paging Memory Management for Game Teah Studios. PROGRAM: class Gaming Server : def \_ init \_ (Delf, total \_ pages page \_ Dize): Delf. total - pages = total - Pages Delf. Page \_ Dize = Page \_ Dize Delf. memory = [None] \* total \_ Pages Delf. Zwoient \_ Page = 0

Dell. Devoiono = 13 det allocate\_ memory ( self, session\_name, Veguested\_ Pages): if Delf. coment\_page + requested\_Pages > Delf. total - Pages: Pount (f" Not enough memory for ! Dersi on - namey. Allocation failed") return False allocated\_ pages = list ( Vange (. Delf. = werent \_ Page, Self. - current - page + Yearnestool \_ Pages ) ) dely. Dessions [ Dession \_ name] = allocated\_ det memory sallization (Delf). Milizer - pages = Dum (1 for Pages in Delf. memory of Page is not None) total\_data = utilized \_ Pages \* Self, page 3012c

```
Print ( f " Total memory utilized: { utilized ! Pagesy
       pages, Etotal_data & write of game data")
def fragmentation_analysis (Self):
  tree_pages = Delf. total_pages_ Delf. curvent_Page
  Print (f " Free pages remaining: (tree_pages 4")
if tree pages > 0:
  Print (" Potential fragmentation: Free Pages are
         available, but they may 70+ be
        Dufficient for large Yequests")
else:
  Print ( 1 No fragmentation; All memory is
          allocated Without leftover Space)
TOTAL_ PAGES = 256
PAGE_ SIZE. = 16.
Dersion - requests = of
  "Game Session 1": 32,
  "Game Session 2": 20,
```

" Game Sersion 3": 40, "Game Version 4": 18 "Game Session 5": 25 Derver = Gaming Server (TOTAL-PAGES, PAGE\_SIZE) for Dession, pages in Dession - requests. items (): Derver. allocate\_ memory (Dession, Pages) Derver memory \_ utilization () Derver, fragmentation - analysis () OUTPUT: -The program is executed Successfully Med Care Hospital: -AIM: - Write a program for Paging Memory Management for MeolGare Hospital.

```
PROGRAM : -
```

Class Hospital Server;

def - init - ( Delf, total - pages, page - Dize):

Delf. total \_ pages = total \_ pages

Belf. page - Dize = Page - Bize

Delf. memory = [None] \* total \_ pages

Del. = wwent \_ page = 0

Delf. departments = 13

def allocate - memory ( Delp, department\_ name,

requested - pages):

if Delf. - page + Yearnested - pages >

Self. total- pages ....

Pount (f" Not enough memory for { department\_ name j. Allocation folded ")

return Faloe

allocated - pages = list ( range ( Self. Swint-page, Delf. current - page + veguested - pages))

ROLL NUMBER :

Delf. departments [ department\_ name ] = allocated\_ Pages def memory - utilization (self): ülilized - pages = Sum (1 for page in Belf. memory if Page is not None) total - data = utilized - Pages \* Delf. Page - Size Print († " Total memory utilized: ( utilized\_ Pages } Pages, of total-datay units of Patient Jala") det fragmentation\_ analysis (self): free - pages = self. total - pages - self. Eurovent -Print (f" Free pages remaining; (free pages ?") if free - pages > 0: Print Strontial fragmentation: Free Pageo are available, but they may not be Dufficient foi large Vequests") else; Print ("No Fragmentation: ")

department - Leavusts = 1 " Emergency Dept 1: 50, "Radiology Dept": 35, " Laboratory Dept": 45, " Potient Record System": 60, "Swugical Dept": 40, OUTPUT: - The Program has been executed Successfully. Shop Mart : -AIM: - Woute as program for Paging Memory Management for Shop Mart. PROGRAM: class ShopMart Server: det \_ init \_ (Delf, total\_ pages, page\_ Size). Delf. total \_ pages = total - pages

Delf. Page - Oize = Page - Oize Delf. memory = [None] \* total\_ pages Delf. Sworent - pages = 0 Delf. components = 63 allocated\_ pages = list ( range ( Delf. Guvient\_ page , Delf. suvient - pages + requested - pages)) Delf. components [component\_name] = allocated\_Pages def memory - utilization ( Self): utilized\_ pages = Sum (1 for page in Self. memory if page is not None) total - dota = utilized - Pages \* Oly. page - Size Print (f" Total memory utilized: {utilized - pages } pages, I total - data ] units of Product data") det fragmentation\_anolyris (Delf): free\_pages = Delf. total - pages - Delf. current - page Print (f" Free pages remaining: (free - pages 3")

```
if free_pages > 0;
   Print (" Potential Fragmentation: Free pages one
           available, but they may not be Difficient
           for large requests")
  else;
    print (" No tragmentation; All memory has been
            allocated ")
component - requests = {
  "Product Catalog Management": 80
 "Order processing System": 120
 "Customer Database": 150,
Derver = Shop Mart Server (TOTAL_PAGES, PAGE_SIZE)
Derver. Memory _ Wilization ()
Derver. fragmentation- analysis (=)
 OUTPUT: -
The Program has been executed Successfully
```