IARE

**IARE**
**INSTITUTE OF AERONAUTICAL ENGINEERING**
(An Autonomous Institute affiliated to JNTUH, Hyderabad)
Dundigal, Hyderabad - 500 043

# LABORATORY WORK BOOK

Name of the Student : RACHERLA SANTHOSH

| Roll Number | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 3 | 9 | 5 | 1 | A | 1 | 2 | C | 3 |

Class : IT-B Semester : 03

Course Code : ACSD10 Course Name : OS Laboratory

Name of the Course Faculty : Mr. N. Raghava Rao Faculty ID : IARE10924

Exercise Number : 06 Week Number : 06 Date : 25/10/2024

| S. No. | Exercise Number | EXERCISE NAME | MARKS AWARDED | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | Aim/ Preparation | Algorithm / Procedure / Performance in the Lab | Source Code / Calculations and Graphs | Program Execution / Results and Error Analysis | Viva - Voce | Total |
| | | | 4 | 4 | 4 | 4 | 4 | 20 |
| 1 | 6.1 | CloudTech | 4 | 4 | 4 | 4 | 4 | 20 |
| 2 | 6.2 | EduTech University | | | | | | |
| 3 | 6.3 | GameTech Studios | | | | | | |
| 4 | 6.4 | Medcare Hospital | | | | | | |
| 5 | 6.5 | ShopMart | | | | | | |
| 6 | | | | | | | | |
| 7 | | | | | | | | |
| 8 | | | | | | | | |
| 9 | | | | | | | | |
| 10 | | | | | | | | |
| 11 | | | | | | | | |
| 12 | | | | | | | | |

Santhosh

Signature of the Student

N.R.R

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 CloudServer:
    def __init__(self, total_page, page_size):
        self.total_pages = total_pages
        self.pages_sizes = page_size
        self.memory = [None] * total_pages
        self.current_page = 0
        self.processes

def allocate_memory(self, process_name,
                    requested_pages):
    if self.current_page + requested_pages >
                    self.total_pages:
        Print(f"Not enough memory for {process_name}")
        return False
```

```
        allocated_pages = list(range(self.current_page,
            self.current_page + requested_pages))
        self.processes[process_name] = allocated_pages
        for page in allocated_pages:
            self.memory[page] = process_name
        self.current_page += requested_pages
        print(f"{process_name} allocated pages:
            {allocated_pages}")
        return True
    def memory_utilization(self):
        utilized_pages = sum(1 for pages in self.
            memory if page is not None)
        total_data = utilized_pages * self.page_size
        print(f"Total memory utilized: {utilized_pages}
            Pages, {total_data} units of data")
    def fragmentation_analysis(self):
        free_pages = self.total_pages - self.current_page
        print(f"Free pages remaining: {free_pages}")
```

```
        if free_pages > 0:
            print ("potential fragmentation : There are
            free pages left but processes cannot be
            allocated more than this number")
        else:
            print ("No fragmentation : All memory is
                    allocated")
TOTAL_PAGES = 100
PAGE_SIZE = 4
Process_requests = {
    "Process A" : 25,
    "Process B" : 15,
    "Process C" : 30,
}
Server = CloudServer (TOTAL_PAGES, PAGE_SIZE)
for Process, Pages in Process_requests.items():
    Server.allocate_memory (Process, Pages).
```

OUTPUT :- The Program has been executed Successfully.

6.2 Edu Tech University :-

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

PROGRAM :-

```
class UniversityServer:
    def __init__(self, total_Pages, Page_Size):
        self.total_Pages = total_Pages
        self.Page_Size = Page_Size
        self.memory = [None]* total_Pages
        self.current_Page = 0
        self.Systems = {}
    def allocate_memory(self, System_name,
                        requested_Pages):
        if self.current_Page + requested_pages >
                        self.total_Pages:
            Print(f"Not enough memory for
```

```
{system_name}. Allocation failed")
return False
allocated_pages = list(range(self.current_page,
    self.current_page + requested_pages))
self.systems[system_name] = allocated_pages
for page in allocated_pages:
    self.memory[page] = system_name
self.current_page += requested_pages
Print(f"{system_name} allocated pages:
        {allocated_pages}")
return True
def fragmentation_analysis(self):
    free_pages = self.total_pages - self.current_
                                    page
    Print(f" Free pages remaining: {free_pages}")
    if free_pages > 0:
```

```
Print ("Potential fragmentation : There are free
        Pages left, but they may be too few
        to fulfill large requests")
else :
    Print ("No fragmentation : All memory
            has been allocated")
TOTAL_PAGES = 200
PAGE_SIZE = 8
System_requests := {
    "Student Records System" : 40,
    "Faculty Management System" : 25,
    "Library Information System" : 30,
    "Online Learning Platform" : 35,
    "Research Database" : 50
}
Server = University.Server (TOTAL_PAGES,
                            PAGE_SIZE)
```

```
for System, Pages in System_Requests.items():
    Server.allocate_memory(System, Pages)
Server.memory_utilization()
Server.fragmentation_analysis()
```

OUTPUT :-

The OUTPUT has been Executed Successfully.

6.3 Game Tech Studios :-

AIM :- Write a Program for Paging Memory Management for Game Tech Studios.

PROGRAM :-

```
class GamingServer:
    def __init__(self, total_Pages, Page_Size):
        self.total_pages = total_Pages
        self.page_size = Page_Size
        self.memory = [None] * total_Pages
        self.current_Page = 0
```

```
        self.sessions = {}

    def allocate_memory(self, session_name,
                        requested_pages):
        if self.current_page + requested_pages >
                        self.total_pages:
            Print(f"Not enough memory for
                {session_name}. Allocation failed")
            return False
        allocated_pages = list(range(.self.current_
                Page, self.current_page +
                requested_pages))
        self.sessions[session_name] = allocated_
                                        Pages

    def memory_utilization(self):
        utilized_pages = sum(1 for Pages in self.
                memory if Page is not None)
        total_data = utilized_Pages * self.page_
                                        size
```

```
        Print(f" Total memory utilized: {utilized_pages}
              Pages, {total_data} units of game data.")

    def fragmentation_analysis(self):
        free_pages = self.total_pages - self.current_page
        Print(f" Free pages remaining: {free_pages}.")
    if free_pages > 0:
        Print(" Potential fragmentation: Free pages are
              available, but they may not be
              sufficient for large requests")
    else:
        Print(" No fragmentation: All memory is
              allocated without leftover space")

TOTAL_PAGES = 256
PAGE_SIZE = 16
Session_requests = {
    "Game Session 1": 32,
    "Game Session 2": 20,
```

```
    "Game Session 3" : 40,
    "Game Session 4" : 18,
    "Game Session 5" : 25
}

Server = GamingServer(TOTAL_PAGES, PAGE_SIZE)
for Session, Pages in Session_requests.items():
    Server.allocate_memory(Session, Pages)
Server.memory_utilization()
Server.fragmentation_analysis()
```

OUTPUT :-

The Program is executed Successfully.

6.4 MedCare Hospital :-

AIM :- Write a Program for Paging Memory Management for MedCare Hospital.

PROGRAM :-

```
class HospitalServer:
    def __init__(self, total_pages, page_size):
        self.total_pages = total_pages
        self.page_size = page_size
        self.memory = [None] * total_pages
        self.current_page = 0
        self.departments = {}
    def allocate_memory(self, department_name,
                        requested_pages):
        if self.current_page + requested_pages >
                                    self.total_pages:
            Print(f"Not enough memory for {department_
                name}. Allocation failed")
            return False
        allocated_pages = list(range(self.current_page,
            self.current_page + requested_pages))
```

```
        self.departments[department_name] = allocated_
                                            pages
    def memory_utilization(self):
        utilized_pages = sum(1 for page in self.memory
                             if page is not None)
        total_data = utilized_pages * self.page_size
        Print(f"Total memory utilized: {utilized_pages}
              pages, {total_data} units of patient data")
    def fragmentation_analysis(self):
        free_pages = self.total_pages - self.current_
                                        page
        Print(f"Free pages remaining: {free_pages}")
        if free_pages > 0:
            Print("Potential fragmentation: Free
                  pages are available, but they may not be
                  sufficient for large requests")
        else:
            Print("No Fragmentation:")
```

```
department_requests = {
    "Emergency Dept" : 50,
    "Radiology Dept" : 35,
    "Laboratory Dept" : 45,
    "Patient Record System" : 60,
    "Surgical Dept" : 40,
}
```

OUTPUT :- The Program has been executed Successfully.

6.5 Shop Mart :-

AIM :- Write a Program for Paging Memory Management for Shop Mart.

PROGRAM :-

```
class ShopMartServer:
    def __init__(self, total_pages, page_size):
        self.total_pages = total_pages
```

```
self.page_size = page_size
self.memory = [None] * total_pages
self.current_pages = 0
self.components = {}
allocated_pages = list(range(self.current_page,
self.current_page + requested_pages))
self.components[component_name] = allocated_pages
def memory_utilization(self):
utilized_pages = sum(1 for page in self.memory
                     if page is not None)
total_data = utilized_pages * self.page_size
print(f"Total memory utilized: {utilized_pages}
Pages, {total_data} units of Product data")
def fragmentation_analysis(self):
free_pages = self.total_pages - self.
                current_page
Print(f"Free pages remaining: {free_pages}")
```

```
if free_pages > 0 :
    print ( " Potential Fragmentation : Free pages are
            available, but they may not be Sufficient
            for large requests " )
else :
    print ( " No fragmentation : All memory has been
            allocated " )
component_requests = {
" Product Catalog Management " : 80,
" Order processing System " : 120,
" Customer Database " : 150,
}
Server = ShopMartServer ( TOTAL_PAGES, PAGE_SIZE )
Server . memory_utilization ( )
Server . fragmentation_analysis ( ) .
```

23/10/24

OUTPUT :-

The Program has been executed Successfully.