

LABORATORY WORK BOOK

Name of the Student: KAGHERLA SANTHOSH Class IT-3 Semester 0.3								Roll Number								
		e :ACSD10Course		25 Labor	otorur	2	3	9	5	1	A	1 2	G	3		
		Course Faculty				28)	ŧ,		Fac	ulty	D :	TARE.	109	24		
rvali	ne or the	Course Faculty	nagrandi	u												
Exe	rcise Nur	mber: 07-	Week	Number:,.	MARK	(C A)	MAR	DEI	Da		~-1	•••				
S. No.	Exercise Number	EXERCISE NAME		Algorithm	Source Code Calculations and Graphs						Viva - Voce	T				
			Aim/ Preparation	Performance			15					1	Total			
			4	4							4	1	20			
1	7.1	Urban as - RAG	4	Naph	yita	ille	4	jt.	OKS	L	f	4	2	0		
2	7.2	Urban OS - WFG	men I		- Ma	0	(fin			100				
3	7.3	The Library Conference									10a	0				
4	7.4	The Dining Philosopher	tal)	tolet	dofan	7		0	D-/	,		1				
5	7.5	Banken's Algorithm	I Was	18	自用。	=	C		NOV		-					
6		way I graph I po					(2)	R	305	71	1 /	60				
7	'yora	K Fele Nota	Memory	, 00	3 1			5		NA.						
8		a spines vol	o nd	nce ;	un.or	17	d	18	Ball	0.00						
9					Nell											
10		1	(33)0		•							No.				
11		Ach	7 33	16. Ku	100	-	3	3	0,0	3)	*					
12				,				1	1111			40				

Signature of the Student

Signature of the Faculty

i allocatid le vecous

- 7. Resource Allocation.
- 7.1 Unban OS Resource Allocation Graph (RAG)

AIM: - Write a program on Urban OS wing

Resource Allocation for Unban Os employs.

PROGRAM: -

class Resource Allocation Graph:

def_init_(odf):

Delf. graph = default dict (list)

Delf. Process = 1'A', 'B', 'c', 'D';

Delf. resources = { 'cpu', 'memory', | File', Network'}

3.1 Moun 00 - 1916

Whom OS - MFG

Delf. allocated = { resource : None for resource in

Delf. resources z

Delf. requests = defaultdict (list)

def request_ resource (Delf, process, resource):

if self, allocated [resource] is None:

```
Self. allocated [verource] = Process
  Print ( + " { Processy allocated { resource } ")
else:
 Delf. graph [Process]. append (Verource)
 Delf. graph [ Verounce]. append (Delf. allocated [resounce])
Pount (f" processy requests (resource z, waiting for it
       to be freed by { Delf. allocaters [verounce] ?")
def release _ resource (delf, process, résource):
 if self. allocated [ resource] = = process:
   Delf, allocated [resource] = None
  Delf. graph [process]. Vernoue (resource)
 Delf. graph [resource]. Vernove (process)
 Print (f" fprocessy released ( resource y")
 else:
                                    Jalun False
 Print (f" & Process y does not hold frevource ?
         cannot release")
        · (CSDINACTOR + 19
```

```
def detect_deadlock (self):
  Visited = Set()
  Stack = Setl)
 def dfs (node):
  if node in Stack:
   return True
if node im vivited:
return False
            resource that I someway _ moder for
 vivited. add (node)
 Stack. add (node)
                  well, allerand ( regress
 for neighbor in Delf. graph [node]:
  if Ifs (neighbor):
              if it i graph [ resource]. Very
    Yeturn True
              house (to the temporal separation
 Stack. remove (node)
 return False
for node in list (Delf. processes) +
            list ( Delf. resources):
```

```
if nade not is visited:
   Print (" Deadlock Setected in RAG")
    vetwon True
  Print ( No deadlock detected ").
  return False
 def Show_ray (self):
                                      : oak
   Print ("Revource Allocation Graphi")
   for node, neighbors in stell. graph. items ():
    Pount (f" { node y -> {', '. join (neighbors) y")
Vag = Resource Allocation Graph ()
Yag. Yequest_ Verounce ('A', 'cpu')
Vag. Verjuest_ Versource ('A', 'Memory')
Vag. Yeavust verouice ('B', 'CPU')
Vag. Veryuest_ Verounce ('B', 'File')
Vag. request_verouice ('c', 'cpu')
```

Yag. Yequest_Yerowice ('c'; 'Memory') Vay. Veryuest_ Verounce ('D', 'Network') rag. request-resource ('D', 1cpu') ray. Show - ray () if vag. détect - deadlocke): Print ("Deadlock resolution needed") : (the) your _ works tob else: Point (" System running Without feedback") ray. Show - ray() Vag. detect _ deadlock(). OUTPUT : -(a) = Resource Allocation Graph () The Program is executed Successfully. og. remark _ resource ('A ! Mousean) (109) 18 ') Buce (B! CPU!) (tops 1 15') sounce d'aupor ages.

7.2

Unban OS - Wait - fox - Graph (WFG)...

AIM: - Write a Program for Unban OS - WFG, Which helps manage dependencies and Prevent Potential deadlock Situations among Concurrent Processes.: [many _pmidal; PROGRAM: -

from collections import default dict

class Wattor Graph:

def _ init _ (self):

Self. graph = default dict (list)

Delf. Processes = (1A1, 1B1, 161, 1013

Delf. resources = { 'cpu', 'Traffic Data', 'Memory',

"Metwork!, 'Emergency Services!

'Database' }

def odd - dependency (self, waiting process, blocking process):

```
Self. graph [ waiting_ process]. append (blocking_ process)
  Point (f" Process & Waiting_ Process y is waiting for
        Process & blocking - Process y")
det remove _ dependency ( Delf, waiting_process;
                      blocking-process):
  if blocking_process in Self. graph [waiting_process];
   Delf. graph [waiting_ process]. remove (blocking_process)
   Pount († " process of waiting _ process of no longer waits
         for process { blocking_process y")
def detect - deadlock ( Delf):
  Visited = Bett ( pal ) to be the pool - ing, . Mose
  Stack = Set(): 81 1/1 = and only a flow.
def dfs (node):
 if node in Stack:
return True
 if node in Visited:
   Veturn False
     blocking = Process):
```

Vivited. add (node) Stack. add (node) Wfg = Wait For Graph () Wfg. add_dependency ('A!, 'B!) Wfg. add - dependency ('B', 'c') Wfg. add - dependency ('c', 'D') - ; MBRIODAD Wfg. add - dependency ('D', 'A') Wfg. Show_ Wfg() delethrope training milester point if Wfg. detect _ deadlock(); Print (" peadlock resolution needed") else: Pount ("System Vanning Without Leadlock"). Wfg. remove - dependency ('D1, 'A') Print ("In After revolving dependency".) Wfg . Dow - Wfg () grant (+ " Recourse (recovered 20 Wfg. detect _ deadlock() (holding team 1 11. OUTPUT: - Executed Successfully

The Library Conference.

AIM: - Write a Program on Determine if there is deadlock in the Dystem, If Do describe the Circular wait condition and Duggest a way to Prevent or resolve the deadlock.

(short) John . Not 12.

(19, 10,) Somepuedop - ppr 65/10,

6/26:

OUTPUT: - texaculad

PROGRAM: -

from collections Import defaultalist

clars Resource Allocation Graph:

def_init_ (self):

Delf. graph = defaultdict (list)

Delf. Yesources = { 1 R11, 1 R21, 1R31, 1 P11, 1 P213.

Belf. teams = { 1 A1, 1B1, 1c13

Wife , remove _ december def add - holding (Welf, holding - team, resource):

Delf. graph [resource]. append (holding_team)

Point (f" Resource (resource y is held by Team { holding team y ")

```
def detect _ deadlock ( Delf):
   Vibited = Detl)
   Stack = Set()
 def dfs (node):
  if node in Stack:
                     ( ) por - works . yes
   Veturn True
 if hode im vibited: : Hallback tales per !
   return False
Vioi ted. add (node)
 Stock. add (node)
 for neighbor in Self. graph [node]:
 if dfs (neighbor):
return True postonilo dos sonos
          Print "3. Exterior de Recarony
Stack. Vernove (node)
return False
Yag = Resource Allocation Graph()
rag. add _holding ('A', 'R1')
                        - : TUQTUO
ray. add - dependency ('A?, 'P11)
```

Vag. add - holding ('B!; !P2!) Vag. add. dependency ('B', 'R2') Oboto - bestical rage add - holding ('c', 1831) det des (node): Vag. add - dependency (101, 1921) vag. Show - Yay () if vag. detect_deadlock(): : bestield me store Print ("In Deadlock detected. A->B->C->B Pount ("Revolution Strategies:") Pount (1. prevention: Request all resources at once to avoid circular nait") Pount (1 2. Avoidance: Use Banker's Algorithm to ensure Date allocation") Print 1 "3. Detection & Recovery: Deadlock Cycles") elve: Point ("System vunning without Leadlock") OUTPUT ! us executed Successfully.

7.4

The Dining Philosopheus.

AIM: - Wints a program to Analyze the System for Potential Jeadlock for the Dining?

Philosophers & What Strategies Lan be used to Prevent or Verolve the Jeadlock.

PROGRAM: - ([2] deneti [5] dianeli [8) serieselili

import threading with the strong in

Import time

class philospher (threading. Thread):

def _ init_ (Delf, id, left_utensil, vight_untensil):

- : TUGTUO

threading. Thread. _ init _ (Delf).

Delf. id = id

Delf. left_utensil = left_utensil

Solf. right_utensil - right_utensil

def vun (self):

while True:

```
Point (f" Philosopher ( delf. id y is thinking")
   time. Sleep (1)
               AIM: - White a Program to An
utensils = [threading. Lock() for _ in Vange (5)]
Philosophers = [
· Philosopher (1, utensils [0], utensils [1])
 Philosopher (2, utinsils [1], utensils [2]),
 Philosopher (3, utensils [2], utensils [3]), -: MARIONI
 Philosopher (4, utensils [3), utensils [4])
 Philosopher (5, utensils [4], utensils [0])
               class philosophen ( threoding. Thread).
for Philosopher. in philosophers:
 Philosopher. (join (). _ + 100 _ . boordt . pribagerdt
                                   De = 61. Hall
OUTPUT : -
            is Executed Successfully
The Program
              light _ their
                                  det van ( Dell):
```

vitile True;

```
7.5
```

Banker's Algorithm.

AIM: - White a program on Banker's Algorithm,

It is a deadlock avoidance algorithm used in

Operating Systems to manage resource allocation &

Prevent deablocks.

a columbia . No

PROGRAM : -

available = [3, 2, 2]

allocation = [

[0,1,0]

[2,0,0],

[3,0,2],

[2, 1, 1]

[0,0,2]

7

maximum = [

(7,5,3),

[3,2,2,],

[9,0,2],

15/16

```
def calculate_ need (maximum, allocation):
 need = ()
                    AIM: - white a purasant of
for in Tange (len (maximum)):
  need. append ([maximum [i][j] - allocation [i][j] for j in
  range (len (maximum [0]))])
return need
Pount ( " Initial State: ")
                                        YROGRAM :-
Point ( " Available: ", Work)
                                 mainte = [3,2,2
Pount (" Need matrix:", need)
if lem ( Date - Deguence) = = len (allocation):
                                         [0.1.0]
 Pount (" System "es în a safe state").
 Pount (" Safe Dequence: ", ["P"+ Dtr (i+1) for in
         Date - Deguence])
 return True
else: print (" System is not sin a state")
      Yetung False
is _ Doje (available, allocation, maximum)
OUTPUT: - Executed Successfully.
```