

LABORATORY WORK BOOK

Name of the Student KAGHERLA SANTHOSH							Roll Number							
		IT-3 Semester.				2	3	9	5 1	A	1 2	G	3	
Nan	ne of the	Course FacultyMYN.	Racho	uio. Ro	0				.Facul	ty ID :.	TARI	10	924	
Exe	cise Nu	mber :10	Week	Number :	10	'C AM	/ARD	F)		11-		1	
S. No.	Exercise Number	EXERCISE NAME	Aim/ Preparation	Algorithm / Procedure		Source Code Calculations and Graphs		F	Program Execution Results and Error Analysis 4		Viva - Voce		Total	
			4					-			4		20	
1	10-1	The Story of Busy Cafe and its Orders	4		4		4		-i	f	3	1	19	
2	10.2	The Tale of the library and its Book Shelver.		12 20	t.00	enp	Ø	100	tol	0>	loa			
3	10.3	The Story of Busy Cafe and ats Special Recipe			, while	6	07		33	Ca	gani			
4	10.4	The Tale of the Art	1000		1 3 5			-	e Par		136			
5	10.5	The Tale of the library and its popular backs.	mode		HEAVES I		d To			1003	5	+		
6		Make capocities	> 1/10	(DIV) 314			3		(n) ,	-	2			
7		Shows) _ XI	3,	A make						10		A	
8			:	(rebra	1000) 1	(00)	1	- 30	lan.	T			
9	44.	the (Fig.) august	ues .	थें- कर्न	mara	(M)	10	0	bb					
10			a Po	A)	Contra	o=	. }		(a)	ral	1.			
11		· Bringer	1				urc		i bay	ente)				
12		(7+6)000	greb til	00 14										

Santhosh.

Signature of the Student

Signature of the Faculty

Spales - bound of i was

10. Page Replacement Algorithms.

10.1 The Story of a Busy Cafe & the Orders.

AIM: - Write a program for Gafé FIFO to Manage their incoming orders, ensuring that the first order received no also the first one to be processed.

PROGRAM : -

from collections import deque

class Cafe FIFO:

def _ 9nit _ (Delf, max - capacity = 4):

stelf. counter = degre (maxlen = max_capacity)

Delf. max _ capacity = max _ capacity

def adal - order (Delf, order):

"" Add a new order to the country (FIFO greene)""

if len (self. counter) == self max_capacity:

removed_order = Self. sounter. popleft()

Print (f" Counter full. Processing & removing Oldest order: {removed_order 2 11)

```
Delf. counter. append (order)
 Point (f" Added order: forder y")
def display_orders (Delf):
 """ Display courient orders on the counter""
 if Delf. counter:
  Print (" Courent orders on the counter; ", ", ". Join
          ( Delf. counter))
 else:
  Print ("No orders on the counter")
def Process - remaining - orders (Delf):
 "" Process remaining orders in the queue (FIFO)"""
 while Delf. = junter:
   order = Dely. counter. popleft()
  Prant (f " Processing order: (order 3")
def Dimulate_ cafe():
  cafe = Cafe FIFO (max - capacity = 4)
  orders = ['Coffee", "Tea", " Muffin", "Crownant",
            " Smoothie"]
```

for order in orders:

= ofe. add - order (order)

cafe. display-orders()

Pount ("In End of the day - Processing remaining orders:")

afe. Process _ remaining _ orders()

Pount ("In Final orders on the counter (Whould be empty):")

cafe. Lisplay - orders ()

if __ name __ == " __ main __":

Dimulate_ cafe()

def Fracess - remaining orders (self) The program is Executed Successfully. Process remumma

10.2

The tale of the Library and Its Book Shelves.

Joseph and Joseph AIM: - Write a Program for the library of Timelers Tales uses the FIFO Policy to manage the books on the shelf. S aradeling 1

```
PROGRAM :-
from collections import deque
                     and on andro on " the
class Cafe FIFO:
 def - init - ( ody, max - capacity = 4):
  Delf. counter = deque (maxlen = max - capacity)
  Delf. max - capacity = max - capacity
def add - order ( Delf, order):
 """ Add a new order to the counter (FIFO queue)".""
 if len ( Delf. counter) == Delf. max - capacity:
  removed - order = Delf. counter. popleft()
 Print (f" Gounter full. Processing & removing aldest
   Order: ( removed - order 3 ")
Delf. counter, append (order)
Print (f" Added order; horder j")
def Lisplay-orders (Delf):
  Dusplay current orders on the counter.""
 if Delf. counter:
                           duminates, " wasted.
   Point (" Guerent Orders on the counter: ", ", ".
```

```
KOORON!
   ion ( self. counter))
                     waste tragan controller must
 else:
  Print (" No orders on the counter")
det Process - remaining - orders (Delf):
 "" Process remaining orders in the queue (FIFO) "" "
While Delf. counter:
                        = stronges - xago , low
  ander = Delf. counter. popleft()
 Print (f" processing order: 4 order 3")
def Simulate-cafe():
                        == Cretmos . Pace 1 m
 cafe = Gafe FIFO (max_ capacity = 4)
 ordero = [ "Goffee", "Tea", "Muffin", "Croissant",
 "Smoothie"]
for order in orders:
  cafe. add_order (order)
 cafe. display-orders()
Pount ("In End of the day - Processing Vernaining order:")
cafe. Process_ remaining_ orders()
```

Point ("In Final orders on the counter (Double be emply):") cate. display_orders() it _ name _ = = " _ main _": Dimulate_ cafe() Оитрит :-The program is Executed Successfully. The Story of the Busy Cafe & Its Special 10.3 Recipe Book grant (+ " counter full . Remove AIM: - Write a Program for Cafe Nortalgia, It uses the LRU Policy to manage her recipe book on the counter. PROGRAM :from collections import Ordered Dict clars Recipe Book: def _ init _ (Delf, max_capacity = 4): Delf. counter = Ordered Dict()

```
def access - Yecipe ( Delf, recipe):
   "" Simulate accersing a recipe."""
   if recipe in Delf. counter:
    Delf. counter. move _ to _ end (recipe)
    Print (f" Recipe ' (recipe y 1 is already on the counter
          Moved to the most recent")
  if len ( Delf. counter) = = Delf. max_ copacity:
  removed _ recipe = Delf. counter. popitem (last = False)
   Print (f " Gounter full. Removed least recently used
         recipe: 1 1 removed - recipe [0] ? "")
Delf. counter [recipe] = None
Pount (f " Added Yeaps: 1 {Yeape 3 "1)
def durplay - counter (Dely):
 "" " Display the surrent veriges on the counter
     in the order they are accessed. """
 Prient (" Current Yeupes on the counter: "
        list ( self. counter. Keys ()))
```

def simulate _ cofe(): Yecye - book = Recipe Book (max - capping = 4) recipeo = [" Pumpkin Spice Latte" Apple pie", " Bluebevry Muffen", Ginnam on Roll'1 "Apple Pien, " Maple Pancakes", Pumpkin Spice Latte" def_ nit_ (600), capoa for recipe in recipeo; Veripe - book. overs - Veripe (Veripe) Verige - book. display - counter() Print ("In Final Vecipes on the counter") Veripe - book. Lisplay_ counter () if __ name _ = = " __ main _ ":

Simulate_cafe().

OUTPUT :-

10.4

recyc - book in People Book (max _ supper The program is Executed Successfully.

The Tale of the Art Gallery and Its Exhibition.

: Object Thomas 7:6

Maple Tancaken

Blueboong Matter AIM: - Write a program for Gallery of Modern Arts to manage the display using the LRU policy.

PROGRAM :-

from collections import deque Pumphin Spice Latte

class Gallery Display:

def_ init_ (Delf, capacity):

Delf. capacity = capacity

Delf. Lisplay - board = Lequels

def access - witwork (Dely, witwork):

if witwork in Delf. Lusplay _ board:

Delf. dusplay_ board. Vernove Cartwork)

```
elif len ( welf. display_ board) = - self. capacity:
   Delf. display _ board. popleft()
Delf. display_ board. append (artwork)
def ewvent - display ( self):
  return list ( Delf. display _ board)
gallery = Gallery Display (capacity = 6)
artwork_requests = [
"Sumpet Over the Lake",
 " Abstract Dreams",
 " C'ityscope at Dusk"
                                     - ! MARROORY
" Golden Fields",
                            and collections import
 "Havemony in Blue",
                                class LFV bache:
 for artwork in artwork - Veguests:
 gallery. access_ autwork (artwork)
Pount (" Final State of the Lisplay board:")
```

for artwork in gallery. < werent _ display(): Print (artwork)

OUTPUT: -

10.5

Growtha displayed board, opposit (asturone The program is Executed Successfully.

The Pale of the library and Its Popular Books. = 40 = outrogos) &

advice the stayloots = AIM: - Write a Program for the library Manager of the Book Haven to manage the Display wing LFU policy. " Hue I to apo July "

PROGRAM :-

from collections import defaultdict, Ordered Dict clars LFU Gache:

def_ init _ (Self, capacity):

Delf. capacity = capacity

Delf. freq _ map = default dict (Ordered Dict)

to almo so

Belf. book - frear = ly

Delf. Min - freq = 0 12/16

```
def access - book (sself, book):
 if book in Delf. book-freq.;
  Delf. - update - trequency (book)
else:
                   down that they sout fi
 if len ( Delf. book. frea) == Delf. capacity:
   Delf. _ evict _ book()
Delf. frey_map (1) [book] = None
Delf. book - frea [ book ] = 1
Delf. min - freq = 1
def - update - frequency ( Delf, book):
 freq = Delf. book - freq [book]
 del Delf. frea _ map [ frea] [ BOOK]
 if not self. freq _ map [freq]:
  del Delf. frey - mays [ frey.]
 if frew == Selfo min - frey:
   Self. min - frey += 1
Delf. frey_ map [frey+1] [book] = None
```

Delf. book_freq[book] += 1 def - evict - book (Delf): book, = Self. freq map if not self. frear_map [self. min_freq]: del Delf. frew_map [Delf. min_frew] del Delf. book - freq [book] def current_display (Delf). display = [] Delle min - frey = 1 for frey in Dorted (Delf. freq_map. Keys() Yeverse = True): display. extend (list (delf. frea - map ! freat. 1000 1 (100 Keys ()))) return display library_ display = LFU Gache (capacity=5) book - checkouts = [K- checkouts = L (4 The Great Gatsby", 3), (11 1989 12,5) 34/16

(" To kill a Mockingbird", 2), (" Pride and prejudico", 47, ("The Catcher in the Rye", 1), (Moby Dick', 6), (" The Odyssey", 3), (" War and Peace", 2). for book, frequency in book_ checkouts; for _ in vange (freuluncy): library - Lisplay. access - book (book) Pount (" Final state of the Lisplay board:") for book in Sibrary_ Lisplay, covered Lisplay (): Pount (book). OUTPUT : ~ Executed - Successfully. The Program