			No:	
	1N2110 - October 2019			(1)
	fruit pol xital	1000	Pol .	ent r
	Question of 1	[2]	01	oile ha
1	[4] 8 [3]			
(c)	The for loop sterates over the	'my Array	from 0	+0
11	Size-1. Which is the n,	the size of	orray .	1
	Therefore the method find20	has a t	me comple	xity
1		1.51.		
	1 1014 P (8)		. 3	
	Question 02	AM	Alm I Fa	V.
112			For fire	
(a)	-> Elements of an array can be	randomly	accessed i	1 q
	constant (OCI)) time.		1 2	1
Jak.				
6	-> An element can be unsereted	1 to the ex	nd of an	array in
1	a constant time (Oci)).			
,			4	1.3
	-> Arrays are simple to under	stand and	implement	
	[31] 01 [0] <-			I willed
-	-> Arrays has good cache local	ity due th	neir contagiou	15 memory
	allocation. I to Let	1181	07 [3	1 1/
	11011 10 (8)	- (c)	PPP LE	1
	-> Arrays are more memory eff	ectent compa	ared to 191	nked lists.
	160 01 181		1.01 11	
	1-101 3 111	Cal.	17 17	
G 1	ant Search (node + thead; ant	value	I F	
(b)	node * current = * head			
	While (current - Bandana )=	NUL)		
	if (current -> duta =	= Victure		
		Value )		
	return 1	1		
	current = current -> ne	7XT		
	return 0			

SAMMANA RULE PAPER (SINGLE RULE) - SIZE A4

(c)					91	25 nd	11.0	GHSMI
	Index	key	Next	No.	Index	key	Next	
head ->	[0]	10	[6]	head>	[0]	10	[6]	ie And De I grade
	(1)	N/A	N/A		[6]	. 8	[5]	
	[2]	87	[8]	Hipm All	[5]	10	[2]	II IT carrie
A A	[3]	9900	NULL	with the	[2]	87	[8]	1 1 1 1 1 1
11 1 1	[4]	331, 10	[3]	lec Int	[8]	90	[4]	1000 1001
	[5]	10	[2]		[4]	31	[3]	the second
	[6]	8	[5]	A .	[3]	9	NULL	41.4
	[7]	N/A	N/A				1 60	geroff for
( )	[8]	90	[4]					
	" or bas		plination	iui od n	65 (ii)	10 00	to ekna	multi att log!
(i)	[3]				· 3/N	11	00) 1-	Const.
				1				
(11)	[5]	1 to 12	4- 9-11	of both	serit s	d 200	kaarmat a	, 1 E
					((no)	) sout	1,	11 11 11
(iii )		1				,		
	index	key.	Next	ndersland	Index	key	Next	and the
head>	[0]	10	[6]	head ->	[o]	10	[6]	
, prints o	[6]	8.	[5]	p like; of		NIA	N/A	41
	[5]	10	[7]		[2]	87	[8]	Property I I
	[7]	999	[2]	-	[3]	9	NULL	
1 = 1 , 4	[2]	87	[8]	Frank 199	[4]	1 31mg	[3]	Con Visit
. 9-1	[8]	90	[4]		[5]	10	[F]	
	[4]	31	[3]		[6]	8	[5]	
	[3]	9	NULL ,	og (. c. c )	[7]	999	[2]	1
				)	[8]	90	[4]	) . h
				u 11 21 4				
			( 71				1 4	

Overstron 03  TRUE  Using a Stack is a one of the easiest ways to print sorted list in a reverse order. We just have to push the names in order to a stack and pop one one. Then we can simply print the ; each poped elemented using a question of process naturally follows FIFO principle.  Or = []	No:	
TRUE  Using a Stack is a one of the easiest ways to pint sorted list in a reverse order. We just have to push the names in order to a stack and pop one one. Then we can simply print the ; each poped elemented.  FALSE  Breadth-First Search should be implemented using a quality follows FIFO principle.  Or = []  Or = []		
TRUE  Using a Stack is a one of the easiest ways to pint sorted list in a reverse order. We just have to push the names in order to a stack and pop one one. Then we can simply print the ; each poped elemented.  FALSE  Breadth-First Search should be implemented using a quality follows FIFO principle.  Or = []  Or = []  Or = [5, 5, 7, 7, 12, 12, 4, 4, 0, 0, 4, 4, 6, 6]	(passing) man 3 1000 1 1 . 1.	Question 03
TRUE  Using a Stack 15 a one of the easiest ways to print Sorted list in a reverse order. We just have to push the names in order to a stack and pop one one. Then we can simply print the peoch poped elemented.  FALSE  Breadth-First Search should be implemented using a quality process naturally follows FIFO principle.  Or = []  Or = []		
Using a Stack 15 a one of the easiest ways to pint Sorted 175t in a reverse order. We just have to push the names in order to a stack and popone one. Then we can simply print the peach poped elemented.  FALSE  Breadth-First Search should be implemented using a question process naturally follows FIFO principle.  Or = []  Or = []		
push the names in order to a stack and pop one one. Then we can simply print the peach poped elem FALSE  Breadth-First Search should be implemented using a quality follows FIFO principle.  Or = []  Or = []	( ) Jan 19 John Je	TRUE
push the names in order to a stack and pop one one. Then we can simply print the peach poped elem FALSE  Breadth-First Search should be implemented using a quality follows FIFO principle.  Or = []  Or = []	one of the easiest ways to	Using a Stack is a on
push the names in order to a stack and pop one one. Then we can simply print the ; each poped elem FALSE  Breadth-First Search should be implemented using a quality follows FIFO principle.  Or = []  Or = []  Or = []	a reverse order. We just have to	print Sorted 175t in a
FALSE  Breadth-First Search should be implemented using a que  BFS process naturally follows FIFO principle.  Or = [ ]  Or = [ ]	rder to a stack and propone hu	push the names in order
FALSE  Breadth-First Search should be implemented using a que  BFS process naturally follows FIFO principle.  Or = []  Or = []	noly print the 1 each poord plement	one. Then we can symply
Breadth-First Search should be implemented using a quebra process naturally follows FIFO principle.  Or = []  Or = [5, 5, 7, 7, 12, 12, 4, 4, 0, 0, 4, 4, 6, 6]		
$Q_1 = [$ ] $Q_2 = [$ ] $Q_3 = [$ 5, 5, 7, 7, 12, 12, 4, 4, 0, 0, 4, 4, 6, 6]		FALSE
$Q_1 = [5, 5, 7, 7, 12, 12, 4, 4, 0, 0, 4, 4, 6, 6]$	Should be smolemented using a succession	Breadth- First Search sh
0, = [5, 5, 7, 7, 12, 12, 4, 4, 0, 0, 4, 4, 6, 6]	ollows FIFO principle.	BFS process naturally follo
The state of the s		Q1 = []
The state of the s		(194)
The state of the s	4, 4, 0, 0, 4, 4, 6, 6]	0, = [5, 5, 7, 7, 12, 12, 4,
Void StockToQueue ()	at withing I am the great the	
Void StuckToQueue ()		
	King a series of the series of	Void StuckToQueue ()

(b) Q1 = []

(c)

(a)

(i) TRUE

(ii) FALSE

StockToQueue () No:9 (1

While (! 5 empty ()) }

9. enQueue

Stack To Quare (Stack 5, Queue q) } while (! s. empty()){ q.enqueue(5.pop());

SAMMANA RULE PAPER GROOLE RULE) - 5726 A4

	, Date:
***************************************	
Cii	int sizeofayene (Queue q) {
	while (! q. empty) {
	Count++;
1	q. daqueue();
1 6	at report terent int. to man or trole a mining
	return Counting and the land
put so	of gog bon doubt a at what he come and page
· lana	In Pagor whom & all know place are not be fell -
	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
. gurug	
	a press behaviore and blante design design to the
	see frances audienty follows of the prince seems
pi.	
	(\$ - 1 e, e, a, a, a), a), 4, 4, 0, 0, 4, 4, 6, 6]
	- Decomposition of the first pháp (in a sail sin a sa
	A falling at y show of the
	Shapping of the state of the st
	f(p man), 2 half) man Delder 1 (192)
	Al Milyand I William I was
	the state of the page of the p
7	
	SAMMANA RULE PAPER (SINGLE RULE) - SIZE A4

No:	

	IN2110 - January (2019) 10 ( hard short is) 11 11
	Short = farmest show
	Question 01
	discount of the second
(a)	Number of Comparisons - 21
	Number of Swaps - 11 1 males
	1 convertains of faire
(b)	Selection sort
	In selection process sort, sorting process starts from
	the begining of the array. In the intral pass the
	Emallest element in the array would come to the index
	O. In the Second pass the 2nd smallest element would
	come to the index 1. This process will go on for the
	other passes too.
	Therefore even in the part way through in a selection
<u> </u>	sort, we can see the a soited away at the begining
	of the array.
(c)	-> When input Size of the array is small.
	-> When the array is partially soited.
	Question 02
	a describe le visas bound as
(a)	Big O notation 15 used to describe the upper bound or
	worst-case time complexity of an algorithm. It represents
	the growth rate of the alogorithm's running time as the
	User input size increases.
	O(n+) means algorithm's running time, in the worst case,
	15 propotional to square of the input size.
	(Algorithm's time complexity quadratically increases with the
	( Algoriam 5 time complexity

input 5:2e)

No;	

(P)					2. )
(1)	mult (5, 7)			3	(1)
	mult (4,7)				
	mull (3,7)			01	(in)
	mult (2,7)		7		
	mult (1,7)				( La
		1. 12 4	1 14	67 5 A	
(11)	2000 times	[4]	OF	. Cok	
		ni ni ni		111	
coti	) int F(int n) {	[8]	, ka	801	
	1f (n==0)	100A	18	[2.]	
	retuin 1"	[8]	-,8	6-1	
	1f (n==1)	La LEI	0.1	[2]	
	return 1;	[4]	3	fir	
	return F(n-1) + F(n-2);	[8]	PIE	1 P2	
		[ [ pi	0P /	181	
	the state of the s				
			Ĺ		
(X)	Ouestion 03		AQ A	de co	
1					
(0)	-> Dynamic size			1 1	
	-> Roduced memomry waste since	memory 1	s allocate	ed dyna	imically.
	-> No need for contagious men	nory.		,	
		<u> </u>		7,4219	
(b)	void insertAtMiddle (struct node *+ 1				\ ,
		-	1/	t pos	) {
	forbintis	Λ			
	Struct node + Prev = * head-r			0.00	-
	for (int 1=0; 12 pos-2; 1++)	) {	b v -		
	pieu = preu > nexti		11 - 1-1-	1 80 1	
	new-node -> next = pieu -> r	next;	CANID	MANA RULE PAPER (SING	SERUHE) - SIZE AA
	Prev_node -> next = new-noo	96;	JAMI	THE PACE (SING	July Market

				Date:
( c))				
(i)	3	*		( T. 21 from 11)
				(FRA) Hon
(ii)	10			(R. F.) Hour
. /		F		(F, C) Hum
(iii)	1231			(Fig. 1) man
	Index	key	Next	
	[0]	10	[6]	and the state of t
	[1]	N/A	NIA	
	[2]	97	[8]	A CASTA DA DA JOSE, NA
1	[3]	9	NULL	and the second of the second o
	[4]	31	[3]	A Company of the Company
	[5]	10	[7]	en in the Contract of
	[6]	8	[5]	in the state of th
	[7]	999	[2]	" : (sen) 198 (an) to missing
	[8]	90	[4]	
		4		
	Questro	n 04		in the second of
(0)				Dynamic Size
(1)	C , e	2, d, k		mind about pressure friends
		,	, p. 10	ta na promotoure el lessa " a
(:i)	push (			
Teal and				of Tolon I will bloke was a long to
1.1		$\rightarrow$ k		
	push (c			
		push(d)		and the second of the second
	pop()			per song si e a a ing sh
	pop(x)			James and the second
		→ a		
	push(e)		, 1 ·	SAMMANA RULE PAPER ISINGLE RULE) - SIZE A4
	ρορ() -	-> e		SAMMANA RULE PAPER (SINGLE RULE) - SIZE 44

POR CONTRACTOR OF	
(p)	
(1)	53. 52. Push (S1. pop()) → A +0 S2
	52. push (51. pop()) -> B to 52
	53. push (51. pop())> c
	53. push (51. pop()) -> D
	53. push (62. pop(1) -> B
	53. push (son pope) - A
(;)	52. Push(S1. Pop()) - A to S2
	52. push (51. pop()) → B 10 52
	53. push (51. pop()) → c 100 '
	51. push (52. popc)) → B +0 51
	53. push(52. popc)) -> A
	50. push(51.pop()) → B to 52
	53. push (51. pop()) → D
+	53 push (62. pop()) -> B
,	