use this structure definition: Linked List Insertion and Beletion typedef struct node { INSERT FIRST ideally it's better to do matter first INSERT LAST void insertfirst CLIST * A, charc) { void invertlast (LIST * A, char c) { 11 temp is a PN, used for insertion, tray is a PPN LIST temp = (LIST) malloc (size of (struct node)); 0 11 temp is a pointer to a node, used for insertion ino need (1) used for traversing loccessing traversing LIST temp, * trav; temp-> elem = c; // filling in the node (2) il loop for accessing temp -> link = *A; il connecting the node to its next 3 for (trav = A ; * trav != NULL ; trav = 2 (* trav > link) {} * A = temp; // placing the node to first. (4) (3) 11 insertion process (31) temp = (LIST) malloc (size of (struct node)); (3,2) temp > elem = c; //filling in the node -Execution Stack STACK HEAP (33) temp-> link = NULL; Issince last na man parameters *trav . = temp; A, R, OF AC HEAP **temp** STACK elem incert First() link (3,4) C 9 10 G temp elem (3) C 1. L= * A (4) elem link elem 0 (312) (3,3) link AROF main() B • A • Parameter elem 1 xx trav elem link PK 1 A (Oh bew, sab you forgot to ask if dynamic allocation is successful by asking if temp!=NULL) & INSERT SORTED 1 STACK 1 HEAP void insert Sorted (LIST * A, char c) { Il temp is a PN used for insertion, Travis a PPN used for trav temp 0 link 0 2/10 traversing and accessing C LIST temp, * trav ; 3 7 B 33 11 do dynamic allocation Parameters temp = (List) mallor (Sizeof (struct node)); -> il checking if dynamic allocation is successfull link elem link elem If (temp!=NULL) {

Moop for accessing 2 additional condition for surfing

(311)

313

temp + elem = c;

temp -> link = trav;

* trav = temp;

for (trav = A; trav !=NULL 21 (*trav)-> elem <= (; trav=2 (*trav)>nex){}

if unique it's just < only

cuzif <=, you are allowing

0

C

A

[SPECIAL SECTION] Insert Unique with Positions	1 1	-			43	1	134	
Insert Unique with Postions	1 1)	(i) 10!	1.	1 -	4/4		M 1)	
		<u>.</u>		<u></u>	<u></u>		· ·	
Unique I last				F			J. Green	
void insert Unique (LIST * A, charc) {			300	in A	r -1991	+11 -1 - p	m	
1 / temp is a PN used for insertion, trav is a PPN used for	16	mon time.	093 %	Je gill	1250	- west	TO A	
traversing and accessing	1721	ickj	310	(D) HE	API			
LIST temp, * tray;	trai	v temp		elem	link	(3:3)	_
(3) If do dynamic allocation	3] []	17	A				1
temp = (LIST) mailor (size of (struct node));	1	0	1	(ক্ৰ	3,2)	87) 1	
Il checking if dy namic allocation is successfull	A	С ,	1	9 F W	4.0	Sr = /		\bot
if (temp (= NUL) {	I I	A				1 1		
3 1/100p for accessing ladditional condition for unique	111	parameters		lides 1		(8 m) 2(1)		1
for (tray=L; *tray!=NULL 28 (tray) > elem!=() tray= f (tray) > link) {}		The state of	- 10	±.				1
Il vandation. You have reached end of the cop because there's				2	trav	×* t	rav	1
no duplicare		L≈*trav		elem	link		link	4
if (* trav == NULL) {		力	1>	B	•	10		1
A do insertion (SEE how this is similar to meet Lost)	14	T		1			7	
(3.1) temp = elem = c; If ill in the node	1 X	are provided f		Me 9	7	A* = .	!	10
(3.2) temp-7 link = Null; Il since tast na man		O NE	0	(8)	(-+			6-0
(3.3) * trav = temp;								
By for * trav == Null .								
} 11 for temp!= NULL		i tre	17/	14	1. 45	7 7 4 6	11-136	in a
311 Whole close			î	3 (J)	15 400	High. 2	e white	100
Unique l First								
void insertunique (LIST * A, char c) {	97/42	A ZWAT					-	-
(1) 1/4emp is still a PN for insertion, trav here the is a PN used	-	tray (1	emp		7(1) Le		eri e	y à
for holding that other node just for linking purposes	- 11		П	100 100	491 de 10	4-2-14	4-4-54	
LIST temp, trav;				The same	100 M	2 1 1 1 T	1	
11 do dynamic allocation		A c			1 100	* Wild	-172	
temp = (USI) malloc (size of (struct node));			-	A 172		di dan		
11 checking if dynamic allocation is success full		F		100 Lill	HOLDAT (72	galse	-
if (temp (=NULL) {	-0	parameters		-	ad S			-
11 toop for accessing & additional condition for unique					1004	: Smit	1 1	
for (trav= *L; trav==NULL 42 trav= elem != c; trav=trav=link){}	18.6.5		Y	T	7 (f. f.)	1000	Lores Irel	T
//validation: You have reached end of the loop because there's	(A) 13 (A)	L	WE		elem la		elem in	
no auplicate		40		1	B [KE		1
if (trav==NULL) {	134/15/4			10.70	11 8 20	5 100	*	—
temp= elem=c; If fill in the mode					1,1107	2 5.0°T*	T.	_
temp > link = trav; II I howard the inserted to the UST							<u> </u>	_
trave temp. A make what is tricing to the PIRST	4	and the least of		da Torrix			de la Carlo	_

Limelig

Linked List Implem	entation (yelete) Para structure definition: typedef struct rode & tok; struct rode * tok; 3* UST;					
Delete First	Delete Last					
vold delete First (LIST * A) {	yoid delete Last (LIST * A) {					
1) I temp is a FN used to hold the first node than man man	(i) It temp is a PN used to note the node to delete to,					
ang 1-delete)	travis a PPN used for traversing & accessing (sourcust					
LIST temp;	LIST temp, * trav;					
2) // Separating this but you can do this one linera, let	(2) // loop for accessing					
temp point to the first mude	for(tray = A; * tray!= NULL; tray= &(*tray)→link) {}					
temp = *A;	(3) // deletion process					
3) I let the original usi (in the parameters) point to the	(5.1) // let temp point to target node					
Together next mode	temp=*trav;					
ences & A = temp-> link;	(3.2.) # let original usipoint to the next node					
Detaching the (originally) first node by free-ing	Arav= temp 7 link; (into using the parting to trash dange					
free (temp);	(3.3) // Detaching the last node by free-ing wary 6)					
LSTACK! LHEAP!	fra a (tomn).					
A, R, of A temp	dun't see anything O					
ldz:First ()	CUZ NO temper link C					
# &*A (4) &*A	Trave Temp (S.1.)					
L elem link elem link elem link						
A,R,OF LA XA DIN DO	≈ * fray × * fray (3.3)					
main()	L= *Array elem link elem link					
	A P B A					
	4 4 7					
9,454	0					
	When the state of					
Polete Element	The Mark the State of the State					
void delete Element (UST *A, char C) {						
(i) Itemp is a PN used to hold the node to delete to,	Company of the state of the sta					
trav is a PPN used for traversing and accessing	tray temp					
List temp, * trav;	(31)					
2 // 100p for accessing 2 additional condition for companing						
for (trav=A; *trav!=NULL & (*tray) > elem!= C. 1;	A C					
trav= 2(*tray) > 1 lnk) {}	B 3					
// validation: you have found the element & you didn't						
teoched the last element	Strav Strav Strav					
	elem link alem link clam link					
if (*tray!=NULL) { (3) //do delenion	A DA BACO					
	(3)					
temp = * trav; / let temp point to target node	7 4 5					
*trav = temp-> link; //let ORIGINAL us next node						
free (temp); (notaching the last node by free-ing						
	Limelight					

More Derived Deletes

Dale	te Element Sorted	LSTACK	HEAP	
	Void delete Element Sorted (LIST * A , Char c) {	1 0	Constitution of the	
1)	If temp is a PN used to hold the node to delete to.	trav temp	(311)	
<u> </u>				
	trav is a PPN used for traversing 2 accessing	11-		
2	LIST temp, * trav;	A C	The second second	
٧	Il loop for accessing & additional condition for Sorted	B-		
3	for (trav=A; *trav!=NULL 22 (*trav)=relem <c ;="" trav="2(*trav)?</td"><td>parameters</td><td>3/</td><td></td></c>	parameters	3/	
<u> </u>	II validation. You saw the element		* xxxxx	V 8 * HOV (3)
	if (* trav!= NULL 22 (*trav)=elem== = = = = = = = = = = = = = = = = = =	2 1 Table	1,7	etem link
	(3.11) Il lettemp point to target node	1 1	elem link	图图
	temp = * trav;	The state of the s		2)
	(312) If let original LIST point to the next node	1		elem link
	* trav = temp-7 link;			
	(3,3) is Detaching the chosen node by free-ing			Ta a
	free(temp);	The same the same		
	}			
	}	(f) w	to the second	1
		mate land mater the	r Hamilton and	
	void delete All Occurences (List * A, charc){	1 0_ 1		
<u>U</u>	1/ temp is a PN used to hold the node to delete to,	trav temp		
_	travis a ppi used for traversing & accessing	The second secon	(2.1)	
18	LIST temp, * trav			
(2)	il loop for accessing L1 condition only since we need	A		
_	to check all elements)	TO U	-6	36 (81.5)
	for (trav= A 3 * trav != NULL;){	paramores 5	X X	/
0	(i) il additional conditioninere, elemis found	1	1	
_	if ((*trav) -> elem == c) {	+		eta ≈×110
3	// do deletion		elem unt elem	
-	temp = * trav; // let temppoint to target node	134	NO GIVE	D N D
	trav = temp -tink, Het trou point to next note	1 11	63.	
	(313) free(tamp); Il detaching the Mast nade by free ing		6.2)	
	3 else {		The second second	
	Amove trav pointer only if node will not			
	be removed			
19.00	trov= 8 (*trou) + link	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1		
1	The second state of the second		1980 (000)	