Leaf nodes

(5) Programming: (WORKBOOK)

(1) struct node *s[10]
where s is an array, each element of which is a pointer to a structure of type node.

(2) Aliasing in the context of programming languages refers to multiple variables having the same memory location.

(3) Goal of structured programming is:
⇒ to be able to infer the flow of control from the program text.

(4) Abstract Data Type (ADT)
In order to simplify solving the problems the data structures are combined along with the operations called as ADT.
It includes: 1) Declaration of data
2) Operations.
⇒ ADT is a data type for which only the operations defined on it can be used, but none else.

(5) Both logic programming languages and functional languages are declarative.

(6) Features of an object oriented programming languages:
1) Abstraction and Encapsulation
2) Polymorphism in the presence of inheritance.

(7) Program takes 500 values. Range of values is 0 to 100 and program stores only frequencies of values above 50.
Therefore for each value above 50 a value representing its frequency needs to be stored. So an array of 50 integers is sufficient.

(8) $s=1,\ p=1,\ i=1$

for $i=1$: $p = p * x/i = 1 * \frac{x}{1} = \boxed{x}$

$s = s + p = \boxed{1+x}$

for $i=2$: $p = p * x/i = x * \frac{x}{2} = \boxed{\frac{x^2}{2}}$

$s = s + p = \boxed{1+x+\frac{x^2}{2}}$

for $i=3$: $p = p * x/i = \frac{x^2}{2} * \frac{x}{3} = \boxed{\frac{x^3}{6}}$

$s = s + p = \boxed{1+x+\frac{x^2}{2}+\frac{x^3}{6}} \Rightarrow 1+x+\frac{x^2}{2!}+\frac{x^3}{3!}$

for $i=4$: $p = p * x/i = \frac{x^3}{6} * \frac{x}{4} = \boxed{\frac{x^4}{24}}$

$s = s + p = \boxed{1+x+\frac{x^2}{2}+\frac{x^3}{6}+\frac{x^4}{24}} \Rightarrow 1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}$

According to Taylor's theorem:

$$\boxed{1+\frac{x}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}\ldots\ldots = e^x}$$

(9) `int * A[10], B[10][10]`

(1) $A[2] \Rightarrow *[A+2]$ ⇒ Here A is an array of pointer where A[2] contains the address of some integer variable. So it will not give any error.

(2) $A[2][3] \Rightarrow *[*[A+2]+3] \Rightarrow *[A+2]$ will point to the value at that address and +3 will be added and * means value means it will point to some outside array so it will be used as an assignment. But the value will be stored outside the given array.

3) B[1] ⇒ *(B+1) ⇒ it will give error as it is a one dimensional array where B[0][0] contains the ~~base address~~ value at 0th location. So it cannot be used as an assignment as it is not pointing to any location.

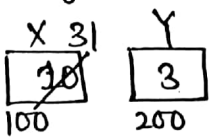
4) B[2][3] = *(*(B+2)+3). In this we can store values in 2nd row and 3rd column. Assignment is possible.

(1,2,4)

(10) C language is Context Sensitive language.

(11) P1 = call by reference.

Program P1

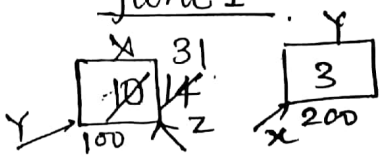


func1(y, x, x)

func1(200, 100, 100)

print x; 31
print y; 3

func1



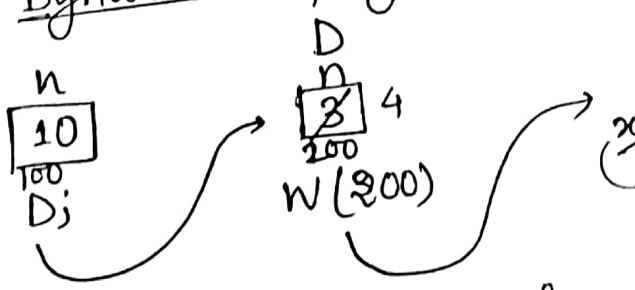
func1(*x, *y, *z)

*x points to 200
*y points to 100
*3 points to 100

*Y = y + 4
= 10 + 4
= 14

*z = *y + *x + *3
= 14 + 3 + 14
*z = 31

(12) Dynamic scoping and call by reference: (d)



Print x; 4 as dynamic scoping (free variables are resolved from previous calling func^n).

(13) In C language, the no. of activation records between the current activation record and the activation record for the main depends on the actual calling func^{n} sequence. The stack containing the activation record.

(14) Short → 2 Bytes
float → 4 Bytes
long → 8 Bytes

Short s[5] takes 2*5 ⇒ 10 Bytes

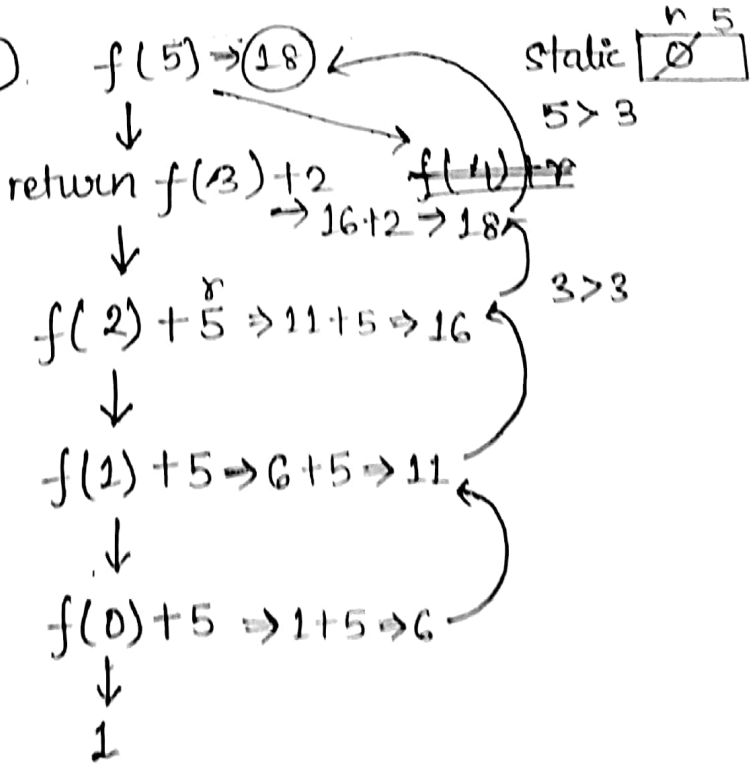
In Union, the largest Bytes the variable is holding is considered.
i.e long z which takes 8 Bytes.

Structure will take 10 Bytes along with 8 Bytes.

Total of 10+8 = 18 Bytes

(15) An unrestricted use of the "goto" statement is harmful because it makes it more difficult to verify programs.

(16)



(17) main()

a	sum
2048	0

Print: 0

foo(2048, 0) k=Ø j=Ø 204

k = 2048 % 10 = 8

j = 2048/10 = 204

sum = sum + k;

sum = 0 + 8

sum: Ø8

foo(204, 8) pf: 8

k	j	
~~8~~4	~~204~~	20

k = 204 % 10 = 4

j = 204/10 = 20

sum = ~~8~~ 12

sum = sum + k = 8 + 4 = 12

foo(20, 12) pf: 4

k	j	
~~4~~0	~~20~~	2

k = 20 % 10 = 0

j = 20/10 = 2

sum 12

sum = 12 + 0 = 12

foo(2, 12) pf: 0

k	j	
Ø2	~~=2~~	0

k = 2 % 10 = 2

j = 2/10 = 0

sum ~~12~~ 14

sum = 12 + 2 = 14

foo(0, 14) pf: 2

~~return~~

return;

Print: 20, 48,

O/P: 2, 0, 4, 8, 0

(18) The given c program is non-recursive. So we will get the value of n after some iteration. The main thing about the program is that.

1. If $m = n$ then n is the GCD.
2. If $m > n$ then $m = m - n$
3. If $m < n$ then $n = n - m$

Ex: Let $x = 40$ & $y = 32$

$m = x = 40$, $n = y = 32$

$40 != 32$ True

$\Rightarrow 40 > 32$ True. so apply 2nd condition.

$m = m - n = 40 - 32 = 8$

$m = 8, n = 32$

$\Rightarrow 8 != 32$

$32 \not< 8$, false, so apply 3rd condition

$n = n - m;$

$= 32 - 8;$

$= 24$

$\Rightarrow 8 != 24$

$8 > 24$

$n = 24 - 8 = 16$

$\Rightarrow 8 != 16$

$8 > 16$

$n = 16 - 8 = 8$

$\Rightarrow 8 != 8$, false.

Print : 8 GCD.

GCD (40, 32) = 8.

(19)

i=1 2 3 4 5 6 7

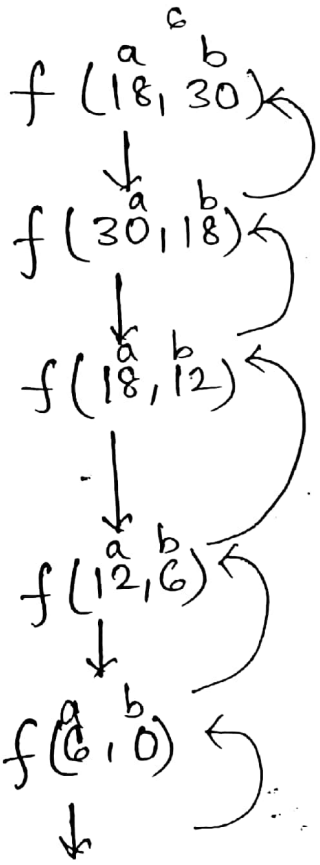
1>=5. false
n=1+1=2

2>=5 false
n=2+2=4

4>=5 false
n=4+3=7

7>=5 True.

(20)



30>18, true

18>30, false
18==0, false
30%18=12

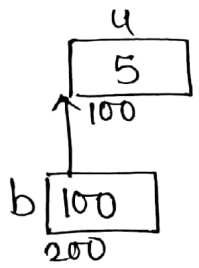
12>18, false
12==0, false
18%12=6

6>12, false
6==0 false
12%6=0

0>6, false
0==0, True

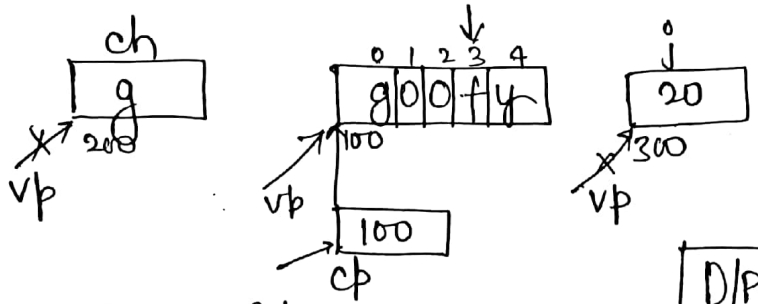
(21) To store "John" in a character array
no. of bytes required is 5 "JOHN\0"
↑↑↑↑↑
5 Bytes including \0 string.

(22)



It produces error.
Not possible a*b. Compiler will not be able to differentiate either it is a pointer or value.

(23)



O/P: g20fy.

vp is a void pointer.

vp = &ch | 200 | Print g;

vp = &j | 300 | Print 20;

vp = cp | 100 | 100+3 Print from f till \0.
=103

(24)

inc(); inc(); inc();

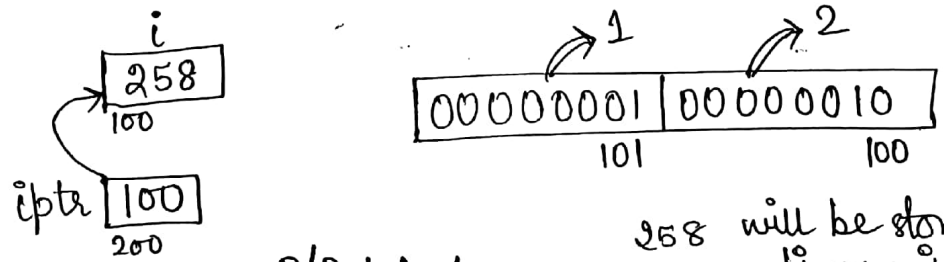
x Intially 0

static 0 1 2 3

static i
value will not ~~chan~~ be gone.
Static scope is till the end of program.

O/P: 1, 2, 3

(25)



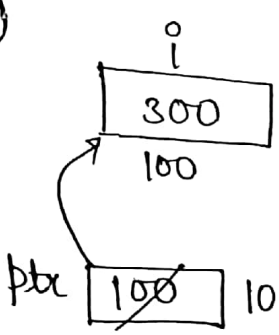
258 will be stored in the form of binary in m/o.

O/P: 2, 1

* ((char*)iptr) ⇒ It will take only the value of 100 as it is char(-type casting) takes only 1 Byte.
In 100 value is 2.

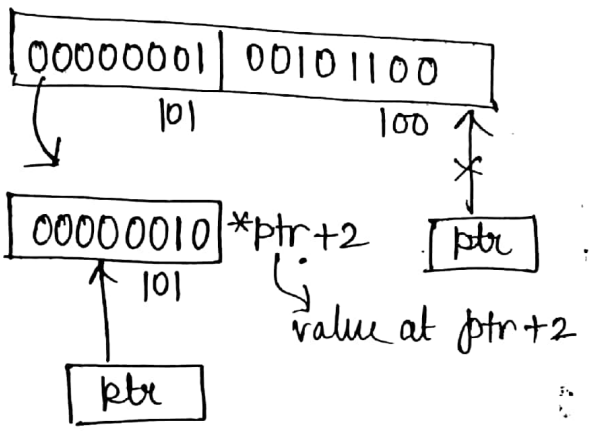
* ((char*) iptr+1) ⇒ 101 contains value 1 which will be printed (100+1).

(26)

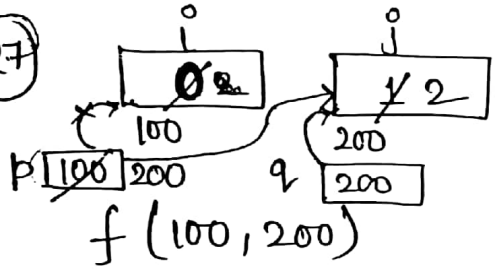


O/P: 556

512 32 8 4
1000101100



(27)



p=q ⇒ 200 is stored in p
*p=2 ⇒ value at p is updated with 2

pf(i, j)

O/P: 0, 2.

(28) if (a==b) ⟹ In this condition it is checking if a is equal to b if all are equal then it will exit by default c=d.

{ S1: a, b, c and d are equal;
exit;
}
else if (c==d).
~~else if {a, b, c and d are all distinct}~~

T1, T2, T4

~~else~~ {a!=b and c=d;} → In this condition a!=b as above if condition fails. if c=d it will execute this case.

If it fails:

else
a, b, c and d are distinct;
}

means a!=b and c!=d so all the a, b, c, d are distinct.

(29) x++ ⇒ post increment.
It will increment after the if condition.

i=0 (0%2 && 0)
0
False then it will not check further as && is given. If any one one value is false whole will be false.

i=1 (1%2 && 0) = F out of if()
T F
x = x+2 = 2+2 = 4

x
0 1 2 4 5 7 8 10 11 13

i=2 (2%2 && 1)
0

i=3 (3%2 && 1)
T T
x = x+2 = 4

i=4 (4%2 && 4)
0

i=5 (5%2 && 4)
T T
x = 5+2 = 7

i=6 (6%2 && 7)
F

i=7 (7%2 && 7)
T T
x = 8+2 = 10

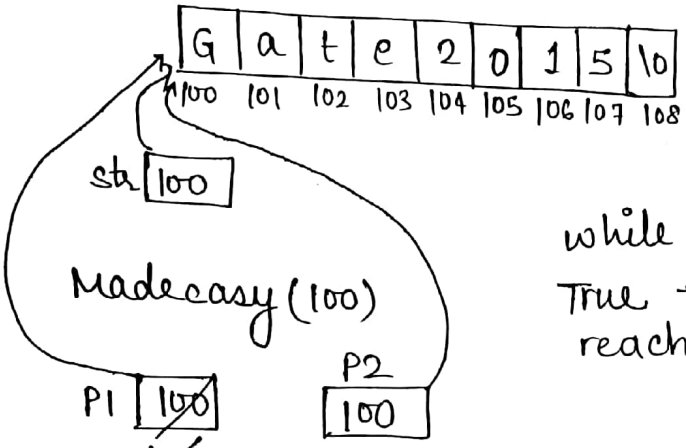
i=8 (8%2 && 10)
F

i=9 (9%2 && 10)
T
x = 11+2 = 13

i=10 false.

x = 13

(30)



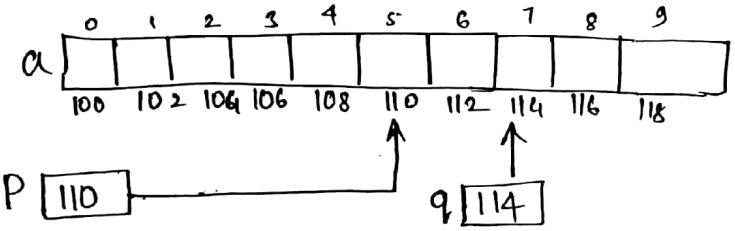
while (*++P1) It will execute from Right to Left.
True till it reaches \0.
1st Increment then value.

~~100~~ 101
~~101~~ 102 ~~103~~ It checks the ASCII value of each character.
~~104~~ 105 ~~104~~
~~106~~ ~~107~~ 108

return (108 − 100)

O/P: 8

(31)



(a) p+1 = (110+1) = 112 possible
(b) q-3 = (114-3*2) = 108 possible
(c) p+q = 110+114 = Not possible = Addition is not possible.
(d) q-p = Substraction of 2 pointer pointing to same array is possible.

(32) a) switch (b*c)
→ 2*3=6
⇒ It will go to case 6 and break out of the switch case.

b) switch (b*c-2)
→ 2*3-2 = 4
⇒ It will go to case 4 and print 2 only and then break out of loop.

c) switch (b+c*2)
→ 2+3*2 = 2+6 = 8
⇒ It will go to case 8 and print value of c i.e 3 and then print value of b i.e 2 as no break is there in switch - this ∴ O/P will be 32.

(33) main()

b
1

a++: Post increment: first-print then increment.

b = f(1) Print: 1

a
1 2 3

return 3

b = 3

++a: Pre increment: first increment then return.

b = f(3)

a
3 4 5 Print: 3

return 5

b = 5

b = f(1 + f(5))

a
5 6 7 Print: 5

return 7

b = f(1+7) · f(8)

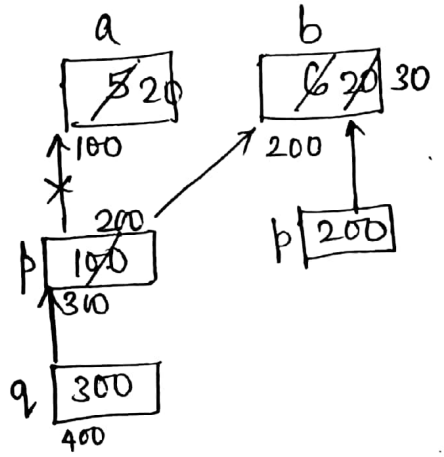
a
8 9 10 Print: 8

return 10

b = 10

O/P: 1 3 5 8

(34)



*p = 20 20 is assigned to value at p.

f(20, 200) → address of b and *p is a pointer holding that address.

O/P: 20, 30.

x
20 10

call by value.

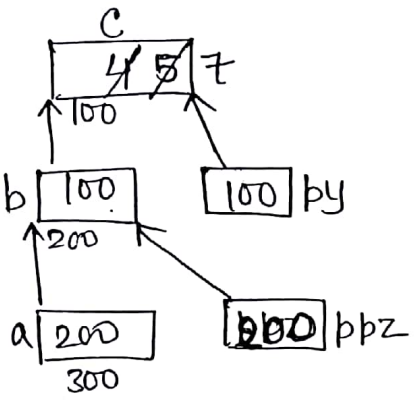
⇓

Call by reference

Q35 Just try with stack,

Ans: DCBA.

(36) main()



f(4, 100, 200)
i b u

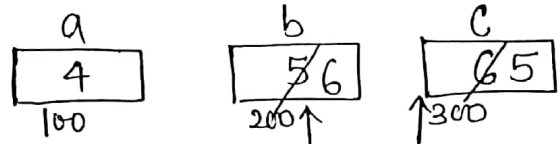
return (7+7+5) ⇒ 19

P/f: 19

** ppz = value at value at that location.

∵ x is a local variable for function f(). So x = x+3 will updated in function.

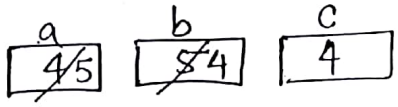
(37)

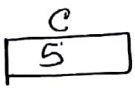


call by value
f1(4,5)

a 200 b 300

call by reference
f2(200, 300)

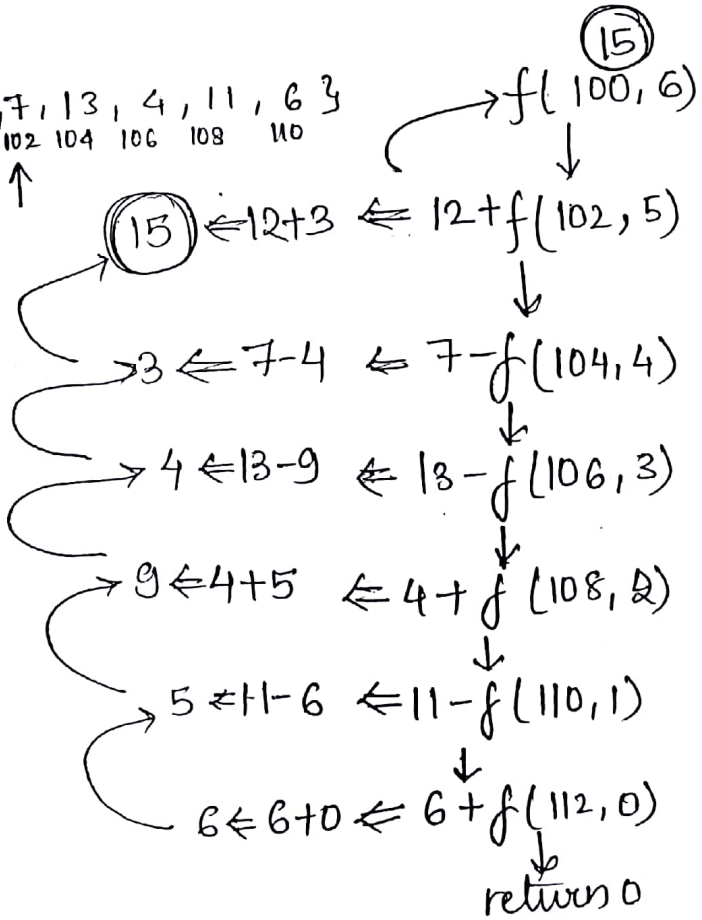




pf: c-a-b = 5-4-6
= -5

(38) main()

a[] = {12, 7, 13, 4, 11, 6}
100 102 104 106 108 110



n: 6 5 4 3

100%2 == 0 True
a+1 = 100+1*2

7%2 == 0 False
a+1 = 102+1*2 = 104

13%2 == 0 false
a+1 = 104+1*2 = 106

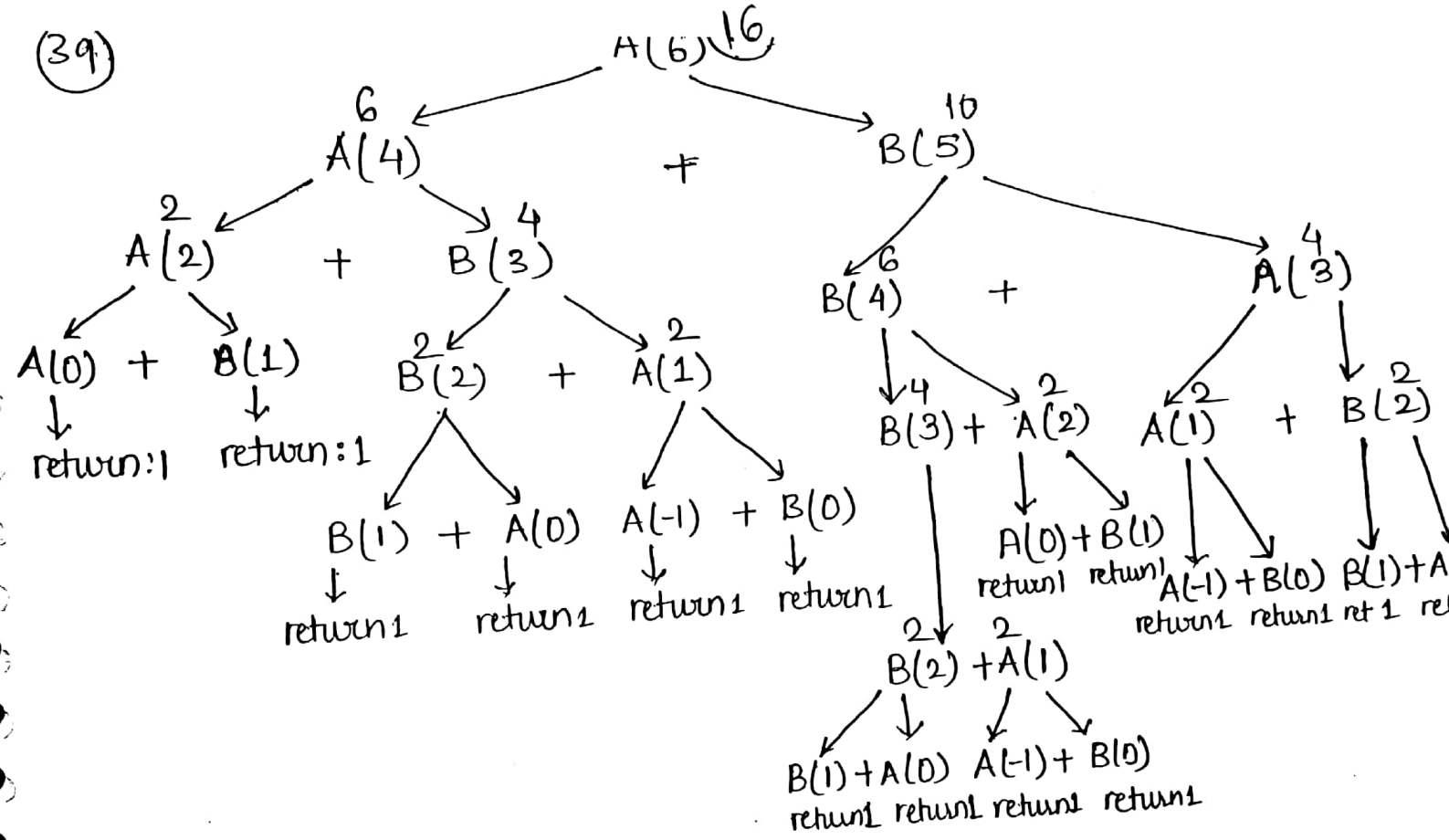
4%2 == 0, True
a+1 = 106+1*2 = 108

11%2 == 0, false
a+1 = 108+1*2 = 110

6%2 == 0, True
a+1 = 110+1*2 = 112

n <= 0, return

(39)

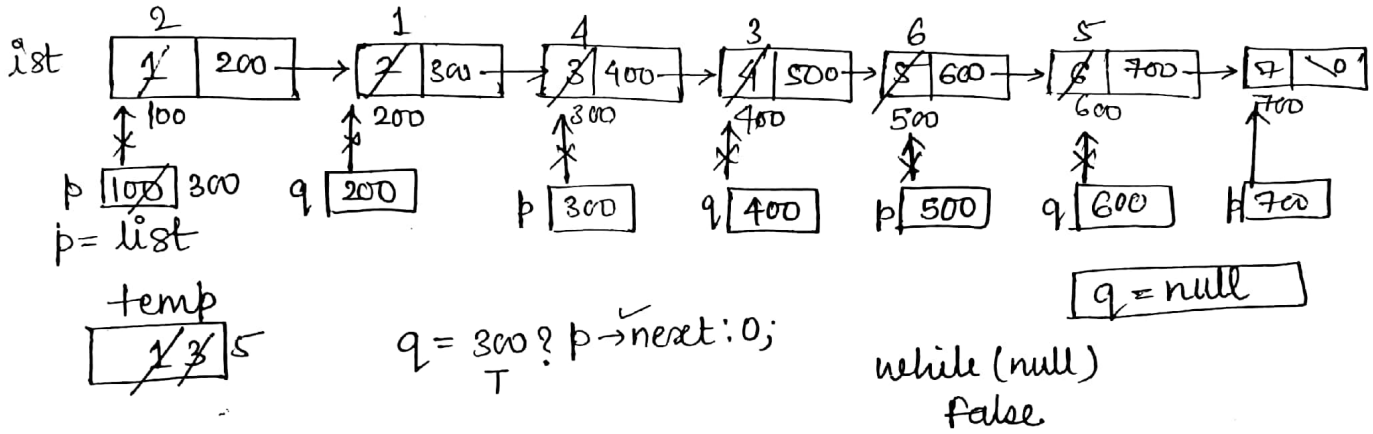


(3) LINKED LIST, HASHING, STACK, QUEUE AND ARRAY:

(1) ~~If p points to list, then enqueue and dequeue can be performed using a single pointer.~~

⇒ Not possible because, here we need to use only single pointer 'p'. (Don't use rear and front pointers).

(2) 1, 2, 3, 4, 5, 6, 7



Contents of list: 2 1 4 3 6 5 7

(3) ⇒ In a labeled binary tree preorder traversal produces prefix expression, postorder traversal ⇒ postorder expression and inorder traversal ⇒ inorder expression.

⇒ Preorder traversal: root is first element and using inorder we can identify left subtree nodes and right subtree nodes nodes for that root. So we can construct unique Binary tree using preorder and inorder.

Similarly postorder and inorder: The postorder numbers assigned to the nodes have the useful property that the nodes in the subtree with root n are numbered consecutively from (postorder(n) − desc(n)) to postorder(n)

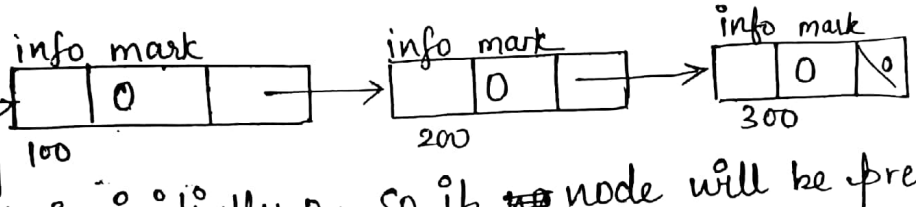
To test if a vertex x is a descendent of vertex y then
postorder(y) - desc(y) ≤ postorder(x) ≤ postorder(y)

A similar property holds for preorder and doesn't hold for inorder.

(4) In union, every element in List 1 and List 2 needs to be present only once. Therefore every element of List 1 has to be compared with List 2 similarly with Intersection, every common element needs to be present in the result.
∴ Both lists have to be traversed which take more time.
unlike in cardinality & Membership lists has to be traversed only one

(5) for deleting the last element the lists needs to be traversed till the previous of last.
∴ Length of the list is required.

(6)



Mark is initially 0. So if ~~too~~ node will be present then:
a → mark will ~~become~~ point to mark field of first node first.
which is initially 0 then it becomes false:
then return

nodecount(a→p) + nodecount(a→p)+1;

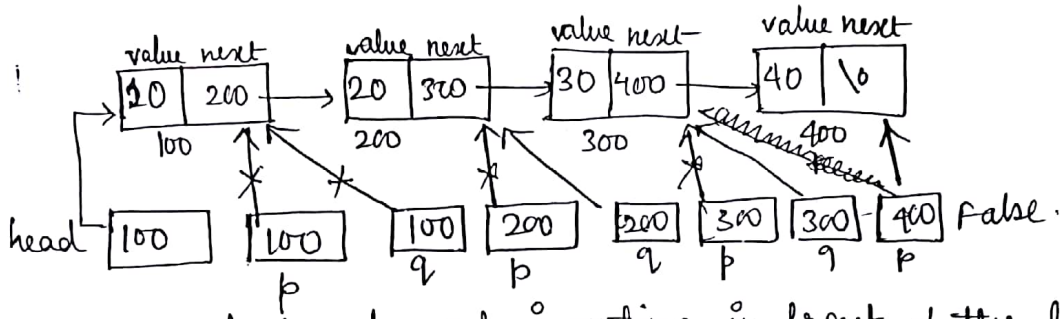
- If we Add a → mark = 1, as the first statement, then it will make if statement True and it will return 0 even if nodes will be present.
- If we Add a → mark = 0 after the if statement then it will be counted one more time if any loop will be there.

(C)
- If we Add a → mark = 1 ; after the if statement then this program will work correctly and as we make a → mark = 1 that node will not be counted by the compiler as we have already traversed that node.
- a → mark = 0 as last statement then also it will not work correctly.

+) In doubly linked list since we have "former pointing to previous and next node.

(7) Therefore if one pointer is lost, the lost pointer can be recovered easily with the help of other pointers whereas remaining options can be done in single link list also and greatest advantage is option (b).

(3)
(8)



Removing last node and inserting in front of the list
So first we will check if list is empty or not or having one node:

```
If (head == NULL) || (head->next == NULL)
then return head;
```

we use two pointer p and q.
where q = NULL and p = head;

```
while (p->next != NULL)   it will go till last node
{
    q = p;
    p = p->next;
}
```

Now we have to delete last node which is pointed by p and previous node is pointed by q.

we make q->next = NULL

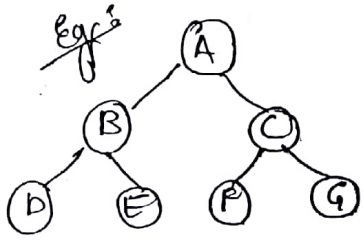
```
p->next = head   // Adding at first.
head = p;
```

(9) As it is a complete binary tree, so LASTIN = LASTPRE
The last vertex visited in
Postorder = Left Right Root.
DEBFGCA

Preorder = Root left Right
ABDECFG

Inorder = Left Root Right
DBEAFCG

Eg:



(10) In m-ary tree where m = 0 or 3 here.

Total no of nodes = m (internal nodes) + 1

Here, $n = 3i + 1$

Total nodes = internal nodes + leaf nodes.

$n = i + l$

$i = n - l$

$n = 3(n-l) + 1$ {From above}

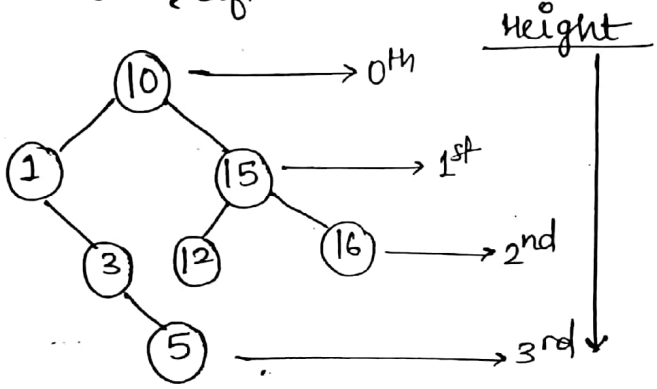
$n = 3n - 3l + 1$

$3l = 2n + 1$

$$l = \frac{2n+1}{3}$$

where l is leaf nodes.

(11)



Height of binary search tree = 3

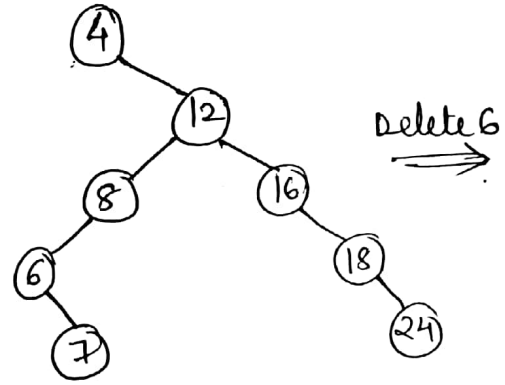
(12) Binary search can be carried out on a set of ordered data items stored in a array.

(13) A binary tree can be reconstructed from the following traversal(s):

Inorder and Preorder ✓

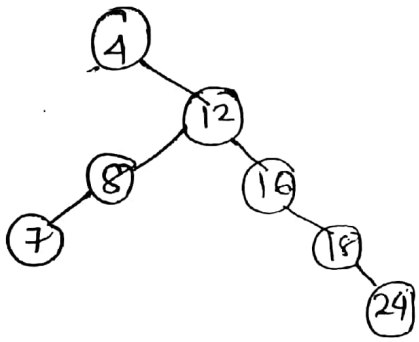
Inorder and Postorder

(14)



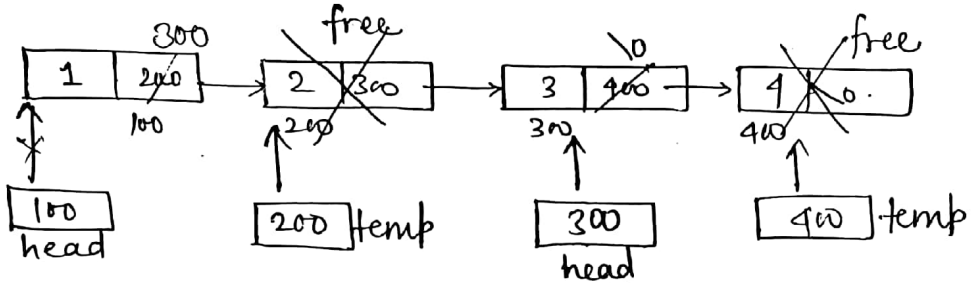
Delete 6 ⟹

Preorder traversal: 4, 12, 8, 7, 16, 18, 24



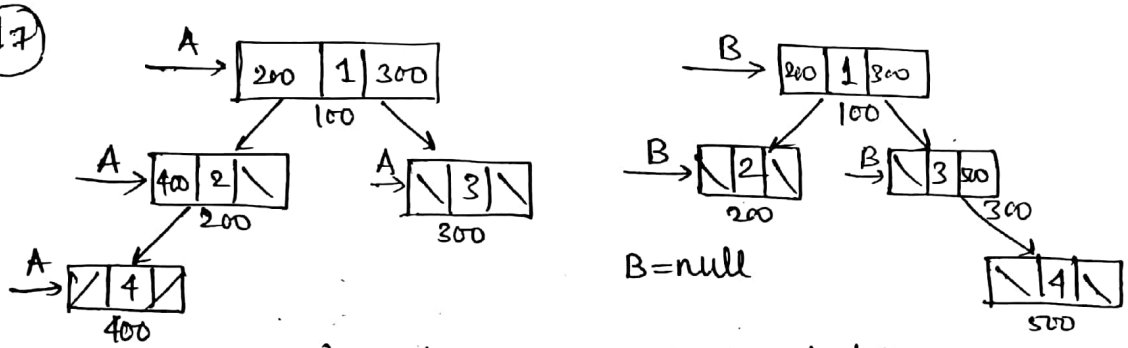
15) The concatenation of 2 lists can be performed in O(1) time in doubly linked list as doubly linked list is circular having two pointers pointing to both previous and next node so it will take O(1) time. and as it is circular last node will point to the first node.

16)



Deleting alternate nodes.

17)

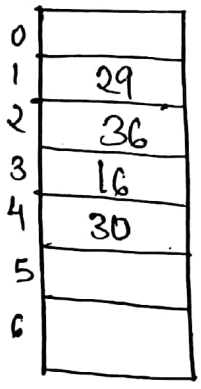


This will return 1 as A != NULL but B = NULL

It compares two given binary trees and return 1 if two trees are different and it returns 0 otherwise

18) $h(K) = K \bmod 7$

Keys: 29 · 36 16 30 Linear probing.



29 % 7 = 1

36 % 7 = 1 // collision (1)

((36 mod 7) + 1) mod 7 = 2

16 % 7 = 2 // collision (1)

((16 mod 7) + 1) mod 7 = 3

30 % 7 = 2 // collision (2)

(30 % 7) + 1) % 7 = 3

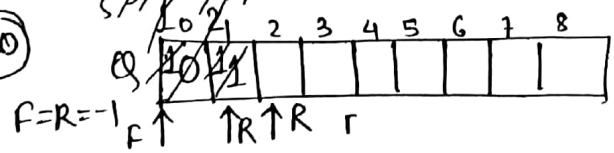
((30 % 7) + 2) % 7 = 4

4

(19) · $h(K) = K \bmod m$

(b) 701 as it a prime number as prime numbers not near to power of 2 is considered as Best mod fun^n gives less collision

(20)



$n = 5$

i=0

x = dequeue(Q)
x = 0
y = dequeue(Q)
y = 1
enqueue(Q, 1)
enqueue(Q, 1)

i=1

x = 1
y = 1
enqueue(Q, 1)
enqueue(Q, 2)

i=2

x = 1
y = 2
enqueue(Q, 2)
enqueue(Q, 3)

i=3

x = 2
y = 3
enqueue(Q, 3)
enqueue(Q, 5)

i=4

x = 3
y = 5
enqueue(Q, 5)
enqueue(Q, 8)

Print: 0, 1, 1, 2, 3

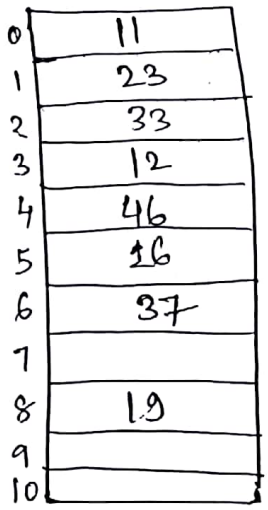
It prints first 5 fibonacci numbe

(21) Choosing the hash function randomly from a class of hash functions such that it is independent of the keys to be stored, is termed as universal hashing.

(22) $h(k,i) = (h'(k) + i + i^2) \bmod 11$

$h'(k) = k \bmod 11$.

23 12 19 11 33 16 46 37



collisions = 4

$h(23,0) = (h'(23)+0+0) \bmod 11 = 1$
$= 23 \% 11$

$h(12,0) = (h'(12)+0+0) \bmod 11 = 1$
$= 12 \% 11$

$h(12,1) = (h'(12)+1+1) \bmod 11 = 3$ // (1)
$= 12 \% 11$

$h(19,0) = (h'(19)+0+0) \bmod 11 = 8$
$= 19 \% 11$

$h(11,0) = (h'(11)+0+0) \bmod 11 = 0$
$= 11 \% 11$

$h(33,0) = (h'(33)+0+0) \bmod 11 = 0$
$= 33 \% 11$

$h(33,1) = (h'(33)+1+1) \bmod 11 = 2$ // (2)
$= 33 \% 11$

$h(16,0) = (h'(16)+0+0) \bmod 11 = 5$
$= 16 \% 11$

$h(46,0) = (h'(46)+0+0) \bmod 11 = 2$
$= 46 \% 11$

$h(46,1) = (h'(46)+1+1) \bmod 11 = 4$ // (3)
$= 46 \% 11$

$h(37,0) = (h'(37)+0+0) \bmod 11 = 4$
$= 37 \% 11$

$h(37,1) = (h'(37)+1+1) \bmod 11 = 6$ // (4)
$= 37 \% 11$

(23) # Slots = 10,000 (M)
entries = 9800 (N)

$\alpha = \frac{N}{M} = \frac{9800}{10000} = (0.98)$

The expected no. of probes in successful search

$= \frac{1}{\alpha} \ln \frac{1}{1-\alpha} = \frac{1}{0.98} \ln \frac{1}{1-0.98} = 3.9 \cong 4$.

(24) The expected no. of probes in unsuccessful search.

$= \frac{1}{1-\alpha} = \frac{1}{1-0.98} = 50$.

(25) keys: 12, 18, 13, 2, 3, 23, 5, 15

$h(k) = k \bmod 10$.

0	
1	
2	12
3	13
4	2
5	3
6	23
7	5
8	18
9	15

12 % 10 = 2
18 % 10 = 8
13 % 10 = 3
2 % 10 = 2
(2+1) % 10 = 3
((2 % 10) + 2) % 10 = 4
3 % 10 = 3
((3 % 10) + 1) % 10 = 4
((3 % 10) + 2) % 10 = 5
23 % 10 = 3
((23 % 10) + 1) % 10 = 4
((23 % 10 + 2) % 10 = 5
((23 % 10 + 3) % 10 = 6

5 % 10 = 5
((5 % 10) + 1) % 10 = 6
((5 % 10) + 2) % 10 = 7
15 % 10 = 5
((5 % 10) + 1) % 10 = 6
((15 % 10) + 2) % 10 = 7
(15 % 10 + 3) % 10 = 8
(15 % 10 + 4) % 10 = 9

(26) M = 11 slots

Keys : 43, 23, 1, 0, 15, 31, 4, 7, 11, 3

0	43
1	0
2	31
3	1
4	
5	7
6	23
7	15
8	11
9	4
10	3

$(h1(43)+0)\%11$
$x=(43+5)*(43+5)$
$x=2304/16$
$x=144+43$
$x=187\%11=0$

$(h1(23))+0)\%11$
$x=(23+5)*(23+5)$
$x=784/16$
$x=49+23$
$x=72\%11=6$

$(h1(1)+0)\%11$
$x=(1+5)*(1+5)$
$x=36/16$
$x=2+1$
$x=3\%11=3$

$(h1(0)+0)\%11$
$x=(0+5)*(0+5)$
$x=25/16$
$x=1+0$
$x=1\%11=1$

$(h1(15)+0)\%11$
$x=(15+5)*(15+5)$
$x=400/16$
$x=25+15$
$x=40\%11=7$

$(h1(31)+0)\%11$
$x=(31+5)*(31+5)$
$x=1296/16$
$x=81+31$
$x=112\%11=2$

$(h1(4)+0)\%11$
$x=(4+5)*(4+5)$
$x=81/16$
$x=5+4$
$x=9\%11=9$

$(h1(7)+0)\%11$
$x=(7+5)*(7+5)$
$x=144/16$
$x=9+7$
$x=16\%11=5$

$(h1(11)+0)\%11$
$x=(11+5)*(11+5)$
$x=256/16$
$x=16+11$
$x=27\%11=5$
$(h1(11)+1)\%11$
$=6$
$(h1(11)+2)\%11=7$
$(h1(11)+3)\%11=8$

$(h1(3)+0)\%11$
$x=(3+5)*(3+5)$
$x=64/16$
$x=4+3=7$
$x=7\%11=7$
$(h1(3)+1)\%11=8$
$(h1(3)+2)\%11=9$
$(h1(3)+2)\%11=10$

(27) $h(K) = K \bmod 10$

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

46, 34, 42, 23, 52, 33

46 % 10 = 6
34 % 10 = 4
42 % 10 = 2
23 % 10 = 3
52 % 10 = 2
(52+1) % 10 = 3
(52+2) % 10 = 4
(52+3) % 10 = 5
(33 % 10) = 3
(33+1) % 10 = 4
(33+2) % 10 = 5
(33+3) % 10 = 6
(33+3) % 10 = 7

(28) [redacted] Ans: 30 (Apply all possible combination and check).

(29) 4322, 1334, 1471, 9679, 1989, 6171, 6173, 4199

$h(x) = x \% 10$

4322 % 10 = 2
1334 % 10 = 4
1471 % 10 = 1
9679 % 10 = 9
1989 % 10 = 9
6171 % 10 = 1
6173 % 10 = 3
4199 % 10 = 9

1) 9679, 1989, 4199 hash to same value.
2) 1471, 6171 hash to same value.

46, 34, 42, 23 — 52, 33 (can't go inside)

6 4 2 3

These 4 can be arranged in 4! ways = 24.

but 46 can come outside after 52.

|34 42 23| |52 46 33|
3! 1!
= 6
24 + 6 = 30

(30) 1, 3, 8, 10

$(3x + 4) \% 7$

$\Rightarrow (3*1+4) \% 7$
$\Rightarrow 7 \% 7 = 0$

$\Rightarrow (3*3+4) \% 7$
$\Rightarrow 13 \% 7$
$\Rightarrow 6$

$\Rightarrow (3*8+4) \% 7$
$\Rightarrow 28 \% 7$
$\Rightarrow 0$

$(28 \% 7 + 1) \% 7$
$\Rightarrow 1$

0	1
1	8
2	10
3	
4	
5	
6	3

$(3*10+7) \% 7$
$\Rightarrow 37 \% 7$
$\Rightarrow 2$

31) size of m/o = 3'
Base address = 1120
A[49] = ?
A[49] = 1120 + (49-1)*3
= 1264.

32) Minimum two stacks of size n is required to implement a queue of size n.

33) Base address = 0
location of A[40][50] = Base Address + [(i - lb1)*n + (j - lb2)]* size
A[100][100] = A[0...99][0...99]
lb1 lb2
n = 99 - 0 + 1
#columns
n = 100
= 0 + [(40-0)*100 + (50-0)]
= 4050

34) S

10	20	30	40	50
0	1	2	3	4

reverse (S, i, j) 1 ≤ k ≤ n, n = 5

K=1 reverse(S,1,1), reverse(S,2,5), reverse(S,1,n)

10	20	30	40	50
1	2	3	4	5

K=2 reverse(S,1,2), reverse(S,3,5), reverse(S,1,5)

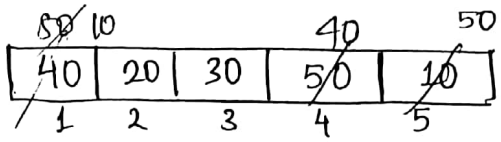
30	20	10	40	50
1	2	3	4	5

K=3 reverse(S,1,3), reverse(S,4,5), reverse(S,1,5)

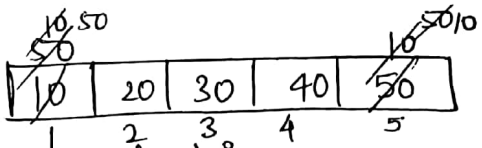
30	20	10	40	50
1	2	3	4	5

→ next ==NULL)

K=4 reverse (S,1,4), reverse (S,5,5), reverse (S,1,5)



K=5 reverse (S,1,5), reverse (S,1,5), reverse (S,1,5)



rotate S left by K positions.

(35) $A=\begin{bmatrix}1 & 0 & 0\\ 2 & 3 & 0\\ 4 & 5 & 6\end{bmatrix}$ $B=\begin{bmatrix}7 & 0 & 0\\ 8 & 9 & 0\\ 1 & 2 & 3\end{bmatrix}$ $B^T=\begin{bmatrix}7 & 8 & 1\\ 0 & 9 & 2\\ 0 & 0 & 3\end{bmatrix}$

$C=\begin{bmatrix}1 & 7 & 8 & 1\\ 2 & 3 & 0 & 2\\ 4 & 5 & 6 & 3\end{bmatrix}_{n\times n+1}$ $B[i,j]=C[j,i+1]$

(36) $A[0\ldots n-1, 0\ldots n-1] = A[0\ldots 99][0\ldots 99]$

$B[0\ldots \frac{1}{2}n(n+1)-1] = B\left[0\ldots \frac{1}{2}100\times 101-1\right] = B[0-5049]$

$n=100$.

if A[0][0] stored in B[0] $\left\{\#\ \frac{n(n+1)}{2}\text{ in lower }\Delta^{le}\right\}$

Q: A[90][80] in B array = ?

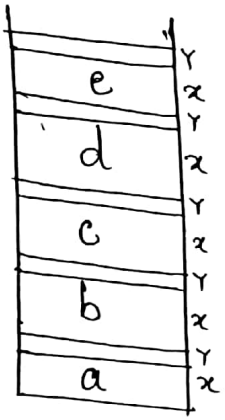
$A[90][80] = 100 + \left(\frac{90\times 91}{2} + (80-0)\right)*1$

$100+(x-0) = 100+(4175)$

$\boxed{x=4175}$

(37) To evaluate an expression without any embedded function calls one stack is enough.

(38)



$e \Rightarrow y$

$d \Rightarrow y+x+y+x+y = 2x+3y$

$c \Rightarrow y+x+y+x+y+x+y+x+y = 5y+4x$

$b \Rightarrow 6x+7y$

$a \Rightarrow 8x+9y$

$\Rightarrow y+2x+3y+4x+5y+6x+7y+8x+9y$

$\Rightarrow 2x+4x+6x+8x+y+3y+5y+7y+9y$

$\Rightarrow 2x(1+2+3+4)+y(1+3+5+7+9)$

$\Rightarrow 2x\left\{\frac{(n-1)(n-1+1)}{2}\right\}+y(n^2)$

$\Rightarrow x(n-1)n+yn^2$ { sum of n odd numbers $= n^2$ }

for any: $\Rightarrow \frac{x(n-1)n}{n}+\frac{y(n^2)}{n}$

$\Rightarrow xn-x+y$

$\boxed{n(x+y)-x}$

(39)

Assume N = 10

Top1 Top2

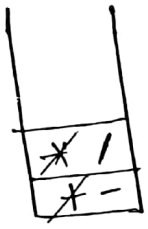
$\boxed{Top1 = Top2 - 1}$ will give the stack full condition

(40) To remove recursion from a program we have to use the following data structures that is stack.

(41) $1*2\wedge 3*4\wedge 5*6$

$1*8*1024*6 = \underline{49152}$.

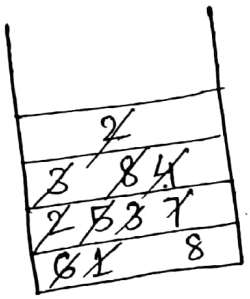
(42) A+B*C-D/E



⇒ ABCDE

Maximum size of the operator stack = 2.

(43) 623+-382/+*



op2 op1
2+3 = 5
6-5 = 1
8/2 = 4
3+4 = 7
7*1 = 8

Maximum size of operand stack = 4

(44) a+b x c - d ^ e ^ f
a+b x c - d ^ ef ^
a+b x c - def ^ ^
a+bcx - def ^ ^
abcx+ - def ^ ^
abcx+def ^ ^ -

(46) Maximum possible height of an AVL tree with 20 nodes = min $(h-1)+\min(h-2)+1$

$20 = (h-1)+(h-2)+1$

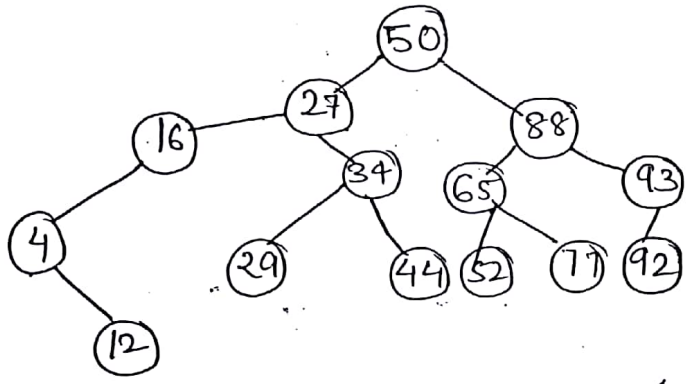
$= 2h-3+1$

$20 = 2h-2$

$10 = h$

$h=0$	1 node
$h=1$	2 node
$h=2$	4 node.
$h=3$	7 node
$h=4$	12 node
$h=5$	20 nodes

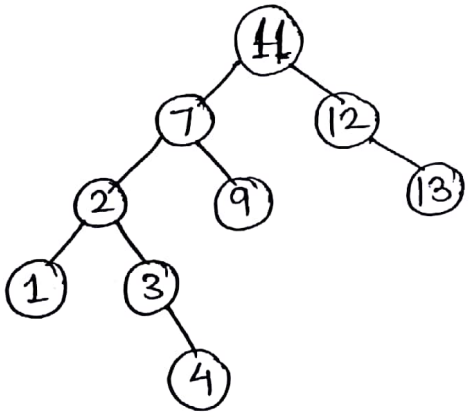
(47) 50, 27, 16, 88, 34, 65, 52, 77, 93, 4, 12, 29, 44, 92



(3)

Preorder:	50	27	16	4	12	34	29	44	88	65	52	77	93	92
Postorder:	12	4	16	29	44	34	27	52	77	65	92	93	88	50

(48) 11, 7, 12, 13, 2, 9, 1, 3, 4

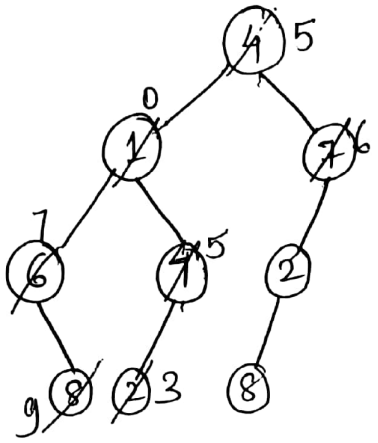


Preorder: 11, 7, 2, 1, 3, 4, 9, 12, 13

(9)

ad → next == NULL)

(49)



5+24+18 = 47

4 % 2 == 1, False

gate = 12

root → data = 4+1 = 5

5 + g(1) + g(7) (24, 18)

root → data = 0

0 + g(6) + g(4) (16, 8)

root → data = 7

7 + g(Null) + g(8) (9)

root → data = 9

9 + null + null

root → data = 6

6 + g(2) + null (2)

root → data = 5

5 + g(2) + Null (2)

root → data = 3

3 + null + null

root → data = 3

3 + g(8) + null (9)

root → data = 9

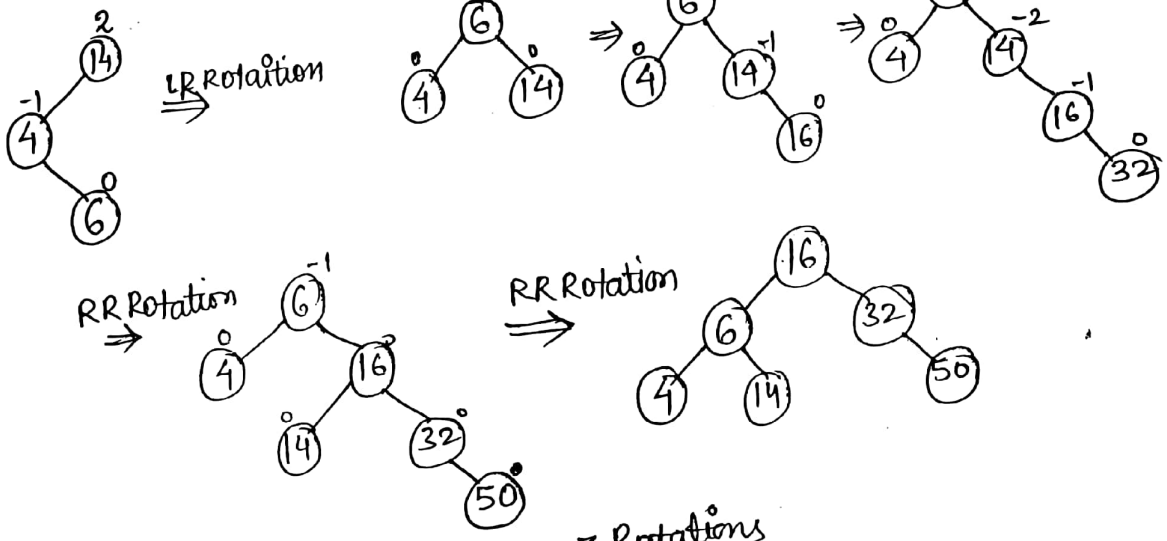
9 + null + null

(50) Left subtree: $2^7 - 1 = \frac{127}{2} = 63$

Right subtree: $12 + 20 + 1 = \frac{33 \times 1}{3} = 11$

63 + 11 + 1 = 75 (Root)

(51) 14, 4, 6, 16, 32, 50



3 Rotations.

(52) 12, 44, 13, 88, 23, 94, 11, 39, 20, 16, 5

0	11
1	39
2	20
3	5
4	16
5	44
6	88
7	12
8	23
9	13
10	94

$h(i) = (2i+5) \bmod 11$

$12 = (2*12+5) \% 11 = 7$

$44 = (2*44+5) \% 11 = 5$

$13 = (13*2+5) \% 11 = 9$

$88 = (88*2+5) \% 11 = 5$

$= (5+1) \% 11 = 6$

$23 = (23*2+5) \% 11 = 7$

$= (7+1) \% 11 = 8$

$94 = (94*2+5) \% 11 = 6$

$= (6+1) \% 11 = 7$

$= (6+2) \% 11 = 8$

$= (6+3) \% 11 = 9$

$= (6+4) \% 11 = 10$

$11 = (2*11+5) \% 11 = 5$

$= (5+1) \% 11 = 6$

$= (5+2) \% 11 = 7$

$= (5+3) \% 11 = 8$

$= (5+4) \% 11 = 9$

$= (5+5) \% 11 = 10$

$= (5+6) \% 11 = 0$

$39 = (39*2+5) \% 11 = 6$

$= (6+6) \% 11 = 1$

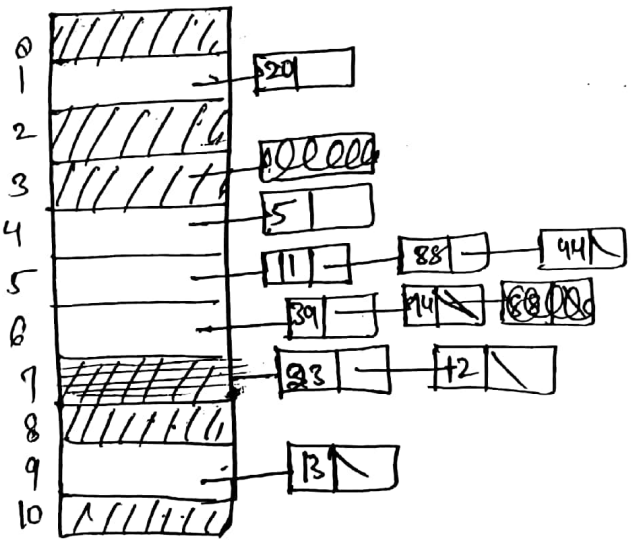
$20 = (2*20+5) \% 11 = 1$

$16 = (16*2+5) \% 11 = 4$

$5 = (5*2+5) \% 11 = 4$

39 location → 1

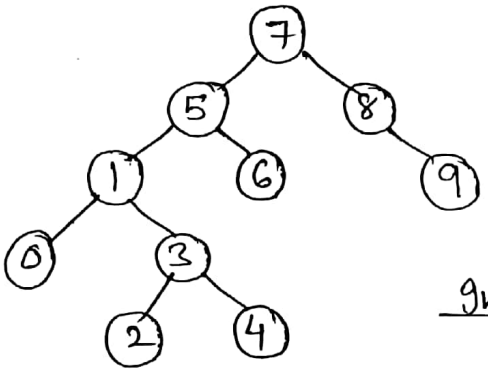
(53)



5 locations left.

(4) Trees

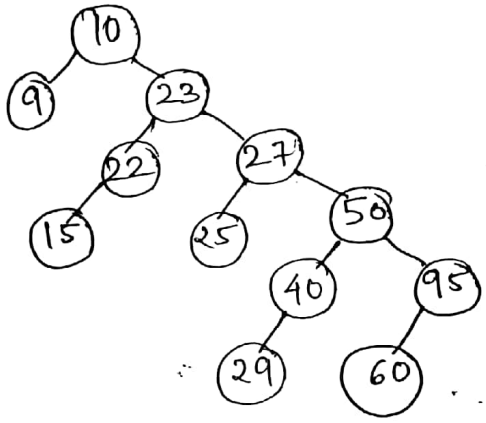
(4) 7, 5, 1, 8, 3, 6, 0, 9, 4, 2.



(c)

Inorder: 0 1 2 3 4 5 6 7 8 9

(9) 10, 9, 23, 22, 27, 25, 15, 50, 95, 60, 40, 29



inorder: 9, 10, 15, 22, 23, 25, 27, 29, 40, 50, 60, 95

(a)

(10) Postorder: B - A - C
Left Right Root

Preorder: C B A Q

definite preorder not possible
ans: (d) ✓

(13) inorder: d b e a f c g
Preorder: a b d e c f g
Postorder: d e b f g c a (a)