Python Date

Week ⇒ 2

⇒ defination (work same as function)

Ex:- def name(m, n)

≡

≡

→ range(1, m+1) take value from 1 to m
Mainly used in loop

→ append :- to add content. Ex:- em.append(i)

⇒ if in in stringname :- to find word in string

⇒ CF=[] to declare a list
min(n, m)

⇒ from filename import * (to include already written code)

- to find type of string type(variblename)
- to denote last element in array use (-1) element
- len(varible) to get string length
- output of s[3:1] is empty because Python not work in backward not given ~~error~~ error only · empty string
- output ~~of~~ if max. no. is out of range then it output the upto string & don't give error s=hello s[1:15] output is ello
- cannot replace a single character in string by s[3]='p'
- string are immutable mean cannot change the characters value of string

Date _____ Week ⇒ 3

Week ⇒ 3

⇒ List :- List can include no, ~~characters~~ string or value & can be updated

- List is Mutable

- nested list :- List ~~or~~ value contain list.

nest = [[2,[47]], 4, ["hello"]]

nest[2][0][2] is "l"

nest[0][1:2] is [[47]]

x = 5 [immutable]

y = x

x = 7

here y value 5 because assignment operator a fresh copy of a value

- in list if copy a list & change in 2 list will effect change in 1st list

- if copy a list then 2 name Point to one list

- in slice create a new list with some value of list.

l[:] is equivalent to l[0:len(l)] means full string

eg l = [1,2,4,7]

k = l[:]

l[2] = 8

then change in l does not effect k list.

Date _____.

- x == y check x & y has same value or not

but "x is y" check x & y Point to same object or not

list 1 = [1,2]

list 2 = [1,2]

list 3 = list2

list 1 == list 2 True list 1 is list2 False

list 2 == list 3 True list 2 is list3 True

- we can combine 2 list

list 3 = list 1 + list 2

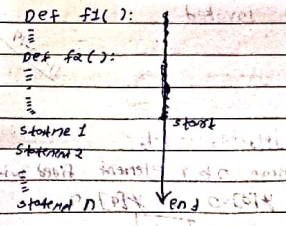
- list 1 = [1,2]

list 2 = list 1

list 1 = list 1 + [4]

then list 1 & list 2 no longer Point to same object

⇒ control flow :-



here execution start from statement 1 from top to bottom but destination of function does not execute it only execute on it use

Date

- control flow :- order in which statement are executed
  - i) condition execution
  - ii) Repeated execution - loops
  - iii) function definitions

i) conditional execution :-

- numerical value 0, empty ~~string~~ sequence "", [] is treated as false other wise all True
- here for use else & if use at anthor then "elif :" is used

ii) Repeated execution :-

for i in [1,2,..n]: for fix no. of sequence

while condition : for don't know no of sequence

iii) function definition :-

- if mutable value (list) is pass as argument then orginal value is changed.
- Define a function before it invoked

11-09-19

⇒ range()

- range (i,j) mean sequence from i, i+1, i+2, ... j-1,

ex:- ~~y=[] #range(~~ y = list (range(0,9)) mean 0 to 9 element filled with value 0 to 9 y[0]=0 & y[9]=9

- range (j) Mean sequence from zero to [j-1]

Date

- range (i,j,k) here sequence from i to ~~k~~ j where increment is k exp :- i, i+k, i+2k, ... i+nk
- default value of k is 1

i+nk < j <= i+(n+1)k

- range (i,j,~~k~~) :- here i > j produce i, i-2, i-4, ..., j+1
- here on increment i does not cross j
- if k ~~k~~ = +ve & i >= j, empty sequence because of infinite —"— -ve —"— i <= j —"—
- if k = -ve step "before" j
  exp :- range (12, 7, -3) produce 12, 9, 6, 3
- range (0, len(l)) produce till all element of ~~array~~ l
  value of l ⇒ 0 to len(l) - 1 range = 0 to len(l) - 1
- range (0, 10) is not equal to [0,1,2,3,4,5,6,7,8,9] for python 3 for python 2 it is true
- list(range(0,8)) convert range into list
- list 1 = [1,3,5,6] if we want new list with [2] = 5 then

list2 = list1 [0:2] + [7] + list1 [3:] here no. 2 list now to some list

Date

- list.append(value) :- to add new element at end (one value at a time)
- list1.extent(list2) :- to concate 2 list & store in list1
  list2 is list like list1 = list1 + list2
- list1.remove(x) :- remove x value from list & give error is no x in list
  - if more then one x is there first x is only remove from begining
- list.append(x) :- here list is object, append() is function, x is value.
- Slice is part of list list1[3:0]

  list1 = [1,3,5,7]
  list1[2:] = [9,10,11]
  list1 = [1,3,9,10,11]
  list1[0,2] = [7]
  list1 = [7, 9, 10, 11]

- if x in l:- here if x element present in list l then if
  l.remove(x) part is executed
  remove 1 x
  while x in l:
  remove all x l.remove(x)
- l.reverse() :- reverse sequence of list
  l.sort() :- sort the list
  l.index(x) :- find leftmost position of x in list l & give error is no x in list

Date

x l.rindex(x) :- find the rightmost position of value x in list

- y = x + y, gives error if x is not assignment
- for create a variable as list x = []

⇒ loop :-

- for :- if we know the no of iteration or used a list
- while :- if we dont know no. of iteration

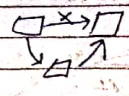
- (found, i) = (False, 0) here found = "False" & i = 0
- if condition1 and condition :- 2 condition combine by and

```
if cond :
    ≡
else :      elif condi :
    ≡   or      ≡
```

⇒ array :- • insertion & deletion is difficult in array as we shift every element below it

List :- list element are pointed to it next element not in next memory location so insertion & deletion is easy

plumbing meon :- add new element in list



- exchange value is easy in array but difficult in list
- deleting —"— list —"— array

Date

- time/taken / ~~binary~~ sequesial search mean ~~from~~ compare from start to end time taken by array & list is same for unsorted

- (for sorted) :- go to middle value if find stop search if value is less than then search left side other wise right side it is called binary search
  - only work on array

⇒ efficiency :-

* for Linear Scan is $O(n)$ for arrays & lists
* for Binary Search is $O(\log n)$ for sorted array

* Python can do about 10^7 steps in a second

*

⇒ ~~insertion~~ sorting sort :-

Searching of a value :- unsort array → $O(n)$
sort array → $O(\log n)$

by sorting :-
i) easy to find middle value
ii) also check duplicate

74 32 89 55 21 64
21 32 55 64 74 89 — selection sort

Date

74	32	89	55	21	64
21	32	89	55	74	64
21	32	55	89	74	64
21	32	55	64	74	89

— selection sort

Selection sort ⇒ $O(n^2) \rightarrow n + (n-1) + (n-2) \cdots 1 \rightarrow \frac{n^2+n}{2} = n^2$

⇒ insertion sort →

74	32	89	55	21	64
32	74	89	55	21	64
32	74	89	55	21	64
32	55	74	89	21	64
21	32	55	74	89	64
21	32	55	64	74	89

— insertion sort

insertion sort → $1 + 2 + \cdots (n-1) = \frac{n(n-1)}{2} = O(n^2)$

⇒ recursive

```
def sum(l):-
    if l == []:
        return(0)
    else:
        return (l[0] + sum(l[1:]))
```

Date Week ⇒ 4

insertion sort :- maximum recursion depth exceeded mean how Merge Process is Pause in execution the output

* by default Python has Maximum recursion is 998

to set recursion limit :-

import sys

sys.setrecursionlimit (100000)

Week ⇒ 4

⇒ Merge sort :-

Sort A [0 : n//2] # here n//2 = divide & take only integer Part.

Sort A [n//2 : n]

43, 32, 22, 78

[43|32] [22|78]

[43] [32] [22] [78]

[22|43] [22|78]

[22|32|43|78]

i) only 2 list is empty.

ii) if non empty comparing 1 by 1

i) check no. of element < 4

ii) if not, divide A in 2 Part

L = A [0 : n//2]

R = A [n//2 : n]

if function (A, L, R)

if (RL <= 1)

return (A[L:R])

L = Left index R = High index

Date

def mergesort (A[], L, R)

if (r - L > 1)

Mid = (L+R)/2

L = function (A, L, Mid)

R = function (A, Mid, right)

return (Merge(A, L, R))

Merge function combine 2 list

def merge (A, B):

(C, m, n) = ([], len(A), len(B))

(i, j, k) = (0, 0, 0)

while i+j < m+n :

if i == m :

C.append (B[j])

j = j+1

if j == n :

C.append (A[i])

i = i+1

if A[i] <= B[j]

C.append (A[i])

i = i+1

if [A[i] >= B[j]

C.append (B[j])

j = j+1

return (i)

Index Comma Position of A & B

(May also use

while m > i && n > j :

if A[i] > B[j]

{ arr[k] = A[i]

i++ ;

else : arr[k] = B[j]

j++

k++

while m > i :

{ arr[k] = A[i]

k++; i++; }

while n > j :

{ arr[k] = B[j]

j++; k++; }

return (arr[])

efficiency of Merge sort is (Max(m, n)) =

T(n) = O(n log n)

Date

⇒ Quick sort

i) Choose a Pivot element
ii) divide list w.r.t. to Pivot in 2 half

worst case is already sorted array that is $O(n^2)$
average case is $O(n \log n)$

- Quick sort is not stable because swap operation during partitioning disturbs original order

- Merge sort is stable because
  i) do not allow element from right to overtake element from left

⇒ Tuples :-
ex:- (age, name) = (23, "Hari") or Date (16, 7, 2013)

Tuples are immutable Means do not change it value

→ Dictionaries
test1 ["Dhawan"] = 84 , test1 ["Kohli"] = 56 ,
(key, value)

Dictionaries = {} empty
Tuple = ( ) empty tuple

Key can be any immutable value - int, float, bool, string, tuple
key can not be list or dictionaries

Date

Dictionaries of Dictionaries

- Score ["Test1"] ["Dhawan"] = 84 ⇒ List
  Score = {"Test1": {"Dhawan": 84, ..}} ⇒ dictionaries

first declare outer Dictionaries then inner dict. then assign value
score = {}
score ["Test1"] = {}
score ["Test1"] ["Dhawan"] = 76 ;

- d.keys() return sequence of key

  for k in d.keys()

- to get sorted key :-
  for k in sorted (d.keys()):

- d.values() is sequence of value in d
  total = 0

- d = {}
  d[0] = 7 Mean d = {0:7}

- Dictionary allow a flexible association of values to keys

⇒ function :-

⇒ Default argument

Def func (a=10, b=15) here if we call func with no parameter then 10 & 5 is used

Date

```
def f(a, b, c):
    ≡
g = f
g(a, b, c) x        here f function name as g
```

⇒ list comprehension :-

→ filter (P, l) where P is Property & l is list
here filter return sublist of element that satisfied the P Property of list l

→ list(map(f, l)) → here give f as function & l is list
here each element goes to function & add to result list but map(f, l) does not give list

ex :- [square(i) for i in range(100) if iseven(x)]

- square(i) → mapper → power even value
- for i in range(100) → generator → get 0 – 100
- if iseven(x) → filter → get even value

for Pythagorous theorem

[(x, y, z) for i in range(100) for j in range(100) for k in range(100) if $x^2 + y^2 == z^2$]

Date Week=5

⇒ exception handling

Syntax →

```
try:
    ≡ code
except IndexError:
    ≡ action on error
except (NameError, KeyError):
    ≡ action for all listed errors
except:
    ≡ action for all error
else:
    ≡ ← if try is not execute
```

- add 's'

```
try
    scores[6].append(s)
except KeyError:
    scores[6] = [s]
```

→ Type of error :-

i) SyntaxError
ii) NameError
iii) ZeroDivisionError
iv) IndexError
v) ValueError ⇒ when convert english alphabet into integer

→ for avoid "\n" on Print at end use end="\n"

Print("I am superman", end="")

Date

→ on use Print("a","10") these is space bet a & 10 to avoid space use sep="..." as end
ex:- Print("a","10",sep="")

⇒ Print "a" is used in Python 2
but Print("a") is used in Python 3

⇒ storage file is read is more slowly than memory

⇒ file handling

hvariable = open("Path","mode") → to connect to file
string variable = hvariable.read() → to read file (whole)

string variable = hvariable.readline() → to read 1 line

string variable = hvariable.readlines() → to get list which have each element as line of file

hvariable.seek(n) → move pointer to n position
hvariable.read(12) → to read fixed characters

for next line use "\n" externally in writing
hvariable.write(s) → to write into file
hvariable.writelines(s) → to list & write each element one by one

hvariable.close() → flush output buffer
hvariable.flush() → write data to file upto now

Print(Round(22/7, 5)) // Print result of 22/7 upto 5 decimal place.

Date

• White space are characters which not display like tab, space, \n

to remove white space in string use string.rstrip()
also remove from left side string.lstrip() & for both side use string.strip()

S.find(Pattern) → return starting position of pattern if pattern occur in S
if not found return -1

S.find(Pattern, start, end) → same as above but find in range (start, end)

S.index(Pattern) or S.index(Pattern, left, right) → same as above but raise ValueError if Pattern not find

S.replace(f1, f2) → return string in which f1 is replace by f2 in S

S.replace(f1, f2, n) → same as above but replace first n f1

joinstring = " "
joinstring.join(list) → join list by given character

S.lower() ⇒ convert all to lowercase
S.upper() ⇒ —"— uppercase
S.title() ⇒ convert uppercase first letter in each word
S.center(n) ⇒
S.isalpha() ⇒ return bool on if S has alphabet or not
S.isnumeric() ⇒ —"— numeric —"—

Date _____ Week-6

⇒ format()

a = "first {0} seond {1}".format(47,11)

pass → to do nothing

→ function related to list :-

i) sum(list1)
ii) list1.extend(list2)
iii) list1.count(1) count occurance of 1
iv) list.index(a) get index
v) min(list) / max(list)
vi) sorted(list)
vii) list.sort(reverse = True)
viii) list.pop(1) delete element by index if not given remove last
ix) del list[2] delete element by index
x) list.remove(0) —)) ——— value

⇒ back tracking Week-6

for N queen Problem on chess for NxN board

Place N queen as non of them is attack on each other

- 2 & 3 is not possible

if sol^n not found go back to previous step & change it

Date _____

⇒ for global scope :-

global x

⇒ Set

colour = set() declaration

colour = {'red', 'black', 'red', 'green'}

Print(colours) → {'black', 'red', 'green'} remove duplicate

- 'black' in colours this give true

- convert list into set

no. = set([0,1,3,3,4,1])

set has {0,1,3,4}

- letters = set('banana')

letters has {'a', 'n', 'b'}

- odd = set([1,3,5])
Prime = set([2,3,5])

Union odd | Prime → {1,2,3,5} ⇒ union

& ⇒ for intersection

odd - Prime → difference

odd ^ Prime → {1,2} → exclusive (remove common item)

Date

→ Queue are useful for breadth-first exploration

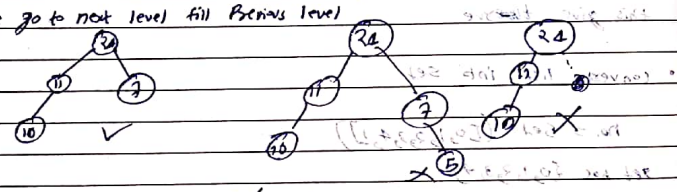
⇒ Priority queue :-

delete_max() → delete of high Priority

insert() → insert.

→ balance tree is called heap

- Parent value is greater than child
- child filled from left to right
- before go to next level fill previous level



heap insert take O(log N) time

for delete root Move up to bottom

Date Week-7

Behaviour of stack ⇒ Push() & Pop()

queue ⇒ add() & removeq()

Heap ⇒ insert() & delete_max()

- if Heap implement list this not mean list.append(5) is legal in Heap

⇒ class :-

int a : setting current object

every function in class → def functionname(self, Parameter):

self.a = 10

Constructor __init__()

- __str__() return string as object value

str(a) = a.__str__() both are same

- p1+p2 = p1.__add__(p2) it will return add both value
- p1.p2 = p1.__mult__()

```
def append(self, v):
    if self.isempty():
        self.value = v
    elif self.next == None:
        newnode = Node(v)
        self.next = newnode
    else:
        self.next.append(v)
    return()
```

Date

⇒ Dictionary :-

- to get all keys as list then dict.keys()
- —//— values —//— dict.values()
- —//— keys & values as pair then dict.items()

```
• for x in my_dict:
      Print(x)      // Print all keys
• for y in my_dict.values():
      Print(y)      // Print all values
• for z in my_dict.items():
      Print(x, ":", y)    // Print key-value pair
```

```
• My_dict ['Dave"] = '004'  //if Dave already exist then update otherwise creat key-value pair
• del My_dict['Dave']       // remove key-pair value
• My_dict.pop('Ava')        // remove 'Ava' value
• My_dict.popitem()         // remove last inserted item
```

⇒ Pointer in C / C++ :-

```
• int Var=20;
  int *IP;
  IP= &Var;
  cout<< &Var;      /* Print address of var variable */
  cout << IP;       /* Print address contain by IP Point */
  cout << *IP;      /* Value store in address space */
  * IP= NULL;
  cout << IP;       /* Print "0" by default
```

Date

*Ptr :- Point that contain address of variable
**ptr :- Point that contain address of another Pointer.

```
• int nums[2][3] = {{16,18,20},{25,26,27}};
  nums = &nums[0][0]
```

	no. Print.
*(*nums)	16
*(*nums+1)	18
*(*nums+2)	20
((nums+1))	25
((nums+1)+1)	26
((nums+1)+2)	27

```
•Pointer in linked list :-

node_t * head = NULL          Creat node Point of LL
head = (node_t*) Malloc (sizeof (node_t));  // index address
head -> val = 1;              // Put value in first list
head -> next = (node_t*) Malloc (sizeof (node_t)); // another list is added
head -> next -> val = 2;      // Put value in second list
head -> next -> next = NULL   // next Point to second set to NULL
```

• To know the height of the binary Tree :-

```
Int Height (CurrentNode)
{
 if (CurrentNode -> Next == NULL)
   { return -1 }
 LeftNode = 1 + Height (currentNode -> LNode)
 RightNode = Height (current Node -> RNode)
```

Date

return (max (LNode, RNode) + 1)
}

⇒ Quicksorting algo :-

```
Quick (List)
{ if (len (List) > 1)
   { Pi = List [len (List) - 1]
    for i in List :
    if (i > Pi) :
        RList = RList + [i]
    else :
        LList = LList + [i]
    Rfinlist = Quick (RList)
    Lfinlist = Quick (LList)
    finlist = Rfinlist[] + [Pi] + Lfinlist [ ]
    return(finlist)
}
```

⇒ Mergesorting algo :-

Date

Aptitude :-

Date

1) Divisibility 2) HCF & LCM 3) Profit & loss
4) simple & compound interest 5) Time, speed & Distance
6) Inverse 7) log 8) Permutation & combination
9) Probability

1) Divisibility :-

2 → last digit is 0, 2, 4, 6, 8
3 → sum of digit is divisible by 3
4 → last 2 digit is divisible by 4
5 → last digit is 0 or 5
6 → if given no. divisible by 2 & 3
7 → subtract the twise the last digit with remaining digit give a multiple of 7
ex:- 658 → 65 − 2×8 = 49 so 658 is divisble by 7
8 → last 3 digit is divisible by 8
9 → if sum of digit is divisble by 9
10 → if last digit is 0
11 → different bet the sum of alternet digit is multiple of 11
ex:- 2343 ⇒ (2+4) − 3+3 = 0 ; so 2343 is divisible of 11
12 → if given no. is divisible by 3 & 4

- if we divisible a given no. by even no. then answer is also a even no

13 → 4 time the unit digit is add to remaining no. (remove unit digit) is divisible by 13
45238 ⇒ 4523 + 4×8 = 4555 ⇒ 455 + 4×5 ⇒ 475 ⇒ 47 + 4×5 = 67
67 is not divisible by 13

Date

17 ⇒ 5 time the last digit subtract from remaining no. is divisible by 17
2278 ⇒ 227 − 8×5 = 187 , so 2278 is divisible by 17

19 ⇒ same case as 17 but 2 times adding so resultant is divisible by 19

~~23 / HEF & LCM~~
23 ⇒ same case as 17 but 7 times.

29 ⇒ same case as 23 but 3 times

2) HCF & LCM

Product of 2 NO = LCM × HCF

LCM of (20, 30) = 60
HCF of (20, 30) = 10

Date

SQL languages :-

1) 4 type of languages :-
i) Data Defination Languages :- define the database
example :- Create, Drop, Alter, Truncate

ii) Data Manipulation languages :- Manipulate the data present in DB
example :- insert, update, delete

iii) Data control languages :- deal with user Permission & control of DB system. example :- Grant, Revoke

iv) Transaction control languages :- deal with the traction of DB
example :- commit, Rollback

v) Data query language :- used to fetch data from DB
example :- select

DDL

2) Comment --(select all (single line) /* */ (multiple line comment)

3) Create Table Tname AS Select C1, C2 from Emp

4) Truncate table Tname ; // delete all info. of table but not table

5) Alter table Tname ADD C1 Int ; // use to add, delete & Modify columns
ALTER C1 char ;

6) type of keys :- there are 7 type of key

Date

i) candidate key :- all those colomb whose tuple are uniquly identify. Mean. all the eligible Primay key are candidate key. a table has more than 1 candidate key but Primary key is one

ii) Primary key :- all tuple of Primary key uniquly identify only one coulme is know as Primay key

iii) usuper key :- set of all attributes that uniquly identify one tuple, include candidate key, Primary key & unique key

iv) unique key :- ~~candidate~~ Candidate key colomn - Primary key colomn all attributes that uniquly identify except Primary key & allow one NULL value in colomb

v) attribute ~~unique~~ key :- Candidate key - Primary key. all candidate key except Primary key & not allow NULL value

vi) foreign key :- colomb that are common in tables

vii) Composite key :- combination of 2 or more colomb to identify each colomb Uniquely.

~~viii) NULL NOT NULL :-~~

ix) Constraints :- these are used to specify the rules for data in a table

Date

i) NOT NULL :- that colomn does not contain null value
ex:- create Table hr (e_id int NOT NULL);

ii) UNIQUE :- ensure all value of colomn is unique
ex:- create Table hr (e_id int NOT NULL UNIQUE);

iii) CHECK :- all value in colomn satisfy a specific condition
ex:- create Table hr (e_id char(225) CHECK (country = 'India'));

iv) default :- if no value is given then Default value is stored
ex:- create Table hr (e_id char(225) DEFAULT 'India');

v) INDEX :- This constraints is used to create index in table, by which we can easily create & retrive data from DB
ex:- CREATE INDEX id ON TableName (colomn_1, 2);

DDL

DML

8) USE :- select the DB on which work is done
Ex:- USE DBName;

9) INSERT INTO :- insert new record in the table
ex:- Insert INTO Tablename (c1, c2, c3) Value (v1, v2, v3);

10) UPDATE :- Modify the already present record.
ex:- Update TableName set c1 = v2, c2 = v3 WHERE condition;

11) DELETE :- delete the existing record.
ex:- DELETE FROM Tablename Where E_Id = 2;

Date

12) DELETE SELECT :- select data from DB & store in a table know as result-set

ex:- select E_ID, E_Name from E_Info ;

These are 5 keyword which are used with SELECT

i) ex:- SELECT DISTINCT C1, C2 from TName ;

SELECT DISTINCT :- return only different value

ii) ORDER ORDER BY :- used to sort the required result. by default it is ~~ascending~~ ascending order. used ASC & DESC

ex:- select * FROM E_INFO ODER BY E_ID ;

~~Sel~~ " ——— ASC/DESC ;

iii) GROUP BY :- Used to group result set with aggregate function.

ex:- SELECT COUNT (E_ID), city From E_Info GROUP By City ;

iv) "Having" :- ~~Having~~ used where "WHERE" is not used.

ex:- SELECT COUNT (E_ID), city FROM E_Info GROUP By city HAVING COUNT (E_ID) > 2 ORDER By COUNT (E_ID) DESC;

v) SELECT INTO :- Used to copy data one table to another.

ex:- SELECT * INTO E_backup FROM Emp ;

New table [in external DB] ← old table

Date

13) Operator in SQL :-

i) Arithmetic Operators :- %, (modulus for reminder), /, *, +, -

ii) Bitwise operators :- i) "^" (XOR operation) $(A\bar{B}+\bar{A}B)$, $A \wedge B$
ii) "|" (OR operation), $A \mid B$
iii) "&" (AND operation), $A \& B$

iii) Comparison Operators / Relational operator :- i) <> (not equal to), $A <> B$
ii) <= (less than or equal to), $A <= B$
iii) >= (greater than or equal to), $A >= B$
iv) < (less than), $A < B$
v) > (greater than) $A > B$
vi) = (equal to), $A = B$

iv) Compound Operator / Assignment operator :- i) "|*=" (Bitwise OR equals), $A \mid *= B$
ii) ^= (Bitwise Exclusive equals, $A \wedge = B$
iii) &= (Bitwise AND equals, $A \& = B$
iv) %= (Modulo equals), $A \%= B$
v) /= (Divide equals), $A /= B$
vi) *= (Multiply equals), $A *= B$
vii) -= (Subtract equals, $A -= B$
viii) += (Add equals), $A += B$

v) Logical operator :- ~~AND~~ AND, OR, NOT, BETWEEN, LIKE, IN, EXISTS, ALL, ANY.

a) AND Operator :- filter records that fulfill ~~the~~ all required condition.

ex:- SELECT * FROM E_Table ~~the~~ WHERE City = "GOA" AND Category = "ST" ;

Date

b) OR Operator :- filter record which satisfied any condition
ex:- SELECT * FROM E_Table WHERE City = "Goa" OR City = "Mumbai" ;

c) NOT Operator :- display record which does not satisfy the condition.
ex:- SELECT * FROM E_Table WHERE NOT City = 'Goa';

all combine query :- SELECT * FROM E_Table WHERE NOT City = 'Goa' AND (City = "Mumbai" OR City = "Banglore") ;

d) BETWEEN Operator :- used when you wan to selet value within a range including both value
ex:- SELECT * FROM E_Table WHERE Salary BETWEEN 40000 AND 50000 ;

e) LIKE Operator :- search for a specific Pattern
i) % :- matches 0 or more character
ii) _ :- matches exactly one character.
ex:- WHERE Custome_Name LIKE 'V%' :- C_Name start with V
- ——" ——— '%and%' :- find value that have "and" at any Position
- ——" ——— '_a%' :- value that have "a" at second Position
- ——" ——— "V_%_%' :- value start with 'V' & have atleast 3 character
- ——" ——— "m %a" :- value start with m & end with a

f) IN Operator :- used for multiple OR condition
ex:- select * FROM E_Table WHERE city IN ('Mumbai', 'Goa', 'banglore') ;

g) EXISTS Operator :- test if a record exist or Not.
ex:- select cy From E_Table WHERE EXISTS (SELECT cy From E_Table WHERE E_Id = 05 AND City = 'Kolkata') ;

h) ANY/ALL Operator :- used with WHERE OR HAVING clause return true if all of the subquery values meet the condition.
ex:- Select E_Name from E_Table WHERE E_ID = ALL (SELECT E_ID from E-Table where city = 'Goa') ;

~~ANY Operator:~~

14) Aggregate function :-

i) MIN() :- return smallest value of selected colovm.
ex:- Selelt MIN (E_ID) FROM E_Table ;

ii) MAX() :- return ~~Maxim~~ largest value of select colov

iii) COUNT() :- return no of rows which match a specified criteria
ex:- SELECT COUNT (E_ID) FROM E_Table ;

iv) SUM() :- Sum of a numeric column that you ch
ex:- SELECT SUM (SALARY) FROM E_Table ;

Date

v) AVG() :- return the average value of a select numeric column

ex:- SELECT AVG (Salary) FROM E_Table;

15) NULL Function :- return an alternative value if an expression is NULL . ISNULL()

ex:- SELECT E-ID * (Month_Year_of_Salary + ISNULL(Salary,0)) FROM E_Table;

16) Aliases :- aliases give a column / table a temporary name & only exists for a duration of query

ex:- SELECT E_ID AS ID, E_Name AS Name FROM E_Table;

17) Case :- Go through all condition & return a value when first condition is met. so if no condition is TRUE then return else part. if there is no else part then return "NULL"

ex:- select E_Name, city FROM E_Table ORDRED by
(CaseE
WHEN city IS NULL THEN 'county is India by Default'
ELSE city
END);

18) JOIN :- Used to combine row from 2 or more Tables on bases on related column

There are of 4 type :-

Date

i) INNER JOIN :- Join all those record which (A∩B) have matching values in both table

ex:- select Tech. TechID, E_Table. E_Name from Tech. INNER JOIN E_Table ON Tech. E-ID = E_Table. E_ID;

ii) FULL JOIN (A∪B) :- Join all record which either have a Match in left or right table

ex:- select E_Table. E_Name, Tech. TechID From E_Table FULL OUTER JOIN Tech ON Tech. E-ID = E_Table. E_ID;

iii) LEFT JOIN (A + A∩B) :- return all record from left table along with record satisfied condition from righ table

iv) RIGHT JOIN (A∩B + B) :- return all record from right table along with record satisfied condition from left table

19) Set operation :- UNION , INTERSECT, EXCEPT

i) UNION :- Used to combine the result set of 2 or More statement.

ex:- select ColumnName from Table1
UNION
SELECT ColumnName from Table2

Date

ii) INTERSECT :- used to show common in two result set.

ex:- Select C_1, C_2 from TName where condition
INTERSECT
Select C_1, C_2 from TName1 where condition

iii) Except :- return tuple that present in first select statement except those who are present in second select statement

ex:- SELECT Colum FROM TName1
EXCEPT
SELECT Colum FROM TName2

20) Dates :- follow datatype store date/time value in DB
i) DATE :- YYYY-MM-DD
ii) DATETIME :- YYYY-MM-DD HH:MI:SS
iii) TIMP
iii) TIMESTAMP :- unique number

ex:- select * from TName where T_Date='2019-04-01';

21) Auto Increment :- generate a new unique no. atuomatically when new record is insert
INDENTITY Key word is used.

ex:- CREATE TABLE Employee_info_table (E_ID INT IDENTITY(1,1) PRIMARY Key, E_Name VARCHAR (255) NOT NULL);

DML

DCL

Date

22) GRANT :- Provide access or Privileges on the DB & its object to users

ex:- GRANT PrivilegesName ON objectName To {USERname | Public | RoleName} [With GRANT option]
;

i) PrivilegesName :- name of the right/access/Privilege grant to user

ii) objectName :- Name of DB object like Table/view /STORED.

iii) USERName :- Name of user who give access

iv) Public :- To grant access to all users

v) RoleName :- name of a set of Privileges grouped together

vi) With GRANT option :- option of Grant

ex:- GRANT SELET ON E_Table To User1;

23) REVOKE :- Withdraw the user's access Privileges

ex:- REVOKE PrivtegesNames ON objectName
FROM {USERName | PUBLIC | RoleName}

Date

24) Views :- it is a single table. which is derived from other tables. so it has field from one

i) CREATE VIEW :- Create a view from table

ex:- SELECT E_Name, E_Phone, FROM E_Table WHERE CITY = "MUMBAI";

ii) "CREATE OR REPLACE VIEW" :- used to Update a view

ex:- CREATE VIEW OR REPLACE [View1] AS SELECT E_Name, E_Phone FROM E_Table where City = "Mumbai";

iii) "DROP" view :- used to delete a view

ex:- DROP VIEW [view1]

25) Stored Procedures :- code which you save & reuse it again.

ex:- CREATE PROCEDURE PName AS SQLStatement GO;

26) Triggers :- Triggers are set of SQL Statement which are stored in DB catalog. the statement execute when a event occur on table so, a trigger can be INVOKED either BEFORE or AFTER the Data changed by Insert, Update or delete statement.

before insert → after insert → before Update → after Update
after delete ← before delete ←

Date

ex:- CREATE TRIGGER [TriggerName] [Before | After] {INSert | UPDate | delete} on [Tablename] [FOR EACH ROW] [Trigger Body]

DCL

TCL

27) COMMIT :- used to save the transaction into the DB

ex:- COMMIT;

28) ROLLBACK :- used to restore the DB to the last committed state.

ex:- ROLLBACK;

ex:- ROLL BACK To SavePoint Name;

29) SAVEPOINT :- Used to temporarily save a transaction. so if we wish to rollback to any Point the save that Point.

ex:- SAVEPOINT SavePointname;

ex:- INSERT INTO E_Table VALVES (05, 'Avinash');
COMMIT;
UPDATE E_Table SET name = 'Akash' WHERE id = '05';
SavePoint S1;
INSERT INTO E-Table Value (08, 'veena');
SELECT * FROM E-Table;
ROLLBack To S1;

TCL

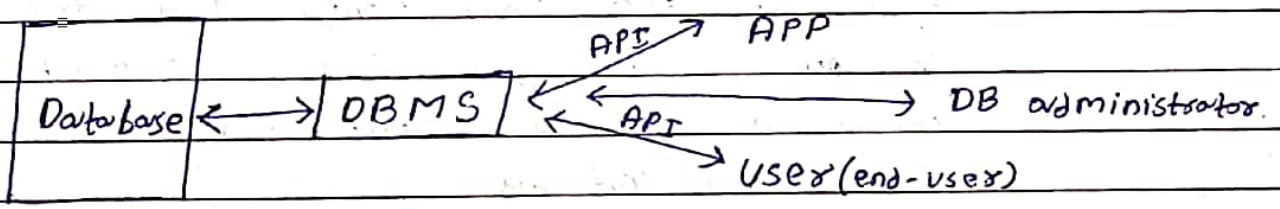
Data Base Management System.

Date

⇒ Basic Concept, three-tier architecture, ER Model functional dependencies, normalization, relation Model, stored Procedure & triggers, locking, Clustering.

DB & DBMS :- i) DBMS is a system software for creating & Managing DB.

ii) DBMS Provide a systematic way to handle DB by the request Provided throught APP API.



Feature of DBMS :-

i) Data Availability :- Data is ready to available whenever required by the user

ii) Data integrity & security :- Protect the existence of DB and Maintain the quality of database.

iii) Data independency :- 2 type - Physical data independence
logical data independence

a) Physical data independence :- which remain unaffected from changes in storage structure & access method

b) logical data independence :- which remain unaffected from change in schema

Date

⇒ ACID Properties :- (Atomicity, Consistency, Isolation, Durability)

i) Atomicity :- There is only 2 state for a transaction either commits or abort. if we update a database the all update will commits or non of update commits (All or nothing rule)

ii) Consistency :- it refer to the correctness of DB if we transfer money from one account to other then after deduct money from first account system goes down & amount does not credit in second account then this is called inconsistent DB

iii) isolation :- this property ensures that multiple transaction can occur concurrently without leading the inconsistency of DB state

iv) Durability :- this property ensure that if a transaction is commits then it must be update in DB even when system or the storage media fail.
Recovery to the most recent sucessful commit.

Database :- i) Database is a collection of related data
ii) data which represent some aspect of real world
iii) A DB is designed to be built & populated with data for a specific type

Date

⇒ Characteristics of DBMS :-
i) Provide security & removes redundancy
ii) support of multiple views of data
iii) follow ACID Property
iv) supports multi-user environment

Ex of Popular DBMS Software :- mysql, sqlite, dBASE

Type of DBMS System :-
i) Hierarchical DBMS :- a) organised in a tree-like structure
b) Data is represent using Parent-child relationship
c) Parent have many child but child have only one Parent

ii) Network Model :- a) allow all child to have multiple Parent. (many to many relation)
b) used to address more complex relationship

iii) Relation Model :- a) most widely used DBMS Model
b) based on normalizing data in rows & columns.
c) having fixed structure & Manipulating using SQL

iv) Object-oriented Model :- a) stored data in form of object

Date

⇒ 3-Tier architecture :- (Presentation layer, Application layer, Data layer)
HTML, CSS, JS — Java, .Net, Python ft — MySQL, Oracle, SQL server

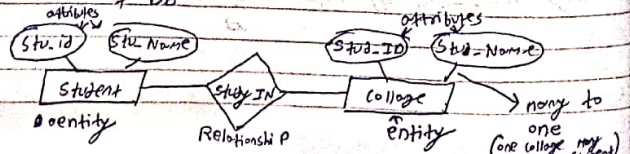
Q. 3-Tier application architecture is client-server architecture consist of 3 tier. Data store at Data layer, logic handles in application layer, GUI in Presentation layer.

benefit ⇒ i) each tier can be developed concurrently without effecting other tier.
ii) easier for enterprise to continously evolve an application as new need.

⇒ ER Model :- i) Entity - relationship Model is a design or blueprint of DB that can later be implemented as a DB
ii) 2 component - entity set & relationship set
iii) ER Model desribe the structure of DB with the help of diagram know as Entity - relationship diagram (ER diagram

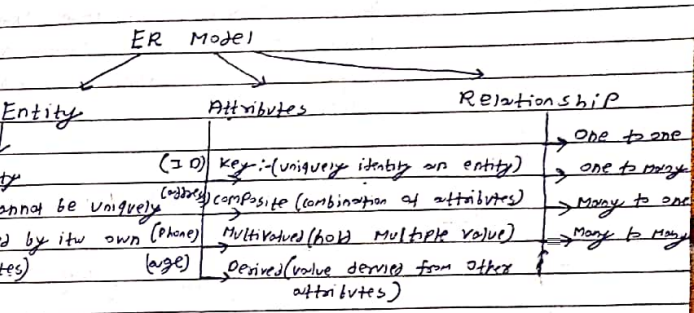
⇒ ER Diagram :- i) an entity set is attribute of table in DB
ii) by showing relation among tables & their attributes, ER diagram shows logical structure of DB

example :-



Date

⇒ Component of ER Diagram :-



⇒ functional dependency :- it is a relationship between 2 attributes typically between the primary key & other non-key attributes within a table.

determinant → dependent
X → Y

SIN → Name, Address, DoB [with sin we determine name, Address, DOB]

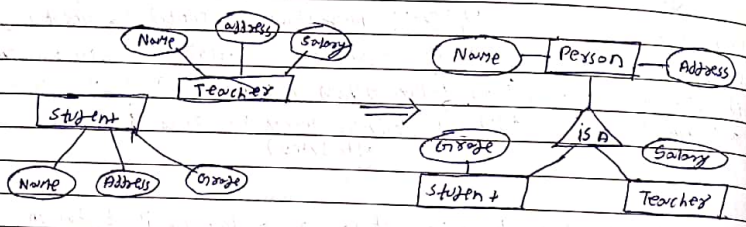
⇒ Normalization :- i) it is technique in which we are systematically approach of decomposing table to eliminate redundancy (duplicate)
ii) elimination of useless data & ensure data dependencies make sense i.e. data is logically stored.

type of Normal form ⇒ 1NF, 2NF, 3NF & Boyce-Codd NF

Date

⇒ Generalization :- It is a process in which the common attributes of more than one entity form a new entity & new entity form is called generalized entity.

before



⇒ Relational Model, Normal form, indexing (B & B+ tree)

⇒ DBMS, transaction, concurrency, file organisation

Relational algebra in Relation Model.

⇒ Normal forms :-

i) 1st Normal form :- • no tuple contain 2 value

ii) 2nd Normal form :- • must follow 1st Normal form
• there is No Partial Dependence.
• if 1 coloum depend on 2 coloum then it must be seperated in 2 different table

iii) 3rd Normal form :- • must follow 2nd Normal form
• there is no transitive dependence
• if 1 coloum is derived from 2 coloum indivially then it must be in 2 different table

Date

iv) Boyce-codd Normal Form :- • must follow 3rd NF
• all table in DB should be only one Primary key

v) 5th NF :-

C Language :-

Date

range of datatype (0-79, 0-24)

⇒ Data type in C language :-

- Datatype
  - Primary Data Type
    - Void
    - int (-32,768 to 32,767) — 16 bit
    - long (-2147 million to 2147 million)
    - char (0-9 ⇒ 48-57, A-Z ⇒ 65-90, a-z = 97-122) (allows -128 to 127)
    - float (3.4*10^-38 to 3.4*10^38)
    - Double (1.7*10^-308 to 1.7*10^308)
    - Short (range same as int)
  - Derived Data Type
    - Array
    - Reference
    - Pointer
  - User Defined Datatype
    - Structure
    - Union
    - Enumeration (-32768 to 32767)

- signed :- Means datatype act as normal datatype
- unsigned :- Means only range of no include. (for int unsigned 0 to 65,535)

- Union :- • allowing storing various data type in same memory location. •
  - Program can define a union with different member or datatype but only a single member can contain value at a given time.

Ex:-
```
union variablename      // same as structure definition
{ char x;
  float y;              // here sizeof(obj) => 4 byte as float occur 4 byte
} obj;                  // object size is size of largest size variable
```

- enum or enumeration :- • User define data type

Date

• enumeration or enum :- • used a assign name to integral constant, the names make a program easy to read & maintain
• keyword 'enum' is used to declare

example :- #include <studio.h>
enum week { Mon, Tue, wed, Thu, Fri, Sat, Sun };
int main()
{ enum week day; // week= ~~variable~~ datatype name
day= object name
day = wed;
Printf("%d", day); //output is 2
return 0;
}

• size of all datatype :-

format specific	datatype	size
%od	~~char~~ int	2 byte (when 16 bit Processor), 4 byte (when 32 bit or 64 bit)
%oC	char	1 byte
%ol	long int	4 byte (for 16 bit Processor), 8 byte (when 32 bit, 64 bit)
%of	float	4 byte
%lf	double	8 byte
	Void Pointer	2 byte (16 bit Processor), 4 byte (32 bit), 8 byte (64 bit)
	Shoot	2 byte

• Reference :- • When a variable is declared as reference, it becomes an alternative name for an existing variable.
• A variable can be declared as reference by Putting '&' in the declaration.
• 2 variable point to same memory location

int x=10;
int& ref = x; // now if we make any change in any one variable then change is also appear in other

Date

• Short :- • compact version of int variable
• range is -32768 to 32,767
• size is 2 byte for an unsigned, signed

• Pointer :- • it is variable that store address of another variable
• Pointer is used to allocate memory dynamically i.e. at runtime

⇒ Variable :- • Variable is a name given to a storage location that our program can manipulate

⇒ format specifier :- • it is use during input & output
• it is a way to tell compiler about the type of does a variable take or print during scanf and Printf

⇒ exit(0) ⇒ terminate normally successfully
exit(1) ⇒ —"— —"— unsuccessfully

⇒ operator in C :-
i) arithmetic operator :- used in mathematics operation (+, -, *, /, %)
ii) Relational operator :- IR Defined relation between 2 operand (==, !=, >, <, >=, <=)
iii) Logical operator :- used for multiple condition (!, &&, ||) OR AND NOT
iv) Bitwise operator :- perform manipulation of data at bit level
(& , | , ^ , << , >>)
Bitwise OR, AND, Exclusive OR, leftshift, right shift

Date

v) assign operators :- Used to assigned varible to a varible using other varible or numbers. (=, +=, -=, *=, /=, %=)

vi) conditional operator :-
- also know as **Teanary operator**
- Expression ? expression 1 : expression 2
- same work as if-else but way of writing is different

vii) special operator :- • size of()
- & ⇒ retrive address of a varible
- * ⇒ point to a varible

Loop :- like

⇒ NULL statement :- it is used in loop when there is no statement is loop body
ex:- while (for(int i=0; i>2; i++)
;

loop :- when we have to execute a block of statement for no. of times then we use loop
i) while loop (for unknow no. of iteration)
ii) for loop (for known no. of —)—)
iii) do while loop (for execute body of loop for atleast one time)

⇒ type conversion in C :- • conversion of one data type into other
- conversion is done where it is possible (int to char, int to float)

Date

type conversion is of 2 type
i) implicit Type conversion :- Perform by compiler automatic /automatic type conversion
ii) Explicit Type conversion :- Perform by user convert

• When a operation is done between 2 operation then lower preference operator converted to higher preference operator

lowest → highest
char → int → long → float → double → long double

⇒ type of varible in C :-

type	storage	initial value	scope	Life
automatic (auto)	stack	Garbage	within block	end of block
external (extern)	Data segment	zero	global	end of Program
static (static)	⊥)) —	zero	within block	—)) —
register (register)	CPU register	Garbage	—)) —	end of block
global	Data segment	zero	global	end of Program
local	stack	garbage	within block	end of block

- Static :- create only one copy if we do call a function multiple time then it use same varible

```
Ex  void function()
    {static int y=1;
     y=y+1
     Printf("%d",y);
    }
```
// if we call this function 3 times then output is 2,3,4 as y is only one copy

- automatic :- all varible declare in a block by default automatic
- external/global :- declare outside of all the block & can access of of part of Program

Date

• Local Variable :- local variable are variable that declare inside a block

• register Variable :- • it tell the compiler to store the variable in CPU register instead of memory
• frequently used variable are kept in register because of fast accessibility
• we can not get address of these variable

⇒ for string variable we use puts for i/p & gets for o/p

⇒ Macros :- • It is a fragment of code that is given a name. you can
• you can define a macros using #define
ex:- #define C 29987 ;
• here replace C with 29987 whenever we use C.

⇒ PreProcessor directives :- • it is responsible for transforms your Program before it is compiled.
• these transform can be inclusive of header file, macro expansion etc
• all PreProcessor directives begin with # symbol

```
#include <iostream.h>
```

→ PreProcess directive
#include → PreProcess
<iostream.h> → name of file

Date

• we can also define function Macros

ex:- #define circle(r) (3.14*(r)*(r))

• ## operator (Token - Pasting operators)

ex:-
```
#include <stdio.h>
#define concate(a,b) a##b
int main()
{ int xy=30;
  Printf("%d", concate(x,y)); // replace with Printf("%d",xy);
  return 0
}
```

⇒ dynamic memory allocation (DMA) :- • use to save memory as to utilised our Memory
• when we don't know the amount of memory we need then we use DMA

• static memory allocation done on stack
• dynamic memory allocation done on stack as well as heap

function used in DMA :-
i) malloc
ii) calloc
iii) realloc
iv) free

Date

i) Malloc function :- • used for Memory allocation
• reserve memory space of specific size & return pointer which pointing to memory location

Syntax ⇒ Ptr = (cast_type *) malloc (byte_size);
↓
datatype

ii) Calloc function :- • used for contiguous allocation
• used to allocate multiple block of memory
• syntax same as above only calloc in place of Malloc

iii) Realloc function :- • we can add more memory in size to already allocated memory.
• increase block size & leaving original content as it is
• some syntax replace Malloc with realloc

iv) free function :-
• in DMA you have to deallocate the memory explicitly if not, you may encounter out of memory error
Syntax free (Pointername);

⇒ type qualifiers in C :-
type qualifiers are the keyword used to modify the properties of variables
There are 2 type qualifiers :-
i) constant
ii) volitile

Date

• constant :- • when we define a variable constant then when ever we assign any value to it will not change
• use "const" keyword

• Volatile :- • volatile is used to create a variable whose values can't be changed in the program explicitly but can change by explicity by hardware or external devile
• for example varible used to store system clock

⇒ Abstract Data type :- it is a special kind of datatype whose behaviours is define by a set of values & set of operations

Some example of ADT are stack, queue, List (operation)

⇒ exception :- • exception are those which can be handeled at run time where as errors cannot.
• Example exception when we divise any thing by zero
• error when there is syntax error

⇒ file handling :- for store information & fetch information we use file handling

Date

- Mode of file handling :-
  - i) r - read
  - ii) w - write
  - iii) a - appent
  - iv) 'r+' - read & write
  - v) x - create new file, operation fail if file already exist
  - vi) t - open file in text mode
  - vii) b - open file in binary mode

⇒ threading :- when multiple tasks are performed parallel then it is called threading

⇒ Multithreading in Python :-
- multithread process share the same data space with the main thread, can share info. & communicate with each other easily.
- there is an instruction pointer which keep track of currently runing thread
- for already runing thread there are 2 state- interrupted or hold

- Process to start new thread :-
  - i) include thread module.
  - ii) call method by thread.start_new_thread (function_to_
  - iii) Create thread using below method
    thread.start_new_thread (function_name, (list of argument))

ex :-
```
import thread                          // old mode for thread
import time                            // new mode is threading
def Print_time (threadName, delay):
    count = 0
    while count < 5:
        time.sleep (delay)
        count += 1
        Print ("threadName" + str(time.ctime(time.time())))
try:
    thread.start_new_thread (Print_time, ("Thread-1", 2))
    thread.start_new_thread (Print_time, ("Thread-2", 4))
except:
    Print ("error")
while 1:
    pass
```

- Method under new thread module in Python :-
  - i) thread.activeCount() :- no. of active thread
  - ii) thread.currentthread() :- no. of thread object in caller's thread control
  - iii) thread.enumerate() :- return list of active thread.

- thread class implement following method :-
  - i) run() :- entry point of thread
  - ii) start() :- start a method by calling run()
  - iii) isAlive() :- whether a given thread is still execute or not
  - iv) getName() :- return name of thread
  - v) setName() :- set name of thread

Date

- Process to create a thread :-
  i) import threading Module
  ii) define a class inherit of threading. Thread class
  iii) overwite the __init__ function by initalising the varible
  iv) define Method like run(), start() etc
  v) define function to perform operation
  vi) declare object of thread using Proper argument
  vii) call start() Method of that class
  viii) thread is exit according to condition

⇒ Synchronizing thread :- • it is the concurrent execution of 2 or more thread that share critical resource

- to avoid the conflict of critical resource by 2 thread then it is necessary that thread must be synchronised
- In Python a thread class Lock is used for locking Mechanism

blocking
1 → acquire()
0 → release()

- the acquire (blocking) method of used for locking a resource by set value of blocking to 1 then it wait for lock to be release by Put it value to 0 by call release() Method

Operating System

Date

⇒ Process, thread, inter-Process communication, concurrency, Synchornization, Deadlock, CPU scheduling, Memory Managment, virtual Memory & file system.

⇒ CPU scheduling :- • it is a Process which allow one Process to use the CPU while the execution of another Process is on hold (in waiting state) due to unavailability of resource like I/O device

Term used in Scheduling :-

i) Arrival time :- time at which Process arrives in the ready queue

ii) Completion Time :- time at which Process complete it execution

iii) Burst Time :- Time required by Process for CPU execution

iv) Turn around Time :- Time Different bet. completion time & arrival time.

v) wait time :- Time Different bet Turn around Time & burst time

Type of CPU Scheduling :-

i) first come first serve :- • simplest scheduling algo in which when a Process request the CPU first is allocated the CPU first

Date

• implementation is Perform on queue

ii) Shortest job first (SJF) :- • Process which have shortest burst time are schedule first
• non-Premitive Process mean we can not interrupt before completion
• if 2 Job have same burst time then follow first come first serve

iii) Longest Job first (LJF) :- • same as short Job first
• non-premitive in nature

iv) Round Robin Scheduling :- • each Process is assigned a fixed time know as Quantum Time in cyclic way
• ready queue treated as circular queue
• if remain CPU burst time is longer than quantum the complete remain Process in next queue

v) Priority Scheduling :- • Process are scheduling according to their Priorities i.e. highest Priority
• if 2 Priorities is same then Schedule according to arrival time.

⇒ Deadlock :- • Deadlock is a situation where a set of Processes are blocked because each Process is holding a resource & waiting for another resource acquired by some other Process

Date

• method of handling deadlock :-

i) Deadlock Prevention or avoidance :-
• we can monitor each Process when is seen that a Deadlock will occur we do not complete that Process until the request resource is free

ii) Deadlock detection & recovery :- • Let deadlock occur
• after the soln make to remove deadlock

iii) ignore the Problem :- • if deadlock is very rear then let it happen & reboot the system
• this approach followed by window & Linux

Process :- Process is instance of a computer Program that is being executed by one or many thread

Gate 2020

65Q ⇒ 100 Marks (3 hours.

Aptitude 5Q of 1 Marks & 5Q of 2 Marks ⇒ 10Q ⇒ 15M

Enggin. 25Q of 1 Marks & 30Q of 2 Marks ⇒ 55Q ⇒ 85M

↓ ↓

engg. Maths ⇒ 75 Marks (10Q) Subject ⇒ 70 Marks (45Q)

5Q of 1 Mark & 5Q of 2 Mark 25Q of 2 Marks & 20 Q of 1 Marks

i) engg. Maths ⇒ Matrices, system of linear eqn, determinants, linear algebra ⇒ decomposition, eigenvalue & eigenvectors
relation, sets

ii) computer organization & architecture. :-
Machine instruction & addressing modes, ALU, data-path & control unit : Instruction pipelining, Memory hierarchy : Cache, Main Memory & secondary storage, I/O, interface (interrupt & DMA mode)

iii) Programming & data structure :-
Programming in C, recursion, array, stacks, queues, linked lists, trees, binary search tree, binary heap, graphs.

iv) Algorithms :-
searching, Sorting, hashing, ~~Asymptotic worst case time & space complexity~~ Asymptotic @ worst case time & space complexity.
Algorithm design technology : greedy, dynamic Programming & divide & conquer. Graph search. minimum spanning trees, shortest paths.

v) Theory of computation :- regular expression & finite automata, context free grammar & Pushdown automata. Regular & context free language, Pumping lemma, Turing Machines & undecidability

vi) compiler design :- Lexical analysis, Parsing, syntax-directed translation, Run time environment, intermediate code generation.

vii) Operation system :- Processes, threads, inter-Process communication, concurrency & synchronization, Deadlock, CPU scheduling, Memory Management & virtual Memory, file system

viii) DBMS- ER model, Relational model : relational algebra, tuple calculus, SQL. integrity constraints, normal forms, file organisation, indexing (e.g. B & B+ trees), Transaction & concurrency

ix) Digital logic X

x) computer Network X

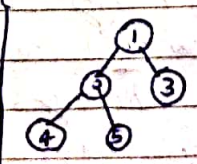
xi) Apptitude :- Numerical computation, numerical estimation, numerical reasoning & data interpretation

Remain

i) 8 - Previous year Paper

ii) 7 - full test

iii) 3 - advance test :-



inorder :- 42513
Postorder :- 45231
Preorder :- 12453

+ & -
/ & *
R & L

Programming & Data structure :-

Pointer :-

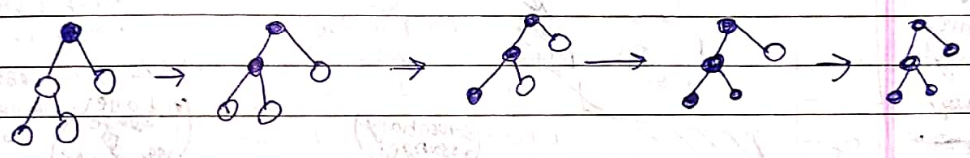
A = [1, 2, 3, 4, 5, 6]

*P = A (Point to 1)

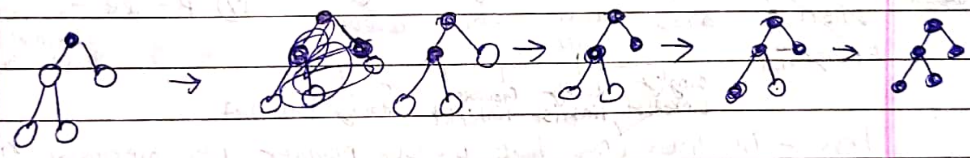
*P = P+4 (Point to 5)

*P[1] (Point to 6)

Depth first search is implement by stack



Breadth first search is implement by queue.



(normal or mix) Infix, Postfix (operator bad mai), Prefix (operator phele)

Infix ↓ a+B*C+D

(a+b) — stack insert operator only — (ab+)

no 2 operator of same Priority can not stay together in stack

or same Priority

or same Priority

~~Same Priority~~

Postfix → ABC*+D+

Prefix → ++A*BCD (+ab)

a+B+C+D → reverse → D+C*B+a ⇒ infix (reverse)

↓ convert

DCB*+a+ ⇒ Postfix

↓ reverse

+a+*BCD ⇒ Prefix

Use (BST)

in order :- left, root, right

Post —"— :- left, right, root

Pre —"— :- root, left, right

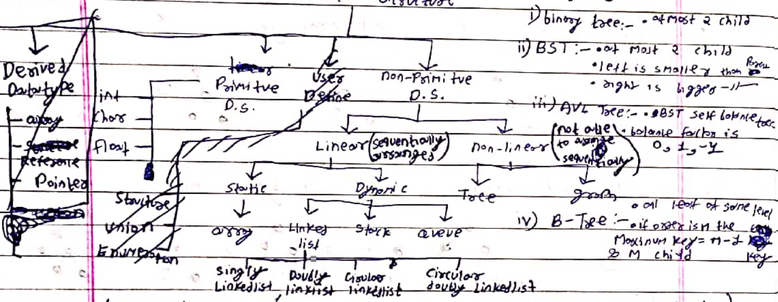
Page No. : Date : / /

→ linked list operation :-

for insert :- newnode.next → RightNode;
Leftnode.next → newnode;

for Delete RightNode.next → ~~T~~ RightNode.next;
~~T~~ RightNode.next → NULL;

→ tree :-



types
i) binary tree :- • atmost 2 child
ii) BST :- • at most 2 child
• left is smaller than parent
• right is bigger — II —
iii) AVL tree :- • BST self balance tree
• balance factor is 0, 1, -1
iv) B-Tree :- • all leaf at same level
• if order is m the Maximum key = m-1 key & M child

tree :- no-linear (not have a order manner i.e. hierarchical view)

→ binary search tree

- left part having lesser value than parent node & right part having higher value than parent node

insert & search ⇒ O(n) (time complexity)

Page No. : Date : / /

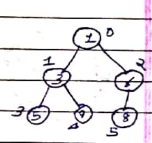
→ Binary Heap :-

- Binary heap is a complete tree (All level are complete upto last second). last level is fullied from left to right

- it is of a type :- min heap or max heap

Min Heap :- root value is minimum to all the element in heap

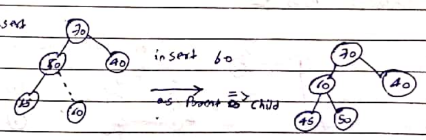
Max heap :- root value is maximum to all the element in heap.



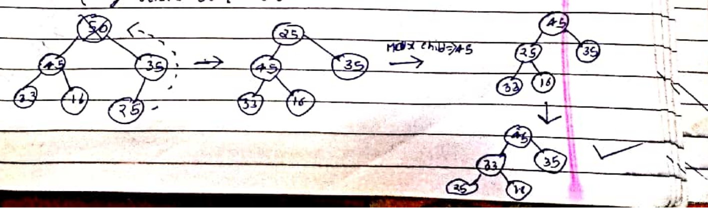
time complexity for sorting = O(nlogn)

Priority queue can be implement using Binary Heap with O(logn)

→ insert



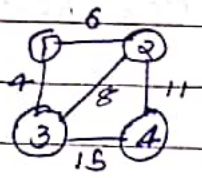
→ Delete (only delete root node)



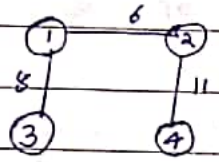
⇒ graphs → consist of (finite set of vertices & edges)

- Kruskal's minimum spanning Tree algo. :- (connected & undirected graphs)

minimum spanning Tree :- connected & undirected graphs with no cycle



weight	Branch
4	1-3
6	1-2
8	3-2
11	2-4
15	3-4

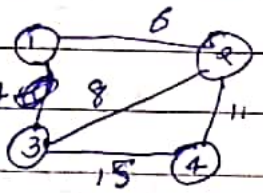


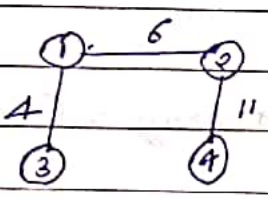
⇒ minimum spanning tree

(least weight is selected)
estop that vertex who create cycle

- Prims minimum spanning tree algo :-

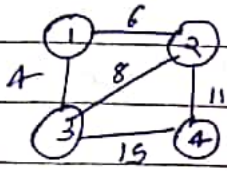
Start with inital node minimum connect node is selected

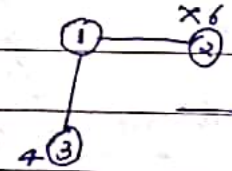




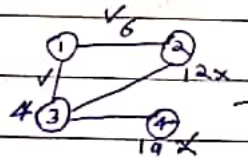
(seslop that vertex who create cycle)

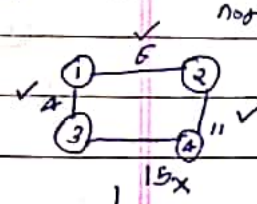
⇒ Dijikstra shorest path :-

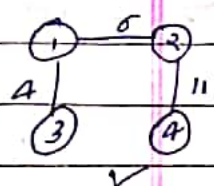




from start node select min. weighted node







⇒

⇒ algorithms :-

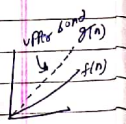
→ Search :- 2 type Linear search, binary search

- linear search :- • Search Scan one item at a time without jumping to any items.
  - Worst case is O(n)
  - time increase as element increase = O(n)
- binary search :- • Search middle element as forward reverse.
  - ~~worst case complexity is O(n)~~
  - check element is less or greater
  - check only half of list,
  - time complexity = O(log n)
- Sorting :-

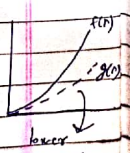
type — Bubble sort, selection sort, Merge sort, insertion sort, Quick sort, Heap sort.

→ Asymptotic Notations :- 3 type. — theta notation (Θ)
Big O notation (O)
omega notation (Ω)

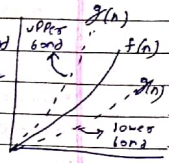
- ~~theta notation~~ Big O notation :- (O)
  - express upper bound of algo.
  - Measure worst case time complexity (Maximum time taken)



- Omega notation (Ω)
  - express lower bound of algo..
  - Measure best case time complexity (minimum time taken



- theta notation (Θ) :-
  - express lower as well as upper bond
  - Measure average case time complexity (average time taken



→ Algo. Design techniques :- types:
- Divide & conquer approach
- Greedy technique
- Dynamic Programming

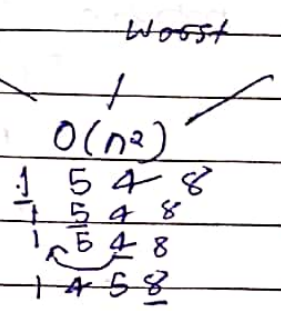
- Divide & conquer approach :-
  - Divide origin Problem into a set of subproblems
  - Solve each subproblem individually
  - combine the soln of subproblems recursively into a soln of whole original problem

- Greedy Technique :-
  - Used to solve optimization Problem
  - optimization Problem is one in which we are given a set of maximized or minimized. i.e. some condition
  - Greedy does not guarantees that soln is optimal but value find is very close to Optimal.
  - Make a choice that best of that moment.

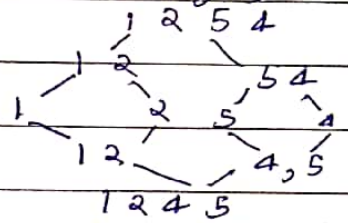
- Dynamic Programming :-

⇒ sorting :-

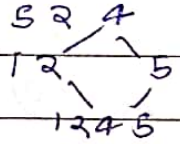
i) insertion sort :- • time complexity is $O(n^2)$ — best, worst, average
- select one element
- if next element is less then select then arrange it
- other wise go to next element



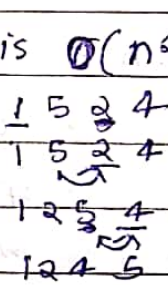
ii) Merge sort :- • time complexity is $O(n \log n)$ — best, worst, average
- divide in 2 part by $\left(\frac{\text{lower} + \text{upper}}{2}\right)$
- Call Merge sort for first half then second
- Merege result of both half



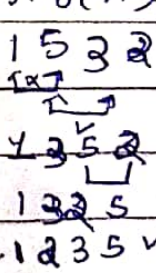
iii) Quick sort :- average case is $O(n \log n)$ — Best
- Pick Pivot Point — worst case is $O(n^2)$ → when Pivat is smallest or greatest
- divide in 2 part greater & less than Pivat
- repeoge it. then sort & Merge. it



iv) selection sort :- time complexity is $O(n^2)$ — best, worst, average
- select first element
- find minimum value equal or less then selected
- swap it with selected
- repeatly it. (2 loop is use)



v) bubble sort :- time complexity of worst case $O(n^2)$ — average, best
- select first element
- if select element is greater than next element swap
- other wise nothing
- compose next 2 element until last
- start from start 2 element repeate until list is sorted.



vi) Heap sort :- time complexity is $O(n \log n)$
- using Heap rule sort the list.

Data Structure

⇒ binary tree :- • binary tree is part of non-linear datatype, i.e. tree
• in which every node or vertex has either 0 or 1 child or 2 child
• two ways to represent it
i) ussing array
ii) using Linked lists
• use to represent hierarchical relationship between element

type of binary tree :-

i) full binary tree (Proper binary tree) :- tree in which except root node has exactly 2 child

ii) complete binary tree :- • tree in which all level is filled & in last level filled left as possible (like heap)

⇒ BST :- Binary search tree

Syllabus e-litmus

3 section are these :- (20 each)

i) Quantitative Ability (8-10) • (3-6) { (6-8)
ii) Logical Reasoning (10-12) (3-6) (6-10)
iii) Verbal Ability (12-15) (6-10) (10-12)

i) Quantitative Ability :-

i) Geometry - (H)
ii) Speed, Time and Distance - (H).
iii) Mensuration (L)
iv) Time And Work - (M)
v) Number System - (H)
vi) Probability - (M)
vii) Permutation and Combinations - (M)
viii) Few Miscellaneous Question on topics like Progress, Algebra
ix) Averages →
x) Co-ordinate Geometry → (Part of Geometry)
xi) Logarithms → $\log_a a = c$ $a = b^c$
xii) Quadratic eqn → Geo. $x^2 - (\text{sum of root})x + (\text{Product of roots})$
xiii) AP, GP, HP (M) →

ii) Logical Reasoning :-

i) Cryptarithemetic - (H)
ii) Logical Reasoning (Mo)
iii) Data interpretation (Bar & Pie charts) (H)
iv) Data sufficient (Algebra & arithmetic, Geometry, Logic & Misc) (H)
v) Data Tabulation based Questions
vi) Arrangement Based Problems.

iii) Verbal Ability

✓ i) Reading comprehension (H)

✓ ii) Para jumbles (M)

- iii) Sentence completion based on vocabular (M)

- iv) Sentence completion based on subject - verb agreement (M)

v) Miscellaneous question related grammer.

vi) Grammatical Errors based Questions

vii) Paragraph Based Questions.

viii) Fill in the blanks

⇒ Cryptoarithmatic conversion :-

i) Search for 0 or 9

$A * B \neq A$ or $A + A = A$ here $A = 0, 9$ (carry 1)

ii) additional 1

Send + More = MONEY here M=1 as answer is one more digit

iii) Search for 1 in multiplication or division

MAD × B = MAD here B must be 1

iv) Search for 6 & 1 in Multiplication

$A \times B = A$ here B=1 or B=6 (when A is even)

v) search 0 & 5 in multiplication

$A \times B = BA$ here A = 0 or 5 (7×5 = 5 (+30 carry))

⇒ i) Previous year Paper :-

- last 2 digit of 7^{45}

Sol) last 2 digit of $7^1 = 07$

$7^2 = 49$

$7^3 = 43$

$7^4 = 01$

$7^5 = 07$

So, $7^{45} \Rightarrow 07$ as last digit

- factor common to 30^{11} & 20^{13}

Sol) HCF of both is 10^{11}

all the factor common to both = $10^{11} = 2^{11} * 5^{11}$

no. of common factor = (11+1)(11+1) = 144

- a work done in 24 day with certain worker, if 7 of them are absent & remain is done work in 30 day the no of employ work on 2nd day is

Sol) total work done if no of employ is x = 24x

$(x-7)30 = 24x$

$x = 35$

2^{nd} day = $x - 7 = 28$

- when integer n is divisible by 8, remainder is 3 then what is remainder of 6n is divisible by 8

Sol) $n = 8k + 3$

$6n = 6(8k + 3)$

$= 8(6k) + 8(3) + 2$

6n is divisible by 8 & remainder is 2

- no of common term in 2 series 17, 21, 25 ... 417 & 16, 21, 26 ... 461 is

Sol) n^{th} term of 1st series = $17+(n-1)4 = 4n+13$

m^{th} term of 2nd series = $16+(n-1)5 = 5m+11$

$4n+13 = 5n+11$

$n = \frac{4n+2}{2}$ $[m = 2, 7, 12 \ldots]$

- How many integers are formed using digit 0, 1, 2, 3, 4 but no. is greater than 999 & not greater the 4000 ?

Sol) $\square\ \square\ \square\ \square$ 4000

$3 \times 5 \times 5 \times 5 = 375 + 1 = 376$

- What are the distinct term in Expansion of $(a+b+c)^{20}$?

Sol) $(20+3-1)C_{(3-1)} = {}^{22}C_2 = \frac{22!}{2! \times 20!} = 11 \times 21 = 231$

- integers 1, 2, 3 ... 40 written on board & 39 operation is done. In each operation one of no. is erased & new no $a+b-1$ is written.

Sol) sum of no $1+2+3+4 \ldots 40 = 820$

no of operation $\Rightarrow$ 39

last no. on board is $= 820 - 39 = 781$

- code generated of 2 digit form 0 to 9 but ~~some digit code~~

- Let $U[n+1] = 2U[n]+1$ $(n = 0, 1, 2, \ldots)$, $U(0) = 0$ then $U[10]$

Sol) $U[0] = 0$, $U[1] = 2\times0+1 = 1$, $U[2] = 2*1+1 = 3$

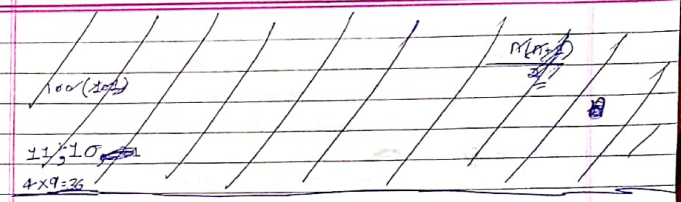
$U[3] = 2*3+1 = 7$ $U[4] = 2*7+1 = 15$ $U[5] = 2*15+1 = 31$

$U[6] = 2\times31+1 = 63$ $U[7] = 2\times63+1 = 127$ $U[8] = 2*127+1 = 255$

$U[9] = 2*255+1 = 511$ $U[10] = 2*511+1 = 1023$

in $U[0], U[1] \ldots = 2^n - 1$

$U[10] = 2^{10} - 1 = 1024 - 1 = 1023$



- Product of 1 to 100 has no of zeroes at end

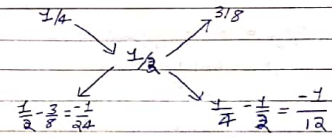
Sol) $\left(\frac{100}{5}\right) + \left(\frac{100}{25}\right) = 24$ as count no of 5! in 100!

- 2 vessel contain Mixture A & B in ratio 1:3 & 3:5 find the ratio in which 2 Mixture mix to get a Mixture of 1:2 from

Sol) ~~Vessel~~ A contain in vessel 1 $\Rightarrow$ 1/4

A contain in vessel 2 $\Rightarrow$ 3/8

rule of allligation



required ratio $= -1/24 : -1/12 = 1/2$

- Vessel full of water & kerosene in which 18% is kerosene. Now 8 L of Mixture is drawn off & again vessel full with Petrol now 15% of Mixture is kerosene so Volume of vessel is :-

Sol) ratio = 18:82 $\Rightarrow$ 9:41

mixture of kerosene = $9x/50$ (before drawn off)

final kerosene after drawn off $\Rightarrow$ $9x/50 - 8*9/50$

$= \left(9x - 72/50\right)$

$\frac{15x}{100} = 9x - 72$ $\quad x = 48$

- a Mixture of Water & Sugar in which Sugar cost 12/L is sell of a Profit of 25% with 13.75/L of mixture, then What is ratio of Water & Sugar

Sol) Let 1 L is Sugar cost = 12
x L — " — = 12x

Profit = 25% of 12x = 3x
sell Price = 12x + 3x = 15x
Let y Le of water in Mixture
Selling Price ⇒ 13.75 (x+y) = 15x
x/y = 11/1

- The value of y in $\log_y 1369\, y = 3$

$\log_y 1369 y = 3$

$\log_y 1369 + \log_y y = 3$
$\log_y 1369 + 1 = 3$
$\log_y 37^2 = 2$
$2 \log_y 37 = 2$

$\frac{\log 37}{\log y} = 1$

$\log 37 = \log y$
$y = 37$

8) In a race of 200 m A beat B by 15m & C by 21m. By how much distance will B beat C in a race of 3700m

Sol) When B at 185m then C at 179m = 6m gap

B x 20 = 3700m C x 20 = 3580m When race is of 3700m

then gap is 3700 − 3580 = 120 m

9) two train having lengths L_1 & L_2 having velocities V_1 & V_2 cross each other in t unit of time when Moving in opposite direction & take T unit time when Moving in same direction if ratio of T to t is 11 then value of V_1 & V_2 is

Sol) $T = L_1 + L_2 / V_1 - V_2$
$t = L_1 + L_2 / V_1 + V_2$

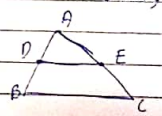
$\frac{T}{t} = \frac{11}{1}, \quad \frac{V_1 + V_2}{V_1 - V_2} = \frac{11}{1} \quad \frac{V_1}{V_2} = 1.2$

10) Compound interest $= P\left(1 + \frac{r}{100}\right)^t$

simple interest $= \frac{P \times r \times t}{100}$

Cubical eqn ⇒ $ax^3 + bx^2 + cx + d$ then, $\alpha+\beta+\gamma = -b/a$, $\alpha\beta + \beta\gamma + \gamma\alpha = c/a$, $\alpha\beta\gamma = -d/a$

triangle Property



$\frac{AD}{AB} = \frac{AE}{AC} = \frac{DE}{BC}$

Q) Stream questi on river:-

boat speed in still water = U km/hr   upstream = U - V
speed of stream = V km/h   downstream = U + V

$(U+V) \div H$

Volume :-
cube = a^3
cubiod = $a \times b \times c$
cone = $\frac{1}{3}\pi r^2 h$
cylinder = $\pi r^2 h$
sphere = $\frac{4}{3}\pi r^3$
Pyramid = $\frac{1}{3} \times$ l $\times$ b $\times$ h {base length $\times$ base width}

Surface area :-
cube = $6s^2$
Cubiod = $2(lw + lh + hw)$
cone = $\pi r^2 + \pi \times r \times$ slant height
cylinder = $2\pi r^2 + 2\pi r h$
sphere = $4\pi r^2$

→ Present, Past, future

had ⇒ Past
have ⇒ Present (I, you, we, they), has ⇒ he, she, it & singular noun (or Plural noun)
, will ⇒ future, mainly uses will, shall = for unsure
going to ⇒ future

mean = average
median = Middle of sorted list (if 2 no. in middle take average)
mode = no. that occur max. no. of times

Q) Milkman can allow water in milk in the ratio 1:4 to ensure the quality. He has 2 samples of milk & water mixture. one is in ratio 1:5 2 other is 2:3 In what ratio both sample to mix to ensure quality of milk

Sol) $\frac{x + 2y}{5x + 3y} = \frac{1}{4}$

$\frac{x}{y} = \frac{1}{6}$

Q) A, B, C start running at the same time & at the same point in the same direction in a circular stadium. ~~~~ they complete round in 252 s, 300 s & 198 s. after what time they
LCM of 252, 300, 198 = 69300/60 = 1155 min

⇒ for equating root of a quadratic eqn find root ~~~~ then compare.

Q) In how many different way can 5 boys & 2 girls be arranged in a row such that all boys stand together & all girls stay together.

let 5 boys be A & 2 girls be B
then arrange A & B = 2!
arrang 5 boys together = 5!
arrang 2 girl together = 2!

no. of way possible = 2! × 5! × 2!
= 120 × 4
= ~~24~~ 480

Q) $|x-y| = 1.2$ & $x^2 - y^2 = 1.6$ Find $|x|+|y|$

Sol) $|x-y| = 1.2$

$(x-y)^2 = (1.2)^2$

$1.44 = x^2 + y^2 - 2xy$

$1.44 = 1.6 - 2xy$

$2xy = 1.6 - 1.44$

$= 0.16$

$(|x|+|y|)^2 = x^2 + y^2 + 2xy$

$= 1.6 + 0.16$

$= 1.76$

$|x|+|y| = \sqrt{1.76}$

⇒ AP, GP, HP

$1+2+3\ldots n = \frac{n(n+1)}{2}$

$1^2+2^2+3^2+\ldots n^2 = \frac{n(n+1)(2n+1)}{6}$

$1^3+2^3+3^3+\ldots = \left(\frac{n^2(n+1)}{2}\right)^2$

Q) Product of 2 Co-Prime no. is 153 then LCM is

Ans) HCF of 2 Prime no. is 1

LCM × HCF = 153

LCM × 1 = 153

LCM = 153

Q) Mehdi can complete a work in 25 hr. If he joined by Jaahavi who is 50% more efficient, in who many days ~~both can~~ work can finish by work together.

Mehdi in 1 hr. = $1/25$

Jaahavi in 1 hr = $1/25 + \frac{50}{100} \times \frac{1}{25} = 3/50$

combine = $1/25 + 3/50 = 1/10$ so to day need to

⇒ Trignometry :-

P	B	P	H	H	B
H	H	B	P	B	P
Sinθ	Cosθ	Tanθ	Cosecθ	Secθ	Cotθ

Angle	0	30°	45°	60°	90°
sinθ	0	$1/2$	$1/\sqrt{2}$	$\sqrt{3}/2$	1
cosθ	1	$\sqrt{3}/2$	$1/\sqrt{2}$	$1/2$	0
Tanθ	0	$1/\sqrt{3}$	1	$\sqrt{3}$	∞

$\sin^2\theta + \cos^2\theta = 1$

$\tan^2\theta - \sec^2\theta = 1$

$\cot^2\theta - \operatorname{cosec}^2\theta = 1$

$\sin(a+b) = \sin(a)\cos(b) + \cos(a)\sin(b)$

$\sin(a-b) =$ ——"—— $-$ ——"——

$\cos(a+b) = \cos(a)\cos(b) - \sin(a)\sin(b)$

$\cos(a-b) =$ ——"—— $+$ ——"——

$\tan(a+b) = \frac{\tan(a)+\tan(b)}{1-\tan(a)\tan(b)}$

$\tan(a-b) = \frac{\tan(a)-\tan(b)}{1+\tan(a)\tan(b)}$

- if A, B, C are angle & a, b, c are side of triangle then :-

$\frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C}$;

$c^2 = a^2 + b^2 - 2ab\cos C$

$a^2 = b^2 + c^2 - 2bc\cos A$

$b^2 = a^2 + c^2 - 2ac\cos B$

Q) 2 men & 3 women can do a work in 10 day while 3 Men & 2 women can do it in 8 day. In how many day can 2 men & 7 women complete the work

Sol) $x \Rightarrow$ efficient of men $\quad y =$ efficiente of women

$$(2x + 3y)10 = (3x + 2y)8$$
$$2x = 7y$$

$$(2x+y)d = (2x+3y)10$$
$$(7y+y)d = (7y+3y)10$$
$$7yd = 100y$$
$$d = 100/7$$

Q) $\Rightarrow$ logrithums :-

$$\log_b a = y \Rightarrow a = b^y$$

- $\log_b(x \cdot y) = \log_b x + \log_b y$
- $\log_b(x/y) = \log_b x - \log_b y$
- $\log_b(x^y) = y \log_b(x)$
- $\frac{\log_c(x)}{\log_c(b)} = \log_b(x)$

- $\log_b(0) = \infty$
- $\log_b(1) = 0$
- $\log_b(b) = 1$
- $\log_e(2) = 0.69$
- $\log_e(3) = 1.09$
- $\log_e(4) = 1.38$
- $\log_e(5) = 1.6$
- $\log_e(7) = 1.94$