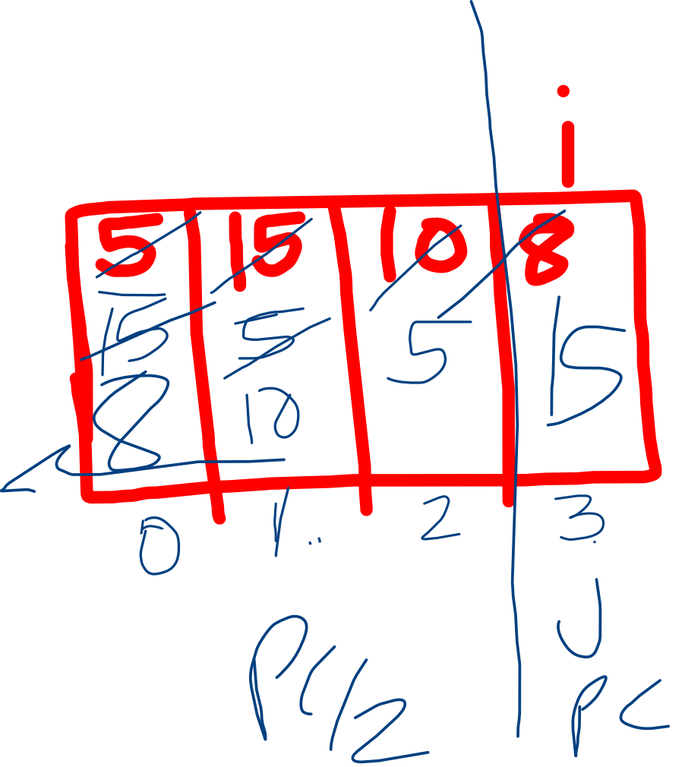
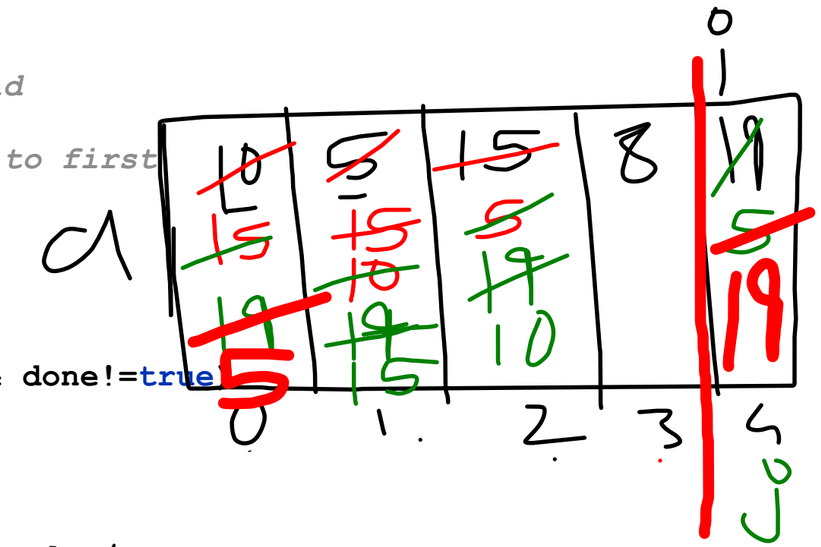
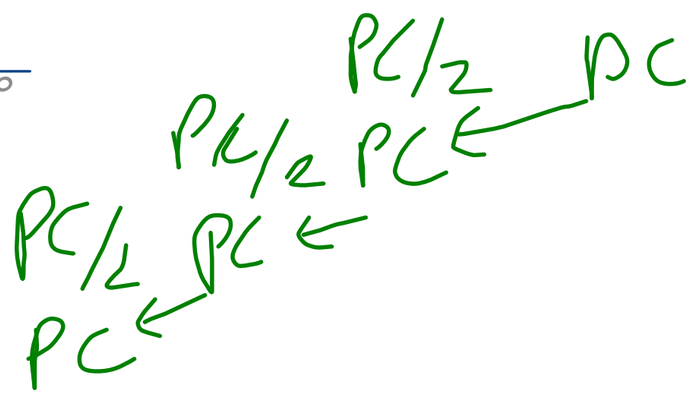
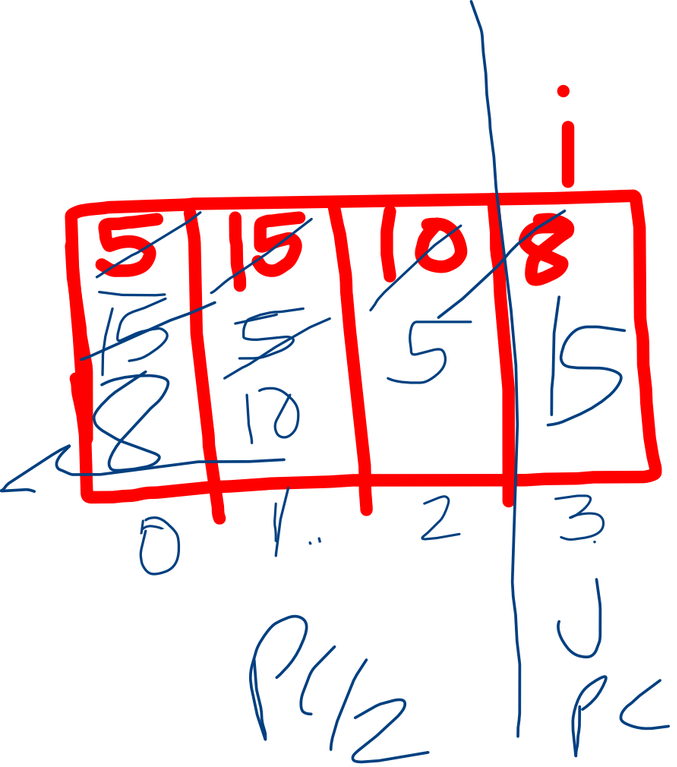


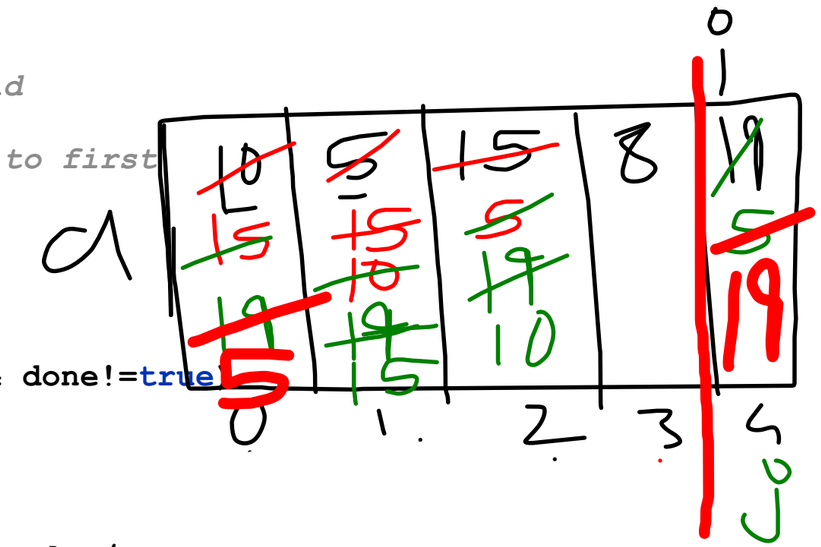

```
static void heap_sort(int a[])
{
    int temp,i,j,pc;//pc:parent-child
    boolean done;
    for(i=a.length-1;i>0;i--)//last to first
    {
        for(j=0;j<=i;j++)
        {
            done=false;
            pc=j;
            while(pc>0 && pc/2>=0 && done!=true)
            {
                if(a[pc]>a[pc/2])
                //child   parent
                {
                    temp=a[pc];a[pc]=a[pc/
2];a[pc/2]=temp;
                    pc=pc/2;//go to parent to
check
                    done=false;
                }
                else
                {
                    done=true;
                }
            }
        }//j closed
        temp=a[0];a[0]=a[i];a[i]=temp;//swap
largest to last
    }
}
```

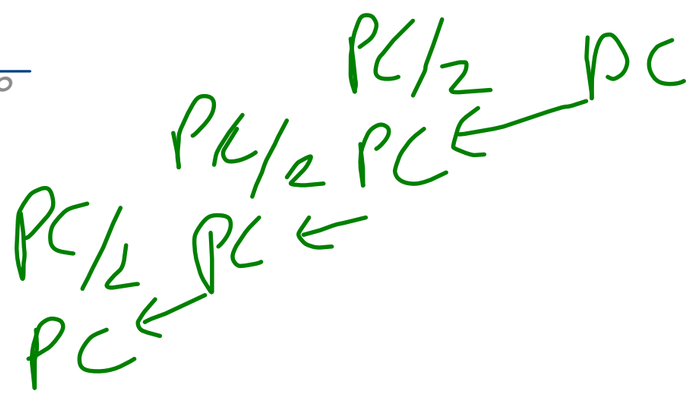







```
static void heap_sort(int a[])
{
    int temp,i,j,pc;//pc:parent-child
    boolean done;
    for(i=a.length-1;i>0;i--)//last to first
    {
        for(j=0;j<=i;j++)
        {
            done=false;
            pc=j;
            while(pc>0 && pc/2>=0 && done!=true)
            {
                if(a[pc]>a[pc/2])
                //child   parent
                {
                    temp=a[pc];a[pc]=a[pc/
2];a[pc/2]=temp;
                    pc=pc/2;//go to parent to
check
                    done=false;
                }
                else
                {
                    done=true;
                }
            }
        }//j closed
        temp=a[0];a[0]=a[i];a[i]=temp;//swap
largest to last
    }
}
```

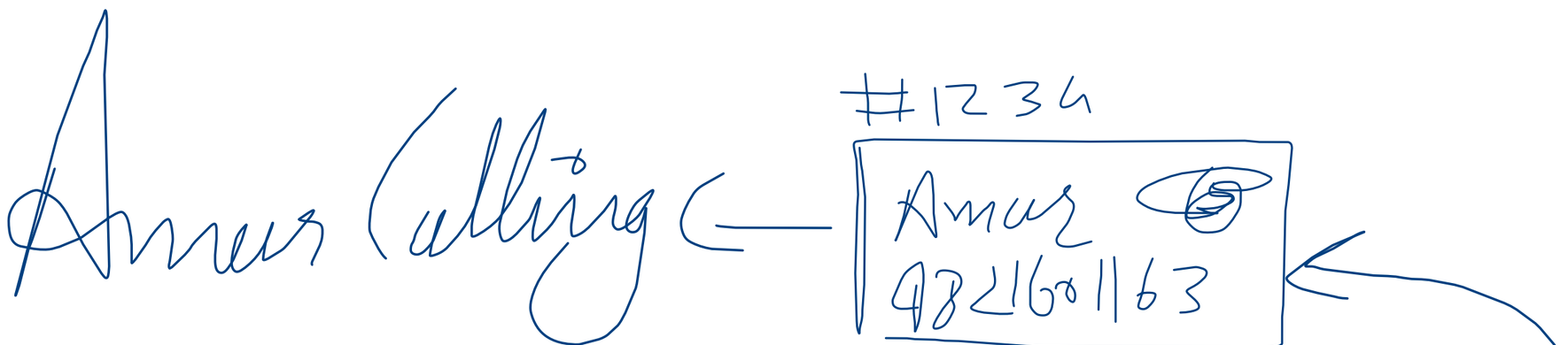




Hashing:
It's a process of finding anything needed in a single pass, theoretically.
In this, a process of hashing is followed while saving the data later on for searching it back in a single pass.

most common example:Mobile:

phase 1: 9821601163:-->save amar --->Hash_Function(9821601163)
|
V
#1234



Amar calls---->CLIP:9821601163--->Hash_Function(9821601163)-#1234

Parts:

1 Hash function

CONVERTS DATA TO INDEX FOR STORAGE

2 Hash Table

SAVES DATA AT INDEX PROVIDED BY HASH FUNCTION

IF ANY COLLISION HANDLE IT

3 Data

INFO TO STORE FOR LATER TO ACCESS.

Hash function:

Hash function should be fast, easy to code, and should spread data evenly(Try to avoid collisions).

Collision is when two or more data units are given the exact index.

Parts:

1 Hash function

CONVERTS DATA TO INDEX FOR STORAGE

2 Hash Table

SAVES DATA AT INDEX PROVIDED BY HASH FUNCTION

IF ANY COLLISION HANDLE IT

3 Data

INFO TO STORE FOR LATER TO ACCESS.

Hash function:

Hash function should be fast, easy to code, and should spread data evenly(Try to avoid collisions).

Collision is when two or more data units are given the exact index.

Hash Functions:

1.direct hash:

Data is treated as index.
Use only when one-on-one mapping of data and index is possible.
STUDENT 1 has Rollno 1

2.Subtraction hash.

A constant is subtracted from data to generate index.
202501110
yyyybbrrr
HF(ROLL-202501000)->index
110

3.digit extraction:

use part of data as index
HF(202501110)--->last 3 digits as index --->110
ex:bank--->account no.

4.mod div

data%n--->index
n:prime no or size of HT
-ve:one of the most collision producing tech
110%10:0 10%10:0 300%10:0

5.random number+ mod div

(a.(data)+b)%n
a,b:random numbers
n:prime no or size of HT

6.mid square method:

data--->square it --> use mid digit/digits as index
11------>121---------->2 index
7-------->049---------->4
3-------->009---------->0

7.folding tech:

data is broken in parts all parts added to make index
9821601163--->98+21+60+11+63----> index

collision

Two or more data given exact index which is already used is called collision.

To simplest, it will take longer time to search something, and to highest, it will induce delay in every operation.

handling:

-probing :linear v/s quadratic

if ith is taken try i+1,i+2,i+3,i+4,.....i+n first free found

quadratic:we jump i,i+2^n:i,i+1,i+2,i+4,i+8,i+16

the idea is jump can get us away from crowded place

longer sequential search called clustering before saying yes/no

-bucketing:2d HT if ith row is filled take ith row jth column

large memory space used 10X100--->1000

-chaining:linked list,in collision create new node and add to list

dynamic but too complex to use

-double/tripple hashing:

Hash function fails and you get a collision. Try secondary, and if it fails, try 3rd one.

kt exams