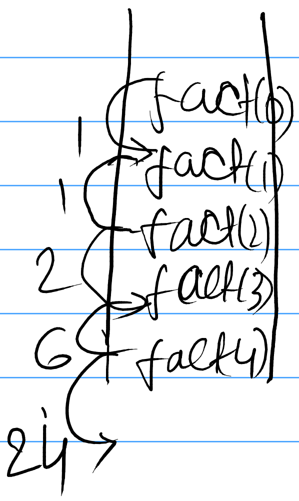
Recursion

We assume that we have solution of smaller problems. We solve big problems by breaking it into smaller problems by breaking it into smaller problem we keep breaking until we reach a case where we cant break it.

I/p 4
O/p 24
$4 \times 3 \times 2 \times 1 = 24$

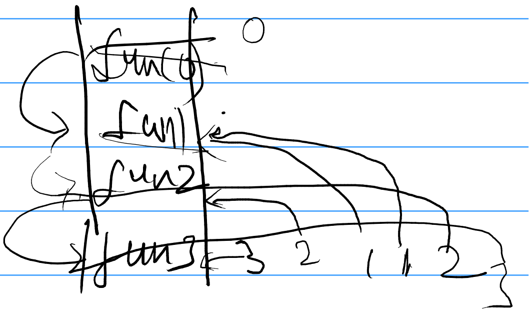
```
int fact(int n)
{   if (n==0)
      return 1;
    return n*fact(n-1);
}
```

for 4



Q)
```
void fun(int n)
{  if (n<1)
       return;
   else
   {  cout << n << " ";
      fun (n-1)
      cout << n << " ";
   }
}
```

```
main( )
{ fun(3) };
```



output : 3 2 1 1 2 3

Writing base case in recursion:

$\boxed{n=0}$

factorial n when $\boxed{n \geq 0}$

```
int factorial (int n)
{
    if (n==0)
      return 1;
    return n* factorial(n-1);
}
```

fabbinocci Number

2

```
int fabbi(int n)
{
    if.( n<= 1 )
        return n;
    return fabbi(n-1)+ fabbi(n-2);
}
```

5

1 1 2 3 5

① Write a recursive funtion to print numbers from n to 1 for a given n
I/P n=5
O/P 5 4 3 2 1

code

```
void printn21(int n)
{
    cout<<n<<" ";
    printn21(n-1);
}
```

② write a recursive function to print numbers from 1 to n for given n
I/p. n=5
O/P 1 2 3 4 5

Code
```
void print12n(int n)
{
        if(n==0)return;
        print12n(n-1);
        cout<<n<<" ";
}
```

Tail Recursion

①
```
Void fun(int n)
{
    if(n<1)
        return;
    cout<<n<<" ";
    fun(n-1);
}
```
recursion at tail

②
```
Void fun(int n)
{
    if (n<1)
        return
    → fun(n-1);
    cout<<n<<" ";
}
```

Tail recursive function run faster due to compiler optimisation. That is called Tail call back elimination.

② can be optimised by adding extra parameter

```
Void print1ton(int n, int k=1)
{
    if(n==0) return;
    cout<<k<<" ";
    print1ton(n-1,k+1);
}
```

even fabbinocci series can be optimesed using
trail reeusinon;

```
int    fact(int N, int val = 1)
{
    if (n == 0) return val;
    return fact(n-1, n*val);
}
```

Q) Write a recursive function to check if a string pallindrome.

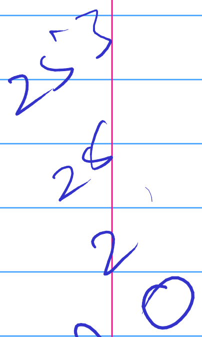
I/p : str = "aabaa"
O/p : yes
cod :

```
    bool    ispal (string s, int s, int e
    {
        if (s > e) return true;
        if (str[s] != str[e])
                return false;
        return ispal (str, s+1, e-1);
    }
```

Q) Write a recursive function to find sum of digits in a number
I/p n = 253
O/p: 10

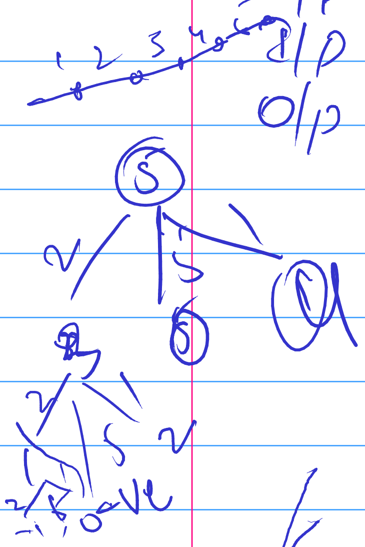
Code:

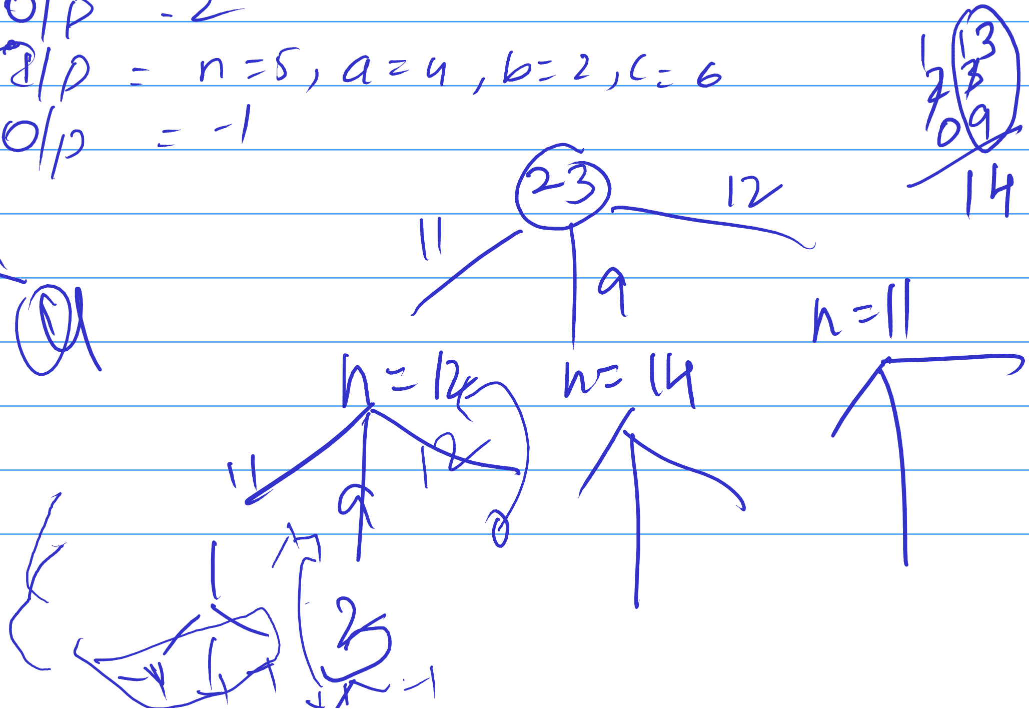
```
int sumofdigits(int n, int val=0)
{
        if( n==0) return val;
        return (n/10 , val+n%10);
}
```



Q) Given a rope length n, you need to find maximum number of pieces you can make such that length of every piece is in set {a,b,c} for given three values a, b and c

I/p n=5, a=2, b=5, c=1
O/p 5
I/p n=23, a=11, b=9, c=12
O/p : 2
I/p = n=5, a=4, b=2, c=6
O/p = -1





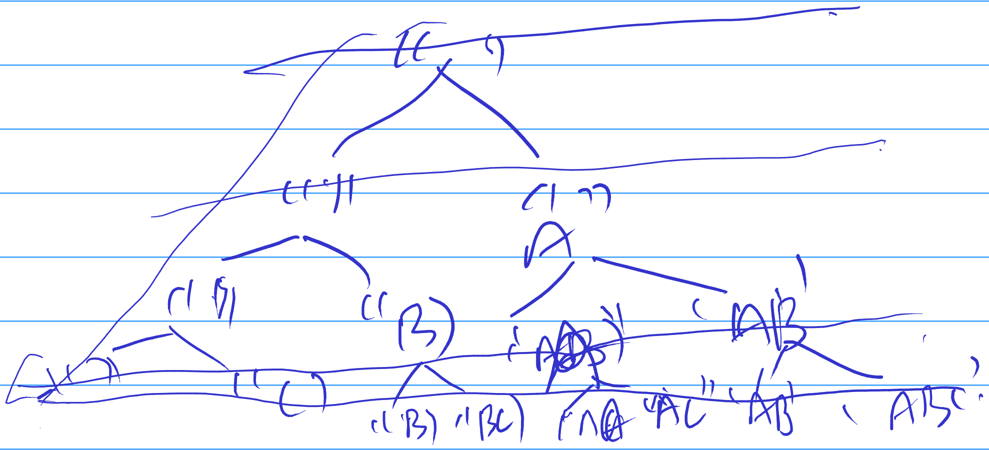
```
int maxcuts(int n, int a, int b, int c)
{
        if (n==0) return 0;
        if (n<0)   return -1;
        int res = max(maxCut(n-a,a,b,c)
                                  , maxcut(n-b,a,b,c),
                                        maxcut(n-c,a,b,c));

        if(res == -1) return -1;
        return   res+1;
```

Q) Given a string, print all subsets of it (in any order)

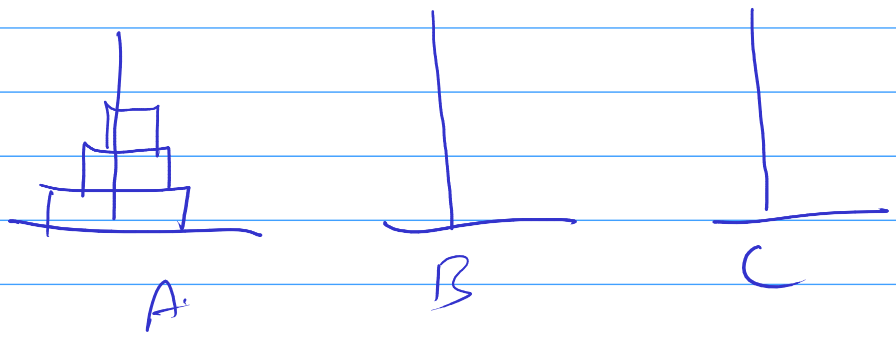
I/p   Str = "ABC"

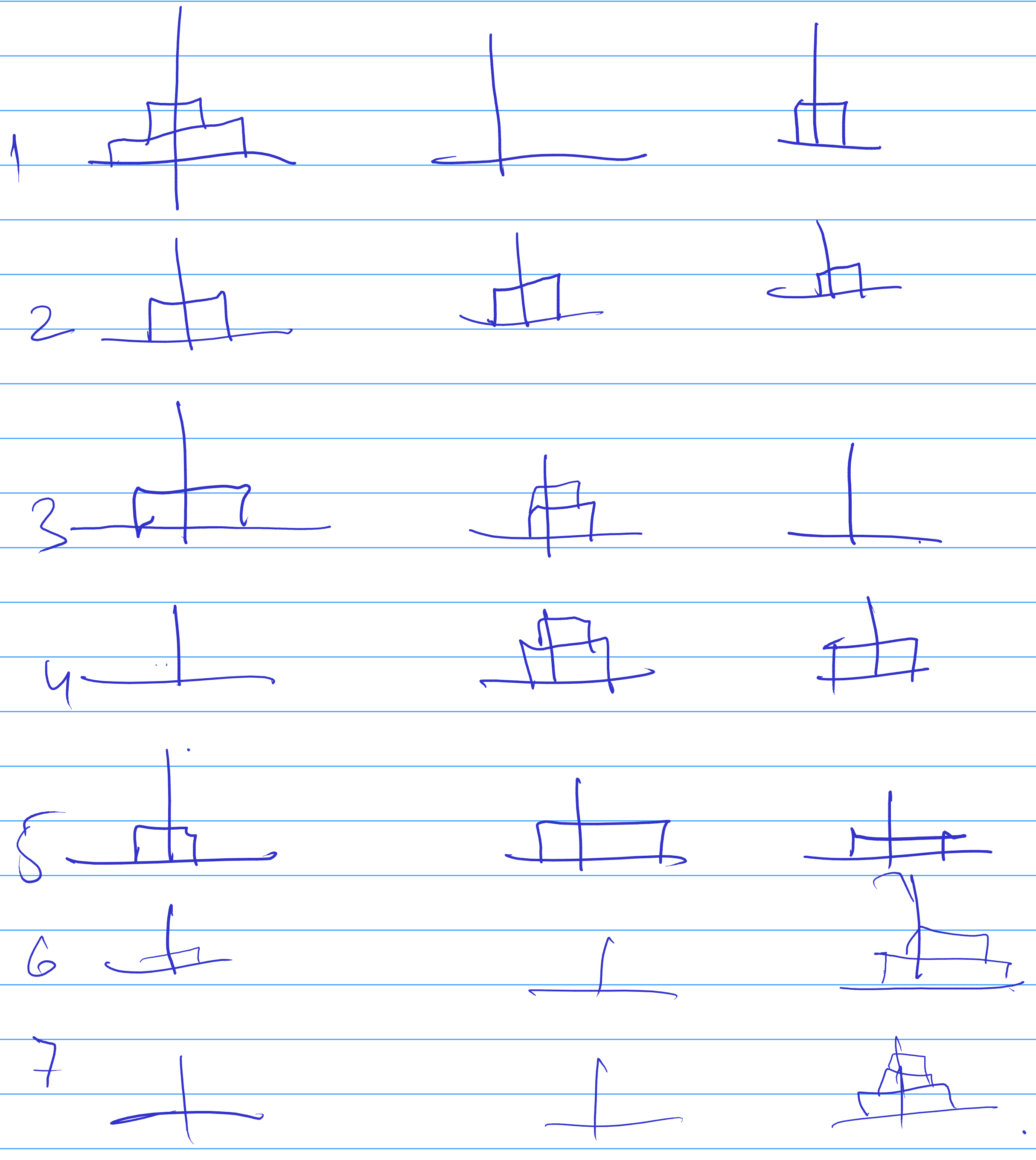
O/p   " " "A" "B" "C" "AB" "BC" "CA" "ABC"



Tower of hanoi

A to C





code:

```
Void TOH(int n, char A, char B, Char C)
{
    if (n = =1)
    { cout<< "move 1 from" << A << "to" << C << endl;
      return;
    }
    TOH(n-1, A, C, B)
    cout <<
```