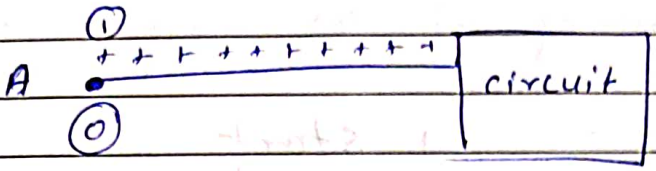
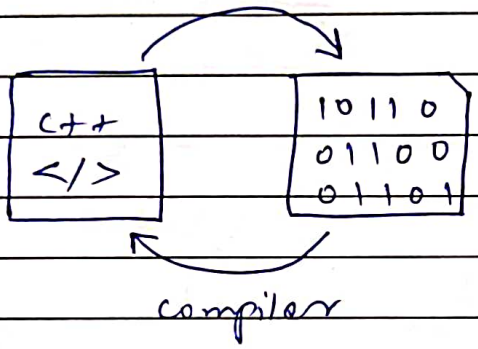
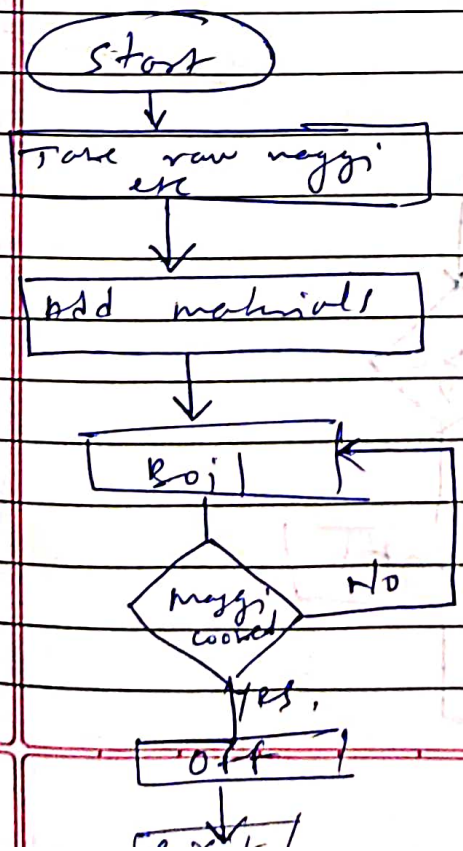
C++ DS + Algo



If current is going in circuit then it is 1 and when the current is not going then it is 0

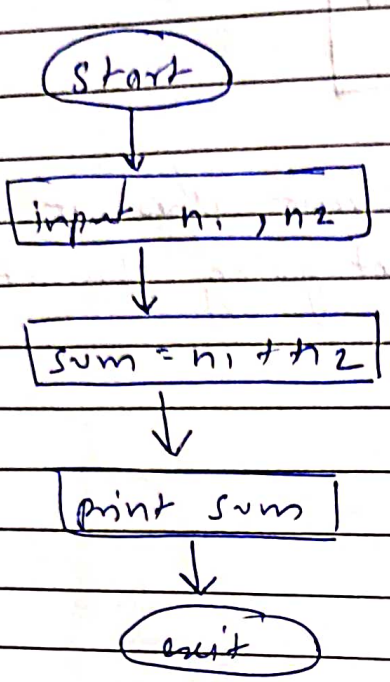


* Example



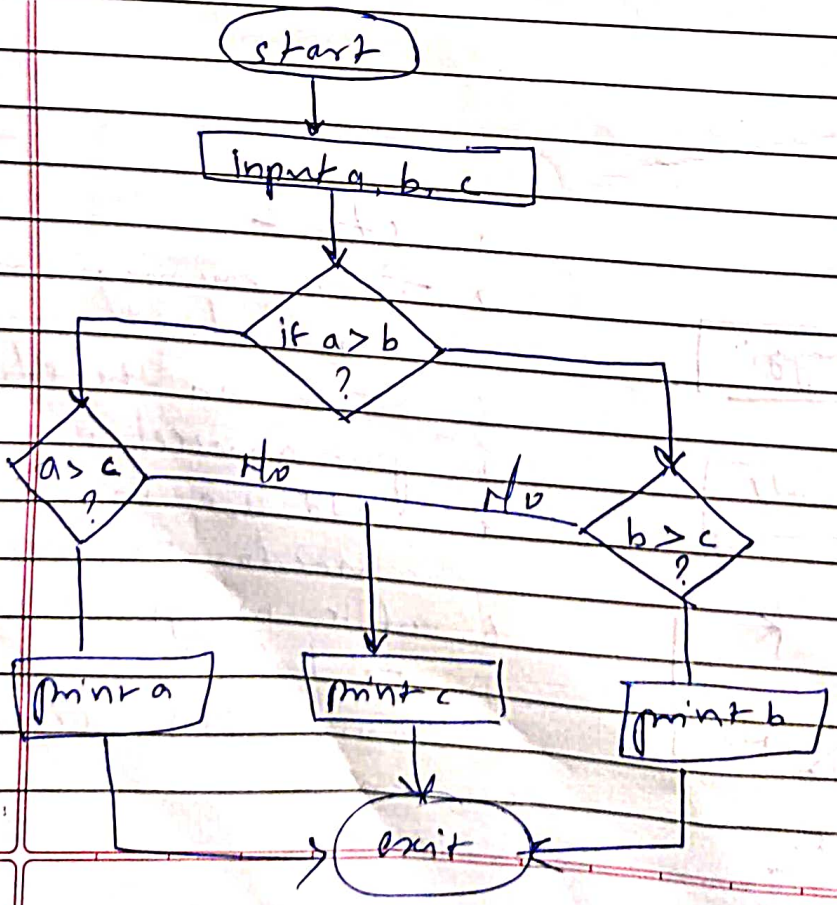
1. start
2. Take raw maggi, masala and water
3. Turn on the stove
4. put a vessel on stove
5. Add H2O, maggi and masala to vessel
6. while maggi is uncooked to boil
5. Turn off stove
6. serve
7. exit.

Q.2. add two no.

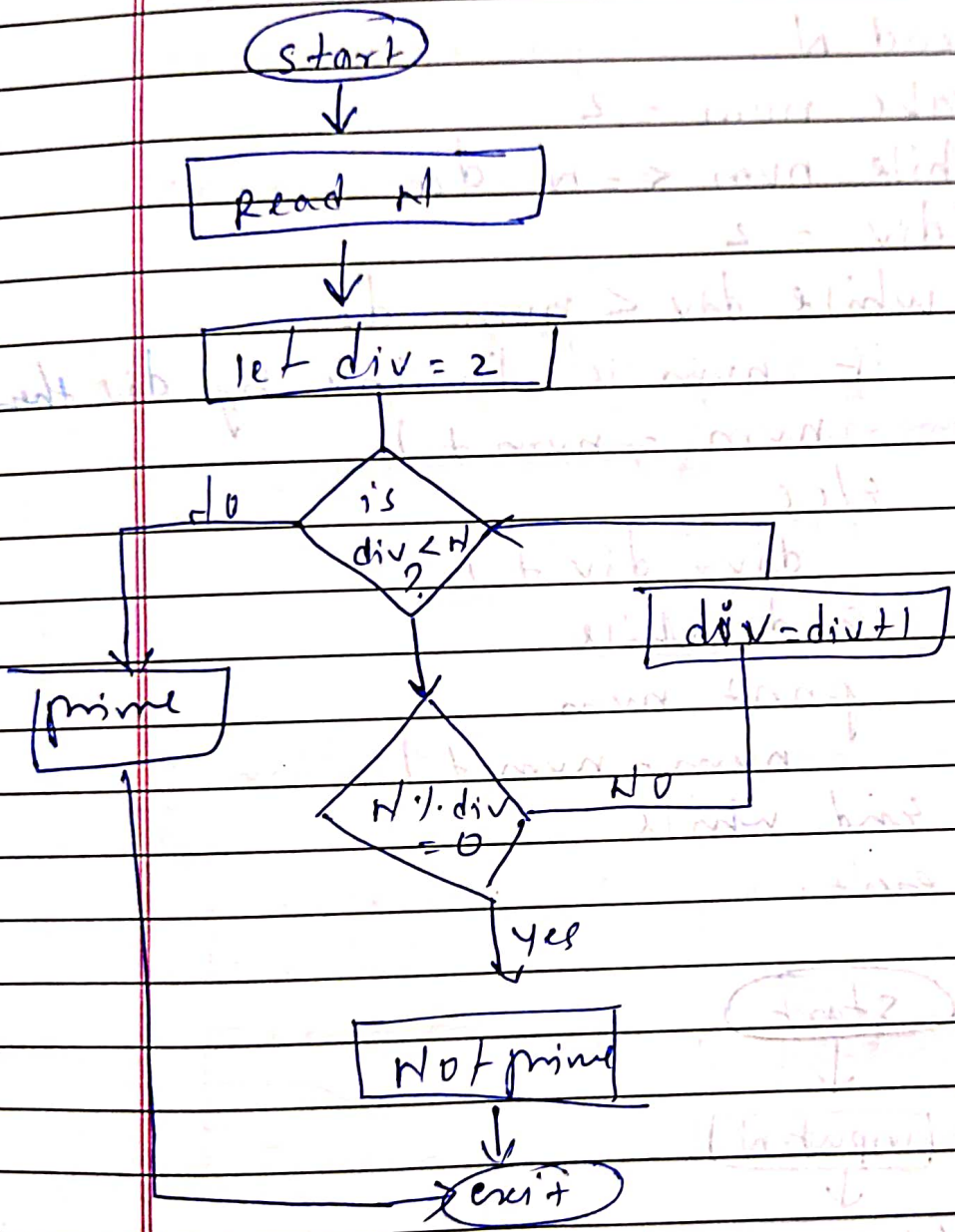


1. start
2. input 2 no.
3. calc sum = n1 + n2
4. print sum
5. exit.

Q.3. find maximum of three numbers.



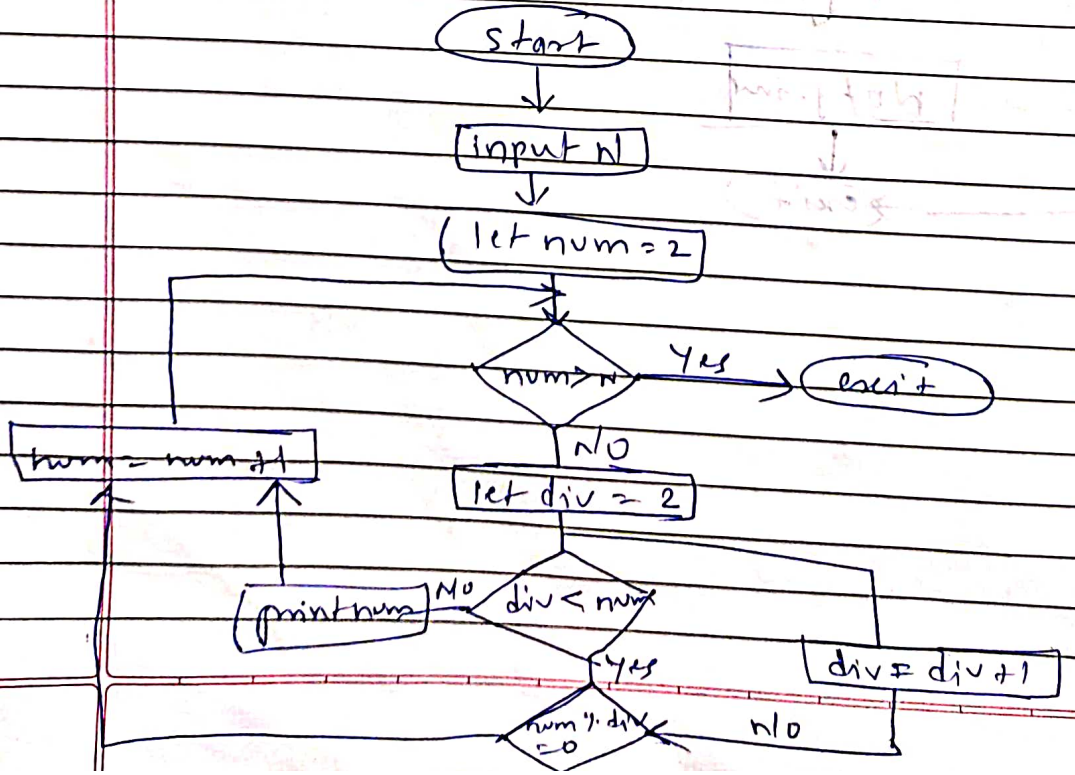
Q. if a given no. is prime or not.



1. Read N
2. Take div = 2
3. while div < N d
   if N is div. is/be by div then,
   print "Not"
   end
   else
   div = div + 1
4. print "prime"
5. exit.

Q. print all prime number till n.

1. start
2. read N
3. Take num = 2
4. while num <= N do
   div = 2
   while div < num do
   if num is divisible by div then
   num = num + 1.
   else
   div = div + 1
   end while
   print num
   num = num + 1
   end while
5. exit.



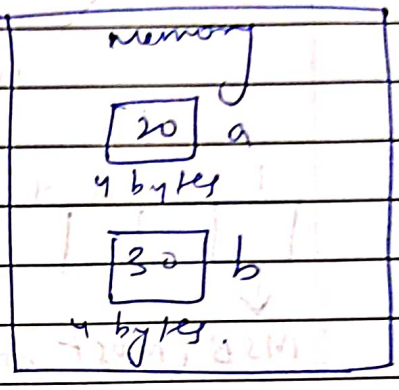
Date
Page

* datatypes.

Int - 4 byte
char - 1 byte.

ex :- avg of 2 no.

input a, b
Avg = (a+b)/2
constant
Variables



primitive	derived	user-defined
- integer	- function	- class
- float	- Array	- structure
- character	- pointer	- union
- Boolean	- Reference	- Enum.

① Int

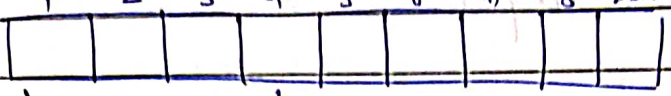
1 byte = 8 bits.
4 byte = 32 bits.

Range (unsigned) = 0 to $2^{32}-1$

Range (signed) = 2^{31} to $2^{31}-1$
(+)

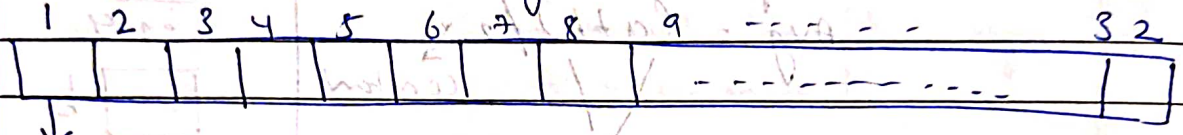
int

memory block



4 bytes

└ 1 byte = 8 bits.



MSB (Most Significant Bit) is used to tell the sign of that integer.

How to find whether a no. is pos or neg?

Ans :- ① If MSB is 1 then it is negative no.

② If MSB is 0 then it is positive no.

* Bool

Size = 1 byte.

newline

Date
Page

* float

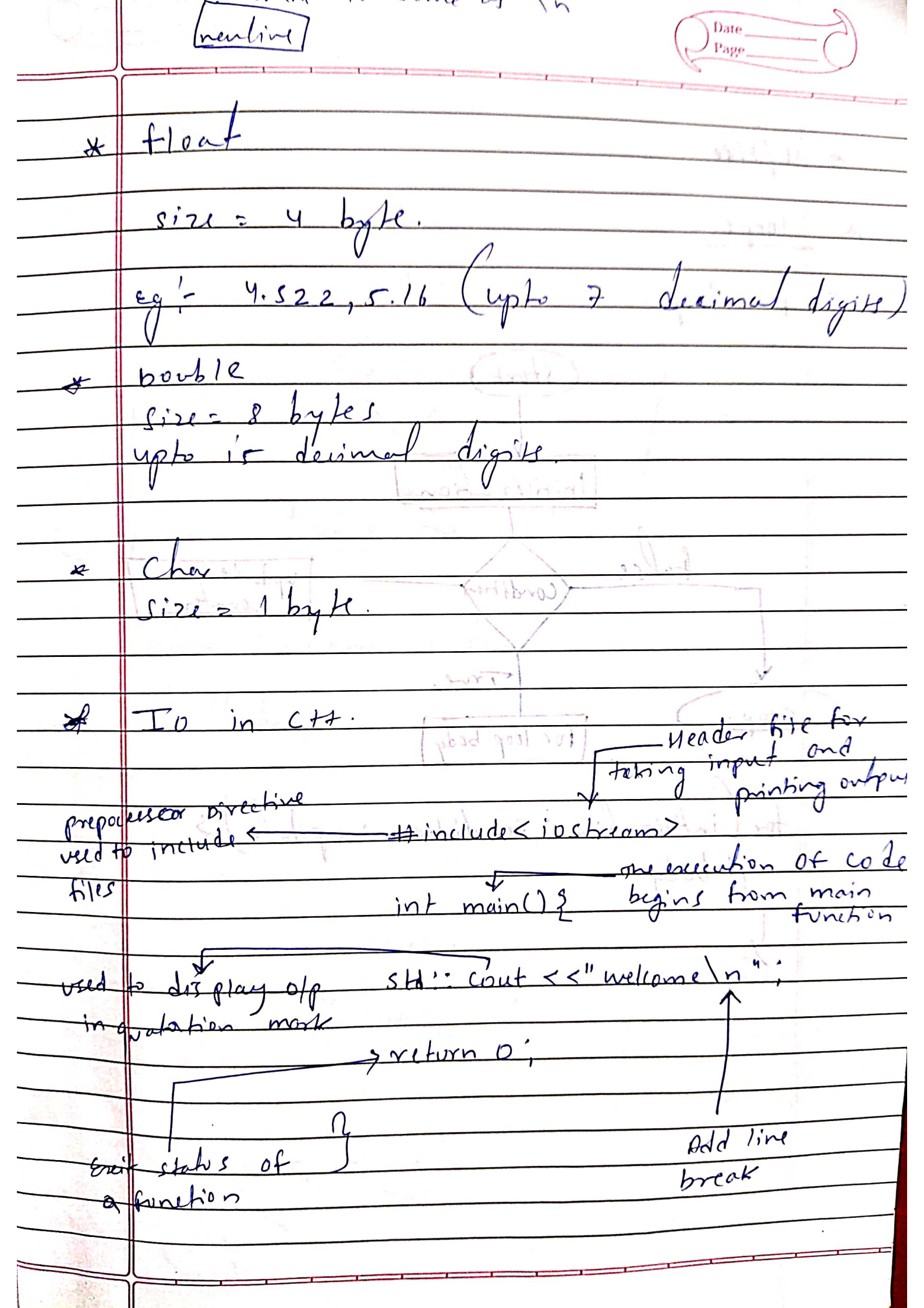
size = 4 byte.

eg:- 4.522, 5.16 (upto 7 decimal digits)

* double
size = 8 bytes
upto is decimal digits.

* Char
size = 1 byte.

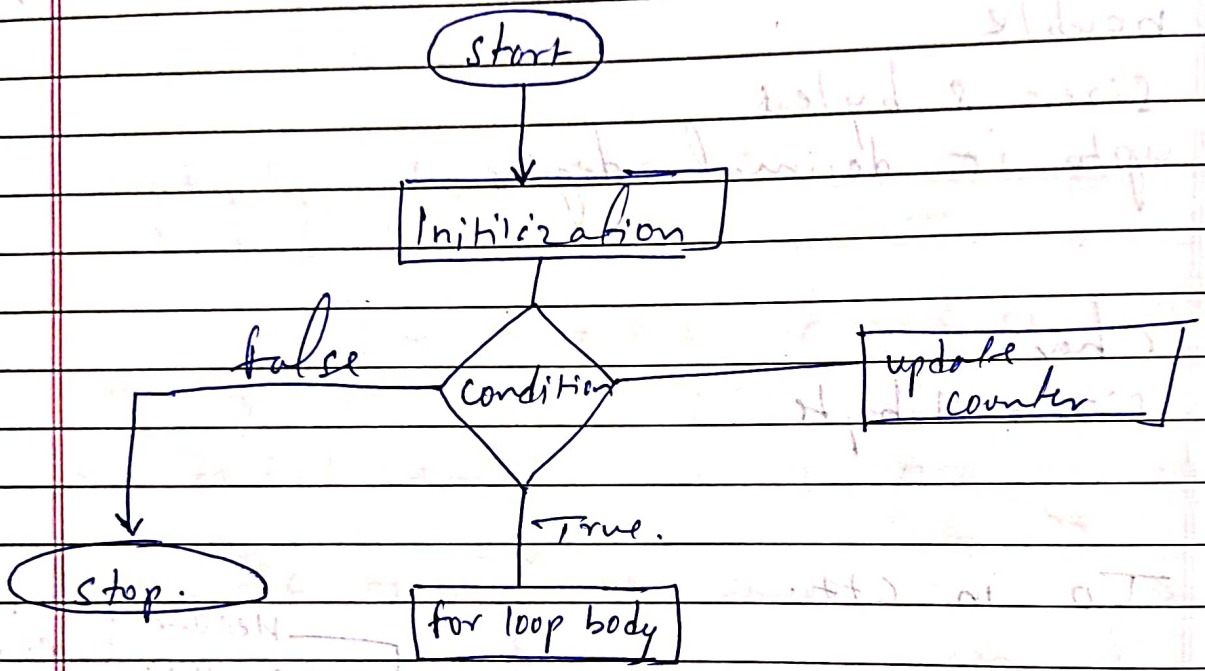
* IO in C++.



* if/else

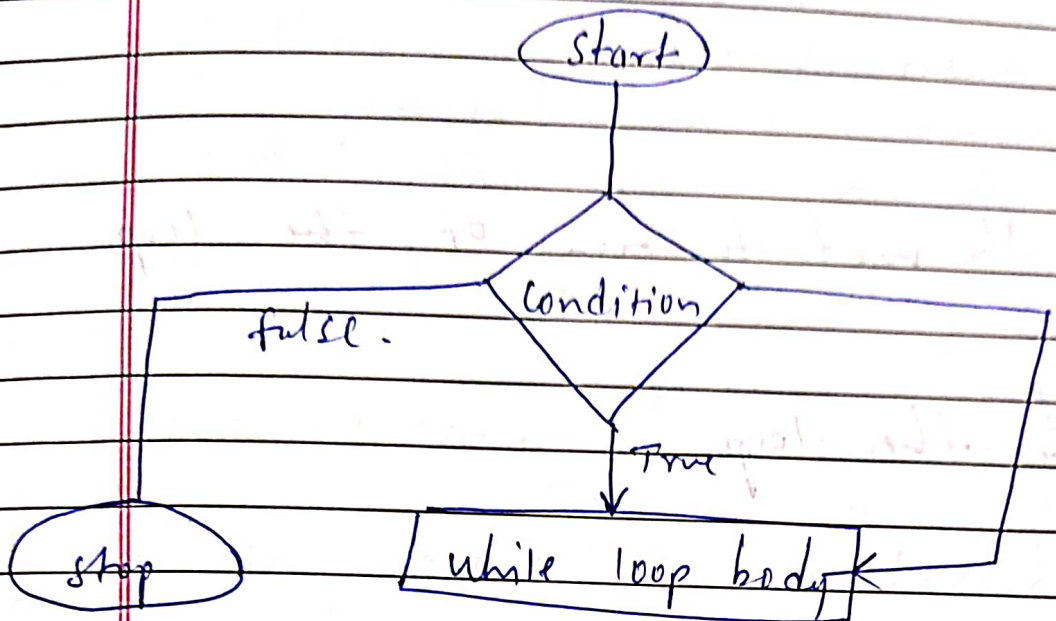
* Loops

① for loop.



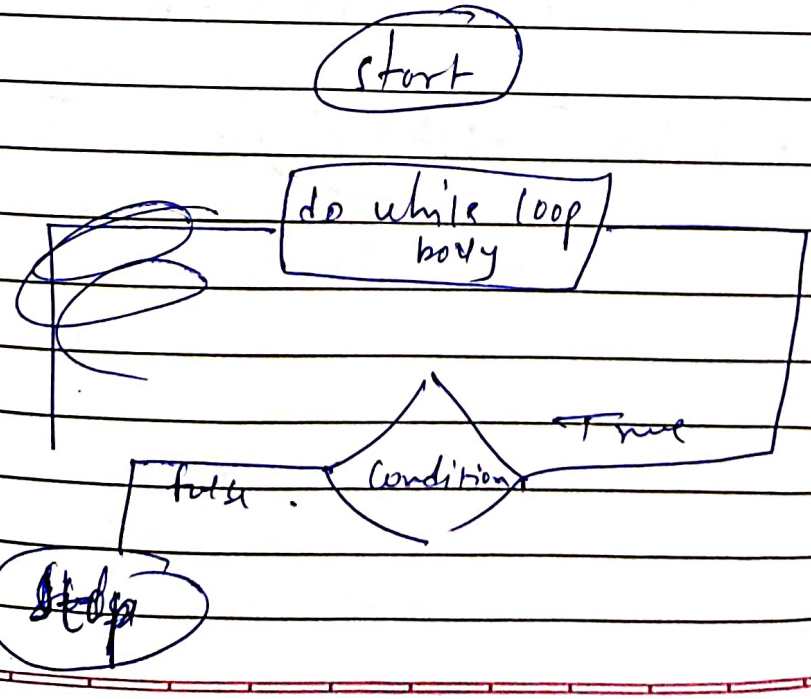

```
for (initilization ; condition ; update) {
    // body
}
```

* while loop.




```
while (condition is true)
{
  // body
}.
```

* do while.




```
do {
  // body
}while(Cond);
};
```

Date
Page

✱ Time Complexity.

① O (Big O) means best case.
② Ω (omega) means Average case.
③ Θ (Theta) means worst case.

① worst case : [O(big Oh) notation]
② Best case : [Ω (big Omega) notation]
③ Avg case : [Θ (big theta) notation]

Example :

```
① int n, m
   cin >> n >> m ;
   int a = 0;
   for(int i = 1; i <= n; i++)
   {
      a = a + 1;          ——— n
   }
   for (int j = 1; j <= m; i++)
   {
      a = a + 1;          ——— m
   }
```

O(n + m)

Date
Page

(2)
```
int n,m;
cin>>n >>m;
int a = 0;
for(int i=1; i<=n; i++)
{
   for(int j=1; j<=m; j++)
   {
      a = a + rand();
   }
}
```

i=1 → 1 to m
i=2 → 1 to m
i=3
⋮
i=n → (m)

O(nm)

* example

x = 238
x = 20

20, 10, 15, 16, 28, 238

O Worst case — x not found. = n

Ω Best case — x found.

Θ Avg case — 1 + 2 + 3 + ... n = $\frac{n(n+1)}{2}$ = $\frac{n+1}{2}$

BC — Constant
WC — ∝ n O(n)
AC — ∝ n O(n)

(3)
```
int m, n
cin>>n, m ;
int a = 0;
for (int i = 1; i<=n ; i++)
{
  for (int i=1; i<=n; i++)        ——— n
  {
    for(int j=1; j<=m; j++)       ——— m
    {
      a = a + rand();
    }
  }
for(int i=1; i<=n; i++)
{
    a=a+1;                        ——— n
}
```

$\Rightarrow O(nm + n)$

(4)
```
int n ;
Cin >> n;
int a=0; i=n
while (i>=1)
{
  a=a+1;
  i/=2;
}.
```

$n \to \frac{n}{2} \to \frac{n}{4} \to \frac{n}{2^k}$

$n \geq 2^k$

$\log n \geq k$

$k \leq \log n$

$O(\log n)$

functions.

```
int n1, n2;
cin>>n1>>n2;
int fact = 1;
for(int i = 2; i<=n; i++){
  fact = fact*i;
}
for (int i = 2;i<=n2 ;;i++{
   fact2 = fact2 * i;
}
```

same code is written to find the fact of 2 no.

So, to avoid this we use functions.

Where function data get stored

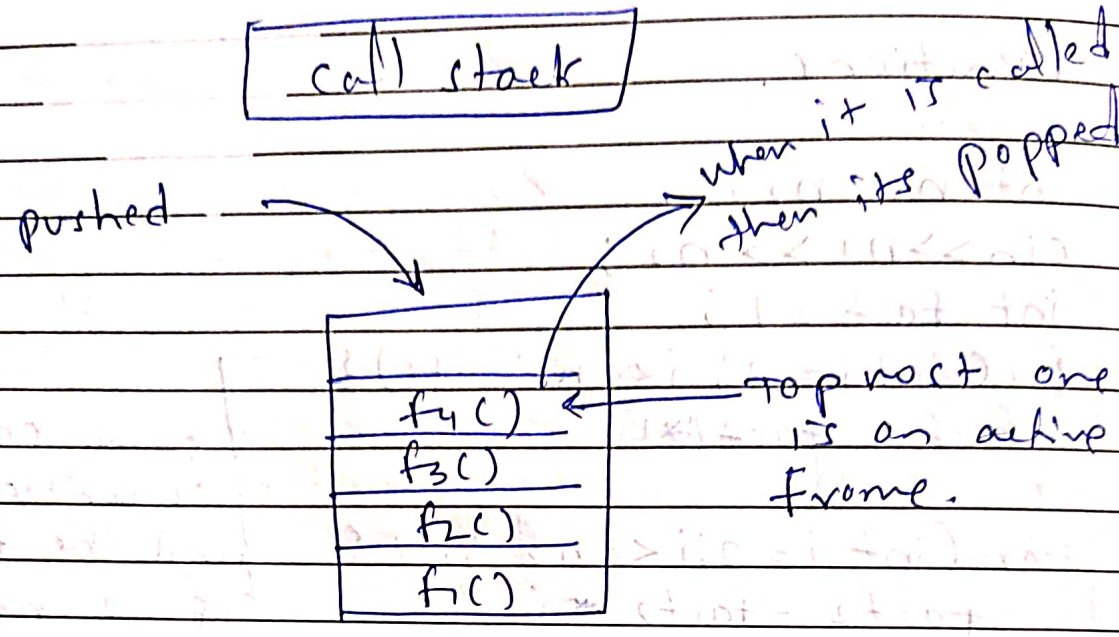
Call stack

```
void function A()
{
}
```

memory assigned →

function A

function code, local variables, parameters, etc.



Q. prime function.

```
#include <iostream>
using namespace std;
bool isprime(int n)
{
  for(int i=2 ; i<=sqrt(num) ; i++)
  {
     if(num%2==0) {
        return false
     }
  }
  return false
}
int main()
{ int a, b;
  cin>>a>>b;
```

```
for(i=a; i<=b; i++)
{
    if(isprime(i))
    {
        cout<<i;
    }
}
}
```

fibonacci series.

```
#include <iostream>
using namespace std;
void fib(n)
{
    int t1 = 0;
    int t2 = 1;
    int nextTerm;
    for(int i=1; i<=n; i++)
    {
        cout << t1,
        nextTerm = t1 + t2;
        t1 = t2
        t2 = nextTerm;
    }
    return 0;
}
int main()
{
    int a;
    cin >> a
    fib(a)
```

* factorial

```
#include <iostream>
using namespace std;
int fact(int n)
{
int fat = 1;
for(int i=2; i<=n; i++)
  {
    factorial *= i;
  }
 return factorial;
}
int main()
{ int n;
  cin>> n;
  cout<<fact(n);
}
```

* calculate $nCr = \frac{n!}{(n-r)! \times r!}$

imp

(1) largest of two no. without using any conditional stmt or operators. ( )

```
#include<iostream>
using namespace std;
int largestnum(int a, int b)
{
    return a * (bool)(a/b) + b * (bool)
                (b/a)
}
int main()
{
  int a = 22, b = 123;
  cout << largestnum(a,b);
  return 0;
}
```

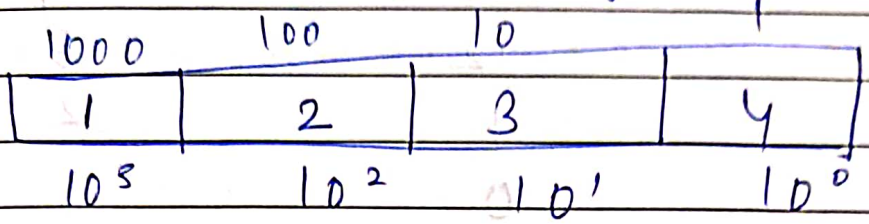
Q. sum of n natural no. until it becomes single digit.

```
#include<iostream>
using namespace std;
int digSum(int n)
{
    int sum = 0;
    while(n>0 || sum>9)
    {
        if(n==0)
        {
            n=sum;
            sum=0;
        }
        sum = sum + n%10; OR sum += n%10;
        n=n/10;
    }
    return sum
}
int main()
{
    int n=1234;
    cout<< digSum(n);
    return 0;
}
```

* ~~Practice Questions~~

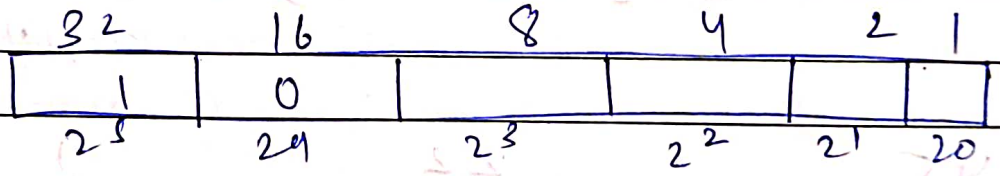
* Binary Number System.

① Decimal number system.



$1234 = 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$

② Binary number system.



$45 = 1 \times 32 + 0 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1$

$45 = 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

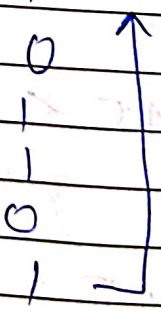
Date
Page

Decimal to Binary.

N	Quotient (N/10)	Remainder (N%10)
1234	123	4
125	12	3
12	1	2
1	0	1

Decimal to Binary.

N	Quotient (N/2)	Remainder (N%2)
45	22	1
22	11	0
11	5	1
5	2	1
2	1	0
1	0	1



* Binary to Decimal.

1	0	1	1	0	1
2^5	2^4	2^3	2^2	2^1	2^0

$1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^1$

$32 + 0 + 8 + 4 + 0 + 1 = 45$.

* Reverse a number.

```
#include <iostream>
using namespace std;
int rev(int n)
{
   int rev, lastdigit;
   while (n>0)
    {
      lastdigit = n %10;
      rev = rev *10 + lastdigit;
      n= n/10;
    }
   return rev;
}
int main()
{
   int n;
   cin>>n;
   cout<<reverse(n);
```

* Armstrong.

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
    int sum = 0;
    int original = n;
    int n;
    cin>>n;
    while(n>0)
    {
        int lastdigit = n %10;
        sum = sum + pow(lastdigit, 3);
        n = n/10;
    }
    if(sum == original)
    {
        cout << "Armstrong";
    }
    else
    {
        cout << "Not";
    }
return 0;
}
```

Triplet Pythagoral Theorm.

```
#include <iostream>
#include <math.h>
using namespace std;
bool triplet(int x, int y, int z)
{
    int a = max(x, max(y,z));
int b, c;
    if(a==x)
    {
        b=y;
        c=z;
    }
elseif(a==y)
{
    b=x;
    c=z;
}
else{
    b=x;
    c=y;
}

if(a*a = b*b + c*c)
{
    return true;
}
else
    return false
```

```
int main()
{
int x, y, z;
cin>>x>>y>>z;
if(triplet(x,y,z))
{
return 4
cout<<"triplet
}
else
    not triplet.
```

* octal to decimal.

{0, 1, 2, 3, 4, 5, 6, 7}

Representation : $(x)_8$

eg :- $(137)_8$

Converting to decimal

$(137)_8 = 7 \times 8^0 + 3 \times 8^1 + 1 \times 8^2$

$= 7 + 24 + 64$

$= (95)_{10}$

Algo :- Traverse over the digits and make the decimal number.

* Hexadecimal to decimal.

Base : 16

{0, 1, 2, ---- 9, A, B, C, D, E, F}

A = 10 B = 11

C = 12 D = 13

E = 14 F = 15

eg:- $(1CF)_{16}$

Converting to decimal.

$(1CF)_{16} = 15 \times 16^0 + 12 \times 16^1 + 1 \times 16^2$

$= 15 + 192 + 256$

$= (463)_{10}$

Same Algo.

* Decimal to octal

Eg:- $(100)_{10} = (x)_8$

$(100)_{10} = 1 \times 8^2 + 4 \times 8^1 + 4 \times 8^0$

$= (144)_8$

Algo:- find the highest power of 8, from which our number is divisible, then reduce the number do this until our number becomes 0.

* Decimal to hexadecimal.

$(100)_{10} = (x)_{16}$

$(100)_{10} = 6 \times 16^1 + 4 \times 16^0$

$= (64)_{16}$.

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    cin>> n;
    cout<< BinaryTODecimal(n);
}
int BinaryTODecimal(int n)
{

int ans = 0
int x = 1
 while (n >0)
  {
      int y = n % 10;
      ans = ans + (x * y);
      x = x * 2;
      n = n/10;
  }
  return ans
}
```

Date
Page

Arrays

program to find min and Max from Array.

```
#include<iostream>
#include<limits>
using namespace std;
int main()
{
    int n;
    cin>>n;
    int arr[n];
   for(i=0;i<n;i++)
    {
    cin>>arr[i];
    }
  int max = INT_MIN;
  int min = INT_MAX;

  for(i=0;i<n;i++)
   {
     if (arr[i] > max)
      {
       max = arr[i];
      }
     if (arr[i]<min)
      {
       min = arr[i];
      }
   }
  cout<<max;
} cout<<min;
```

Alternative

```
MAX = max(max, arr[
min = min(min, arr[i])
```

HOK:- Greatest no. of

Date
Page

* sum of array.

```
#include<iostream>
using namespace std;
int main()
{
  int n;
  cin>>n;
  int arr[n];
  for(int i=0;i<n;i++)
  {
    cin>>arr[i];
  }
int sum=0;
for(int i=0;i<=n;i++)
  {
    sum+=arr[i];
  }
cout<<sum;
return 0;
}
```

* linear search O(n)

```
#include <iostream>
using namespace std;
int linearSearch (int arr[], key, n)
{
   for(int i=0;i<n;i++)
   {
      if (arr(i) == key)
       return i;
   }
  return -1;
}
int main ()
{
   int n;
   cin>>n;
   for(int i=0; i<n; i++)
   {
    cin>> arr(i);
   }
  int key;
  cin >> key;

  cout << linearSearch (arr, key, n)

}.
```

Q. Binary Search.

mid = key.

1, 6, 8, 9, 13, 14, 16.

0 1 2 3

e = 3 - 1 = 2.

```
#include <iostream>
using namespace std;
int BinarySearch(int arr, int key, int n)
{
    int s = 0;
    int e = n;
    while(s <= e)
    {
        int mid = (s+e)/2;
        if(mid == key)
            return mid;
        else if(arr[mid] < key)
            e = mid - 1;
        else if(arr[mid] > key)
            s = mid + 1;
    }
    return -1;
}
int main()
{
    //some code of linear search.
}
```

Time Complexity of Bc

— n
— n/2
— (n/2)/2
— $n/2^k$

$O(\log_2 n)$

$n/2^k = 1$
$n = 2^k$
$\log_2(n) = \log_2(2^k)$
$\log_2(n) = k \log_2 2$
$k = \log_2 n$

* Puzzle

① you have 15₹
② 1 rupee = 1 choclate
③ 3 wrappers = 1 choclate

Ans : 22

* selection sort.

```
int main()
{
    int n;
    cin >> n;
    for(int i=0; i<n; i++)
    {
        cin>>a[i];
    }
```

```
for(i=0; i<n-1; i++) {
    for(j=i+1; j<n; j++)
    {
        int temp = a[j]
        a[j] = a[0]
        a[i] = temp;
    }
    {
        if (arr[j] < arr[i])
        {
            int temp = a[j];
            a[j] = a[i];
            a[i] = temp;
        }
    }
}
for(i=0; i<n; i++)
{
    cout<< arr[i] << endl;
}
}
```

* Bubble sort.

(1) 12, 45, 23, 51, 19, 8
12, 45, 23, 51, 19, 8
12, 23, 45, 51, 19, 8
12, 23, 45, 19, 51, 8
12, 23, 45, 19, 8, 51

(2) 12, 23, 19, 45, 8, 51
12, 23, 19, 8, 45, 51

(3) 8, 12, 19, 23, 45, 51
8, 12, 19, 23, 45 51

Note:- Comparing two elements and swapping it.

* Bubble Sort

```
int main()
{
    // Taking array as input

    counter = 1
    while (counter < n)
    {
        for(int i=0; i < n - counter; i++)
        {
            if(a[i] > a[i+1])
            {
                temp = a[i];
                a[i] = a[i+1];
                a[i+1] = temp;
            }
        }
        counter++;
    }
    for(int i=0; i<n; i++)
    {
        cout << a[i] << endl;
    }
}
```

Insertion Sort

12, 45, 23, 51, 19, 8

12, 45, 23, 51, 19, 8

12, 23, 45, 51, 19, 8

12, 23, 45, 51, 19, 8

12, 23, 45, 51, 8

8, 12, 19, 23, 45, 51

```
#include<iostream>
:
:
//input array
for(int i=0; i<n; i++){
int current = arr[i]
  j = i-1;
   while (arr[j] > current && j>=0) {
      arr[j+1] = arr[j];
      j--;
   }
   arr[j+1] = current i
}
```

Array challenges

problem ①

Given an array a[] of size n. for every i from 0 to n-1
output max(a[0], a[1] ---- a[i]).

example:

1	0	5	4	6	8

max till i: 1 1 5 5 6 8

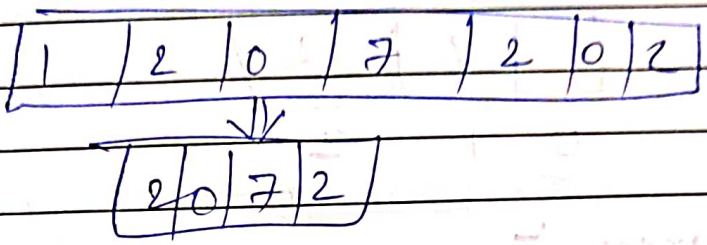
```
#include <iostream>
#include <climits>
using namespace
int main
{ int mx = INT_MIN;
  // input array

  for(int i=0; i<n; i++)
  {
    mx = max(mx, arr[i])
    cout << mx << endl;
  }
}
```

Date
Page

Array challenge 2

subarray VS subsequence.

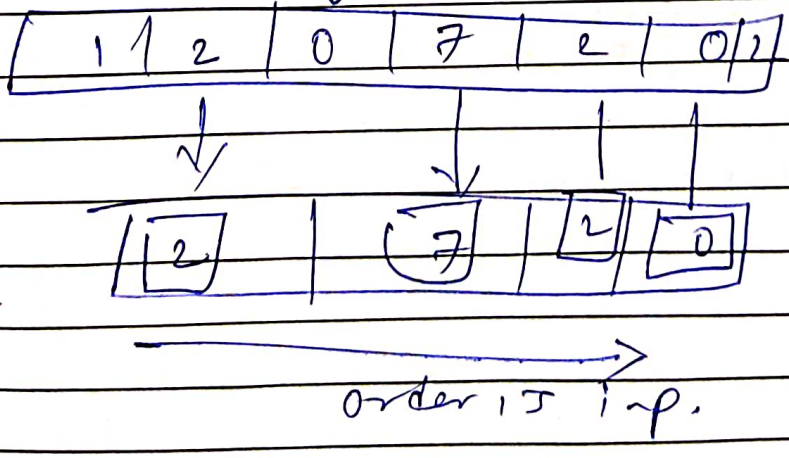
subarray
- continuous part of the array.



No. of subarrays of an array with n elements = $nC_2 + n = n*(n+1)/2$

Subsequence.

A subsequence is a sequence that can be derived an array by selecting zero ~~or~~ more elements, without changing the order of the remaining elements.

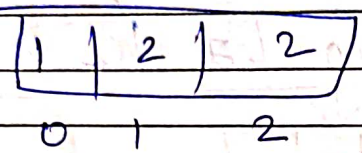


No. of subsequence in an array = 2^n

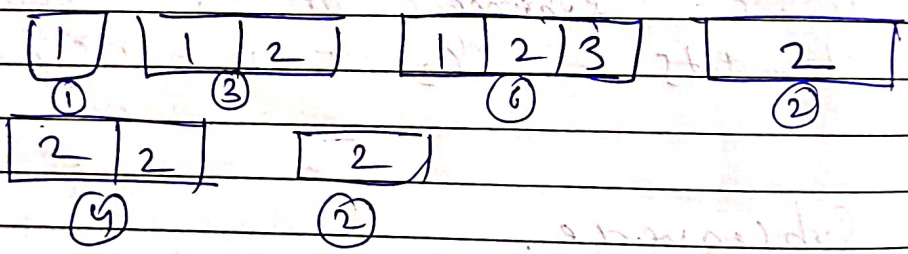
Q. Sum of all subArray.

Given a array[] of size n, output sum of each subarray of the given array.

Example :



subarray :-



sum

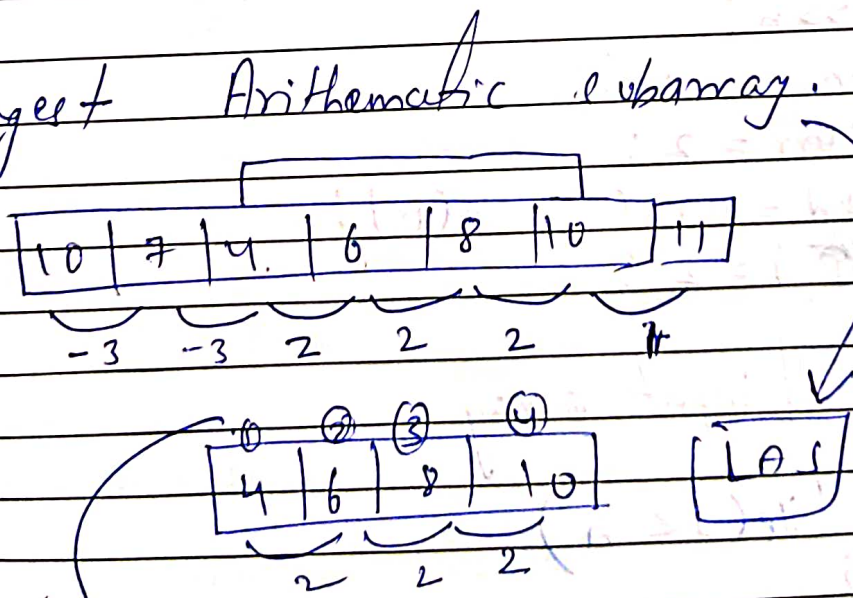
Approach :- ① iterate all over subarrays

②

Code :-

```
int main
{
  // input array
  int curr = 0;
  for(int i=0; i<n; i++)
  {
    Curr = 0;
    for(j=i; j<n; j++)
    {
      curr += arr[j]
      cout<< curr<< endl;
    }
  }
}
```

Q. longest Arithemetic subarray.



O/P :- 4 elements

Approach

(1) Loop over the array and find the answer.
Maintain the following variables

① previous common Diff (Pd)

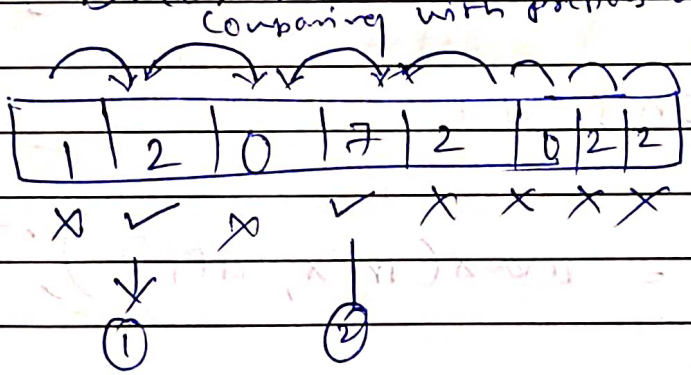
② current Arithmetic Subarray length (curr)

③ Max arithmetic subarray length (ans)

```
int main()
{
    int n, i, arr[n], ans, curr;
    cin>>n;
    ans = 2;
    curr = 2;
    Pd = arr[1] - a[0];
    i = 2;

    // input array
    while(i < n)
    {
        if( Pd == a[i] - [i-1]
        curr++;
```

else {
arr[i] - arr[i-1]
curr = 2
}
ans = max(ans, curr);
i++
}
cout << ans;
}.

* Record Breaker.

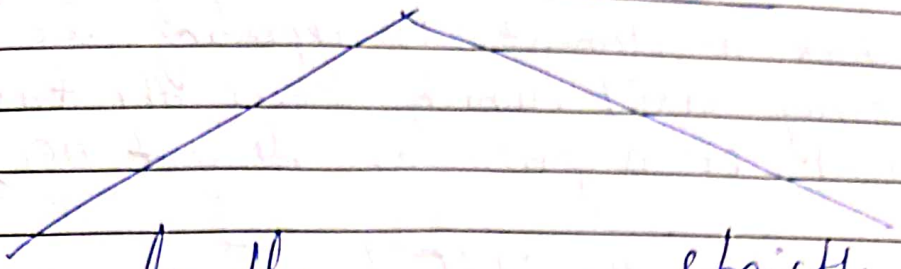


#include<iostream>
using namespace std;
int main()
{
int n;
cin>> n;

int a[n+1];
a[n] = -1;
for(int i = 0; i<n; i++)
{
cin>> a[i];
}

```
if(n==1)
return 0;
{
  cout << "1" <<endl;
  return 0;
}

int ans = 0;
int mx = -1;

for(int i=0; i<n; i++)
{
  if(a[i] > mx && a[i] > a[i+1])
  {
    ans++;
  }
  mx = max(mx, a[i]);
}

cout << ans << endl;
return 0;
}
```

Record Breaking Day.



strictly greater than all the previous values

strictly greater than following value.

Q. First Repeating Element.

example.

Input:
7
1 5 3 4 3 5 6

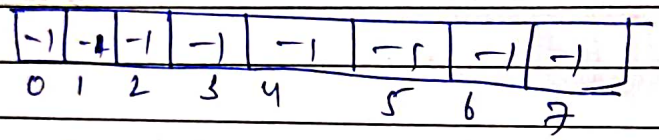
o/p:-
2

Exp:- 5 is appearing twice and its first appearance is at index 2 which is less than 3 whose first occuring index is 3.

Basic idea:-

To check if element is repeating, we maintain an array idx[], which stores the first occurence (index) of a particular element a[i].

(i) Initialize the idx[] with -1 and minidx with INT_MAX.



keep updating idx[] while traversing array

Given array: 1 5 3 4 3 5 6

At i = 0

A :- 1 5 3 4 3 5 6
↑

idx[] -1 0 -1 -1 -1 -1 -1 -1
0 1 2 3 4 5 6

At i = 1

1	5	3	4	3	5	6

minidx = Max_INT

-1	0	-1	-1	-1	1	-1	-1
0	1	2	3	4	5	6	7

At i = 2

1	5	3	4	3	5	6

-1	0	-1	2	-1	1	-1	-1
0	1	2	3	4	5	6	7

At i = 3

1	5	3	4	3	5	6

minidx = ~~MAX_INT~~

-1	0	-1	2	3	1	-1	-1
0	1	2	3	4	5	6	7

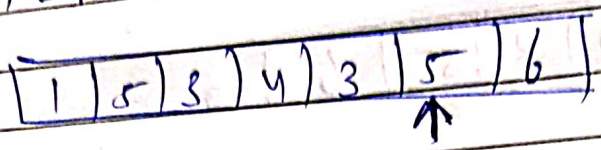
At i = 4.

1	5	3	4	3	5	6

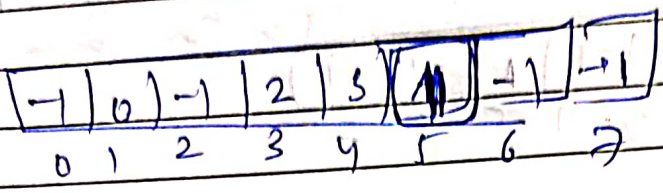
Min-idx = 2.

-1	0	-1	2	3	1	-1	-1
0	1	2	3	4	5	6	7

At i = 5



min-idx = 1

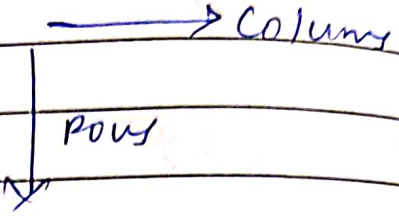


Q.

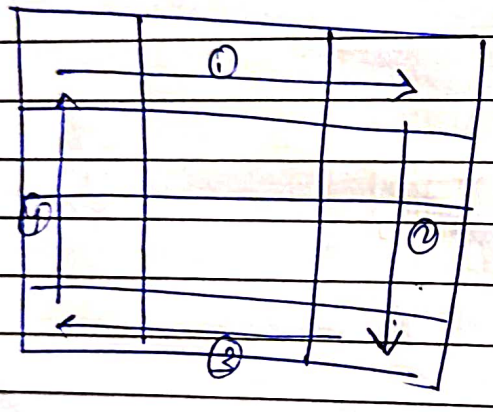
* 2-dimensional Array.

Declaration :

int arr[m][n];



Q.



① row-start + 1
② column-end - 1
③ row-end - 1
④ column-start + 1

row-start < row-end

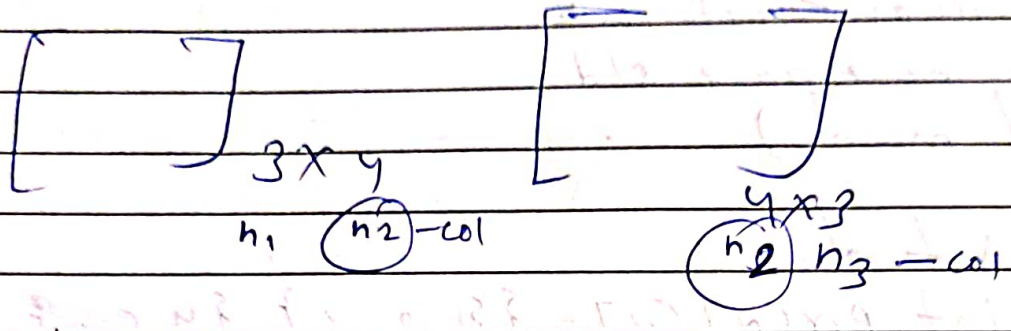
column-start < column-end.

Date
Page

* Matrix Transpose

```
#include <iostream>
using namespace std;
int main()
{
    int N = 3;
    int arr[N][N] = {{1,2,3},{4,5,6},{9,8,7}};
    for(int i=0; i<N; i++)
    {
        for (int j=i; j<N; j++)
        {
            int temp = arr[i][i];
            arr[i][i] = arr[j][i];
            arr[j][i] = temp;
        }
    }
for(int i=0; i<N; i++)
{
    for (int j=0; j<N; j++)
    {
        cout << arr[i][i];
    }
    cout << "\n" << endl;
}
return 0;
}
```

Matrix multiplication



ans += m1[i][k] * m2[k][j];

```
int main()
{
    int n1, n2, n3
    int m1[n1][n2];
    int m2[n2][n3];
    int ans[n1][n3]
    for {
        // input m1;
    }
for {
    // input m2;
}
```

```
for {
    input ans = 0;
}
for(int i=0; i<n1; i++)
{
  for(int j=0; j<n2; j++)
  {
    for(int k=0; k<n2; k++)
    {
      ans[i][j] += m1[i][k] * m2
                   [k][j];
    }
  }
}
return 0; for {
cout<<  cout<<ans
    }
}
```

511 2 3 4 6 1 4 4

Page

Matrix Searching.

① Brute force approach

```
#include <iostream>
using namespace std;
int main()
{
    int n, m, key;
    for(----)
    {
      //input array
    }
//Enter key
for(int i=0; i<m; i++)
  {
    for(int j=0; j<m; j++)
      {
          if(arr[i][i] == key)
          found = true
      }
}
  if (key found)
    {
        cout << found";
    }
  else
    {
      not found.
    }
```

* Character Array.

① print palindrome

```
#include <iostream>
using namespace std;
int main()
{
  int ....
  for {
      Input array
  }
bool check = true;
for(int i=0; i<n; i++)
 {
    if(arr[i] != arr[n-1-i])
    {
       check = false;
    }
 }
if (check)
  {
    return palindrome;
  }
else{
  return not palindrome;
}.
```

Q. longest word in an array.

```
using namespace std;
int main()
{
    int n;
    cin >> n;
    cin.ignore();
    char arr[n+1];
    cin.getline(arr, n);
    cin.ignore();

    int i=0;
    int curLen = 0;
    int maxLen = 0;

    while(1)
    {
        if(arr[i]==' ' || arr[i]=='\0')
        {
            if (curLen > maxLen)
            {
                maxLen = curLen;
            }
            curLen = 0;
        }else
        curLen++;
        if(arr[i]=='\0')
            break;
```

```
        i++;
    }
    cout << maxLen << endl;
    return 0;
}
```

* **Pointers**
  are the variables that store the address of the another variable.

Example)

```
int main(){
    int a = 10;
    int *aptr;
    aptr = &a;

    cout << &a << endl;      //2000
    cout << aptr << endl;    //2000
    cout << *aptr << endl;   //10

    return 0;
}
```

* Pointer Arithmetic (++, --, +, -)

Example ①

```
int main(){
    int a=10;
    int *aptr = &a;
    cout <<aptr <<endl;  //2000

    aptr++;
    cout <<aptr <<endl;  //~~2001~~ 2004

    return 0;
}
```

Because size of the int is 4 bytes.

Example ②

```
int main()
{
    ~~int~~ char a='a';
    char *cptr = &a;
    cout << cptr <<endl;  // 300

    cptr++;
    cout << cptr << endl;  // 301

    return 0;
}
```

Pointers and Arrays.

// printing arr using pointer.

```
int arr = {20, 20, 30, 40};
int *ptr = arr;
for(int i=0; i<3; i++)
  {
    cout << *ptr << endl;
    ptr++;
  }
return 0;
}
```

pointer to pointer.

```
int main (){
  int a = 10;
  int *p;
  p = &a;

  cout << *p << endl; // 10
  int **q = &p;

  cout << *q << endl; // 2000
  cout << **q << endl; // 10

  return 0;
```

2000	a = 10
4000	p = 2000
4200	q = 4000

* Stack and Heap.

Stack memory allocation.
① The memory is allocated on the function call stack. The memory gets deallocated as soon as the function call gets over. deallocation is handled by the compiler.

Heap Memory Allocation.
① Allocation takes place on the pile of memory space available to programmers to allocated and de-allocate. The programmer has to handle the de-allocation.
Note:- It is diff from heap data-structures.

```
int main
{
    int a = 10 // stored in stack
    int *p = new int(); // allocated memory in heap
    *p = 10;
    delete(p);          // DeAllocated memory.
    p = new int[4];
    delete [] p;
    p = NULL;
    return 0;
}
```

* strings.

Char Array	Strings
① Need to know the size beforehand.	① No need to know size beforehand
① larger size required for operation	② performing string operation easier.
② No terminating extra character.	③ Terminating with a special character '\0'

* String Challenges.

multiple will start from i x i = (i^2)

* Sieve Of Eratosthenes.

* suppose there are 50 integers 1 to 50
* step 1 :- multiple of 2 is marked
* step 2 :- multiple of 5 is marked
* step 3 :- multiple of 7 is marked.

```
#include<iostream>
using namespace std;
int primesieve(int n)
{
    int arr[100] = {0};
    for(int i = 2; i<=n; i++)
    {
        if(arr[i]==0)
        for(j=i*i; j<=n; j+=i)
        {
            if(arr[j]==1)
            arr[j]==1;
        }
    }
    for(int i=2; i<=n; i++)
    {
        if(arr[i]==0)
        {
            cout<<i<<endl;
        }
    }
void main()
{
    int n;      primesieve(n);
    cin n;
}
```

* prime factorization using sieve.

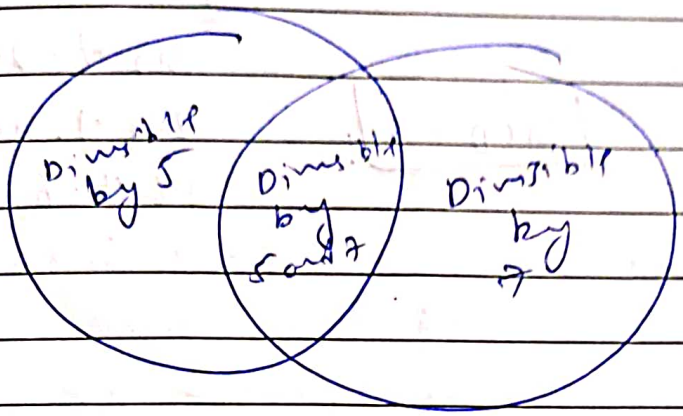
```
void primefactor(int n) {
  int spf[100] = {0};
  for(int i=2; i<=n; i++)
  {
    spf[i] = i;
  }
  for(int i=2; i<=n; i++)
  {
    if(spf[i] == i) {
      for(int j = i*i; j<=n; j+=i)
      {
        if(spf[j] == j) {
          spf[j] = i;
        }
      }
    }
  }
  while(n != 1) {
    cout << spf[n] << " ";
    n = n/spf[n];
  }
}
```

main

input n

* Inclusion - Exclusion

Q. How many numbers between 1 and 1000 are divisible 5 or 7




```
int divisibleInt (int n, int a, int b)
{
  int c1 = n/a;
  int c2 = n/b;
  int c3 = n/(a*b);

  return c1+c2-c3;
}

int main()
 {
   int n;
   cin n;
   int a, b;
   cin>> a, b> b;

   countdivisible(n, a, b)
```

Euclid Algorithm. (GCD)

$24 = 2 \times 2 \times 2 \times (3) \quad = 6$

$42 = (2) \times 3 \times 7$

$42 - 24 = 18$

$24 - 18 = 6$ **GCD = 6.**

$18 - 6 = 12$

$12 - 6 = 6$

$(6) - 6 = 0$

another method

$42 \% 24 = 18$

$24 \% 18 = 6$

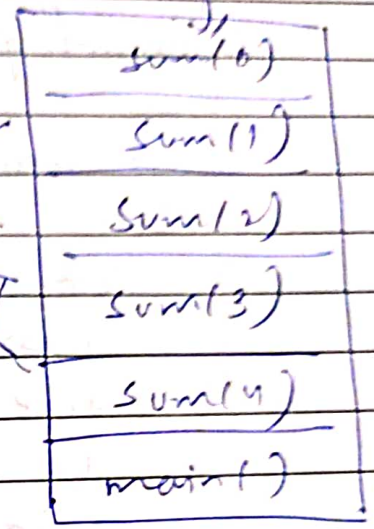
$18 \% 6 = 0$

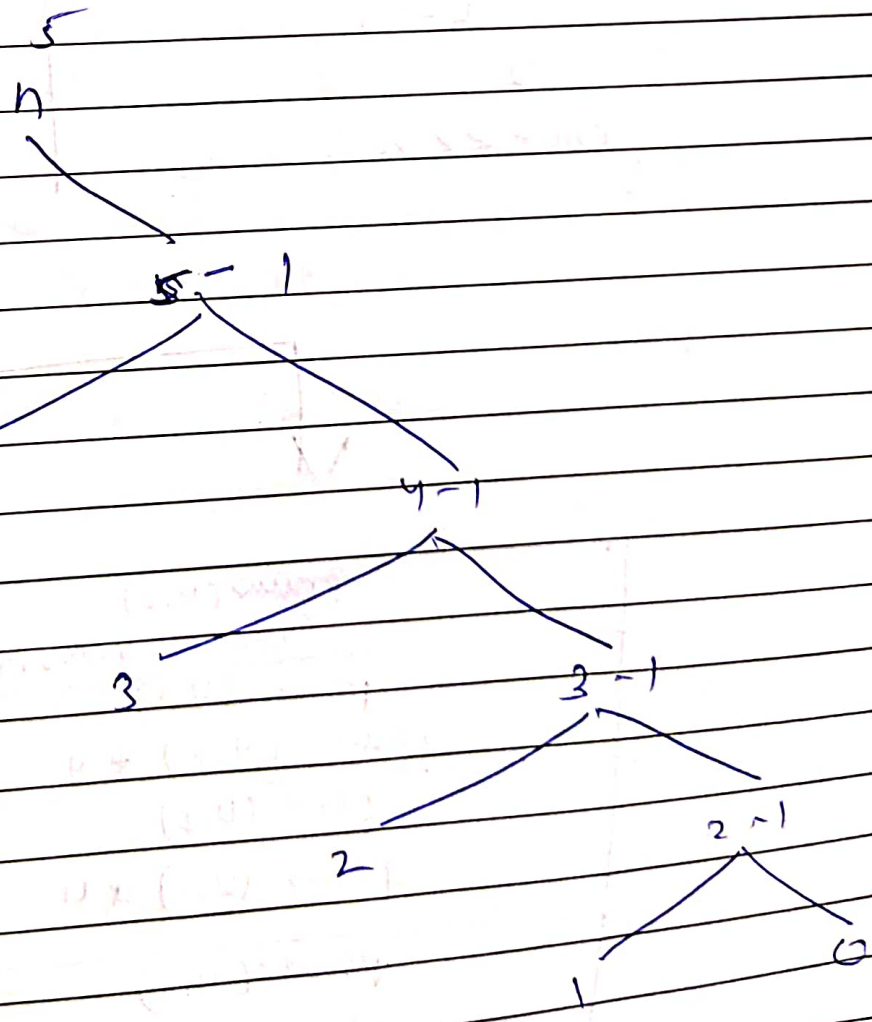
```
int gcd (int a, int b)
{   while(b != 0) {
      int rem = a % b;
      a = b;
      b = rem;
    }
    return a;
}
```

Recursion

* Sum of all n natural numbers.

```
int sum(int n) {
    if(n==0)
        return 0;
}
int prevSum = sum(n-1)
    return n + prevSum;
```



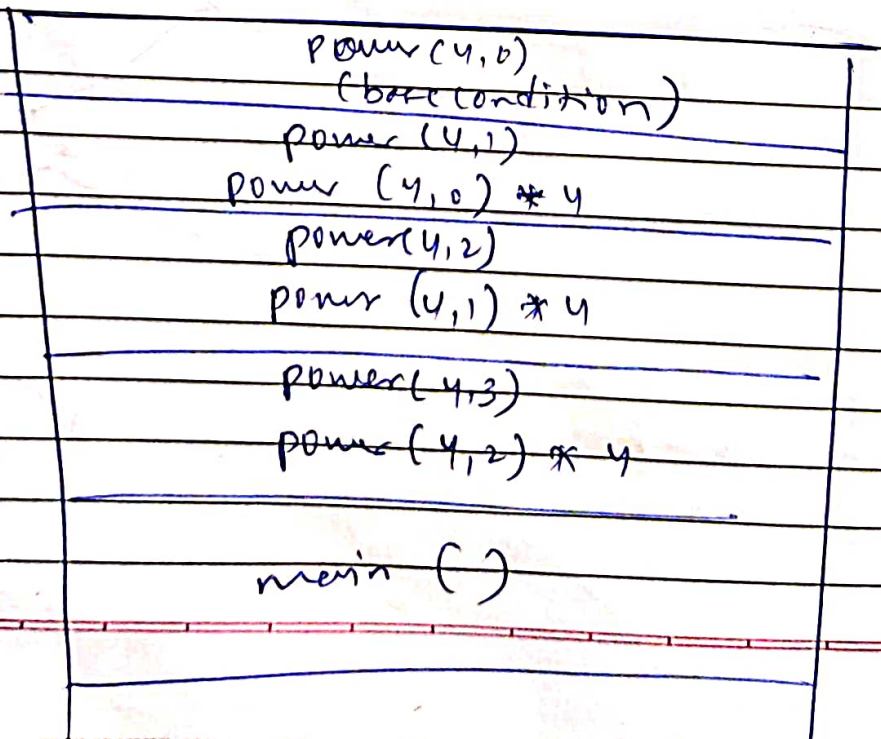


* calculate n raised to power of p.

→ normal program without recursion.

```
int main()
{
    int a, b;
    cin >> a >> b;
    while (b != 0)
    {
        a *= b;
        b--;
    }
    cout << a;
}
```

```
int power(int n, int p
{
    if (p == 0)
        return 1;
    int prev = power(n,
    return n * prev;
}
```



* Recursion (factorial)

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

* fibonacci Series (find nth number)

```
int fibtion(int n)
{
  if(n==0 || n==1)
    return n

  return fib(n-1) + fib(n-2)
}
```

Recursion Question

① Check if array is sorted or not.

```
#include <iostream>
using namespace std;
int
bool sorted (int arr[], int n)
{
    if(n == 1) {
      return true
    }
    bool restArray = sorted(arr+1, n-1);
    return (arr[0] < arr[1] && restArray);
}
int main ()
{
    cin int arr[]={1, 2, 3, 4, 5}
        cout << sorted (arr, 5);

        return 0;
}.
```

5 ... n-1 = 4

0
[1, 2, 3, 4, 5]
arr

rest arr =

```
int arraysort (int arr[], int n)
{
    if(n==1 || n==0)
    return 1;

    if(arr[n-1] < arr[n-2])
    return 0;

    return arraysort (arr, n-1);
}

int main()
{
    //input array
    if (arraysort (arr, n)
    {
        cout << "yes";
    }
    else
        cout << "not";
```

→ Asc and Dec order number printing.

```
#include <iostream>
using namespace std;
void inc(int n)
{
    if(n==1){
        cout<<"1"<<endl;
        return;
    }
    inc(n-1)
        cout<<n<<endl;
}
void dec(int n)
{

    // some base condition
}
    cout<<n<<endl;
    dec(n-1);
}
```