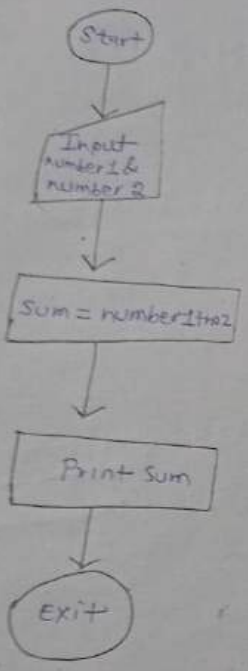
class - 1

Flowchart



Psudocode (सूडोकोड)

1. Start
2. Input 2 numbers
3. Calculate Sum = number1+number2
4. Print Sum
5. Exit

(Psudocode का मतलब है की हमें flow होगा है Program का उसको Simple English की भाषा में Represent करने का तरीका होता है)

Note :

$-\infty$ = Integer.MIN_VALUE

$+\infty$ = Integer.MAX_VALUE

Program:

```
import java.util.Scanner;
public class first {
    public static void main(String args[]) {
        System.out.println("Hello world");
    }
}
```

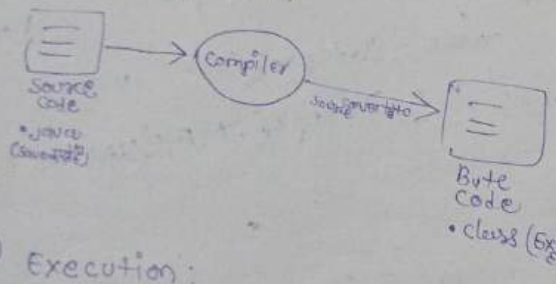
(capital letter s)

output

Helloworld

Q. How is my code running? (Java program)

Ans (i) Compilation

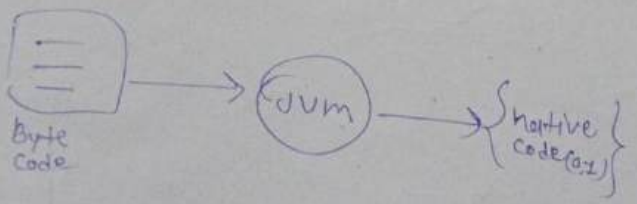


JRE = Java runtime Environment

JVM = Java virtual machine

Compiler → JDK के अंदर होता है

(ii) Execution:



Ex System.out.println("Hello World with Java");
System.out.print("Hello world with Java");

Note: println → Next line print

output: Hello World with Java
Hello world with Java

1. Function

```
void main() {
}
```

2. Class

```
Class main {
  void main() {
  }
}
```

Note:

```
public class main {
  public static void main(String[]args) {
    System.out.Print("Hello world with java\n");
    System.out.print("Hello world with java\n");
    System.out.print("Hello world with java");
  }
```

Output:

Hello world with java
Hello world with java
Hello world with java

Note:

```
System.out.print("Hello world with java\n from\n Apni Kaksha");
```

output: Hellowoarld with java
from
Apni Kaksha

Ex:

```
*
* *
* * *
* * * *
```

Sol:

```
public class First
{
public static void main(String[] args)
{
System.out.println("*\n **\n***\n****");
}
}
```

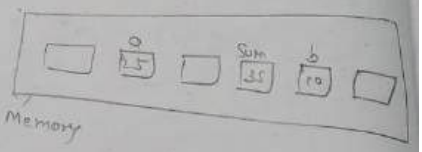
VS code → Sout → Print

* VARIABLES IN JAVA :

a = 25, b = 10

2 * (a + b)

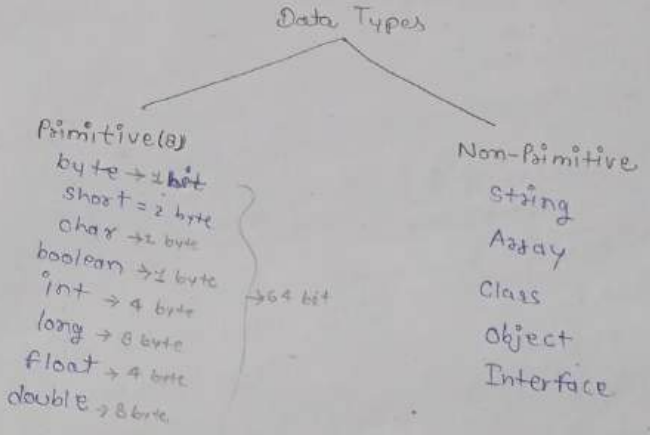
Constant, Variable



Program:

```
Public class main {
  Public static void main(String[] args) {
    // variables
int age = 48;
double price = 25.25;
int a = 25;
int b = 10;
b = 20;
name = "ironman";
}
}
```

* Data Types :



Note :

Java ↓ Typed

(Java एक Typed language है मतलब इस language में आपको हर एक variable declare करने से पहले यह बताना पड़ता है कि उस variable का Type क्या है) Ex : float, int, double etc.

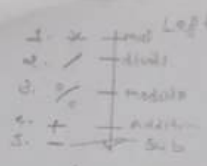
1 byte = 8 bits

Program :

```
Public class main {
    Public static void main(String[] args) {
        // Vambles
        int a = 10;
        int b = 25;
        int sum = a+b;   or int diff = b - a;
        System.out.println(sum);
    }
}
```

output ⇒ 15

output : 35



Quiz

1. to calculate: $\frac{a \times b}{a-b}$ $\frac{10 \times 5}{10-5} = \frac{50}{5} = 10$

a=10, b=5

(A) int ans = a * b / a - b;

Sol: 10 * 5 / 10 - 5

→ 50 / 10 - 5

→ 5 - 5

→ 0 Ans

Program:

```
public class main {
public static void main(String[] args)
{
int a=10;
int b=5;
int ans = a * b / a - b;
System.out.println(ans);
}
}
```

Ans 0

(B) int ans = (a * b) / (a - b);

Sol: (10 * 5) / (10 - 5)

(50) / 5

⇒ 10

Program:

```
public class main {
public static void main (String[] args)
{
int a=10;
int b=5;
int ans = (a * b) / (a - b);
System.out.println(ans);
}
}
```

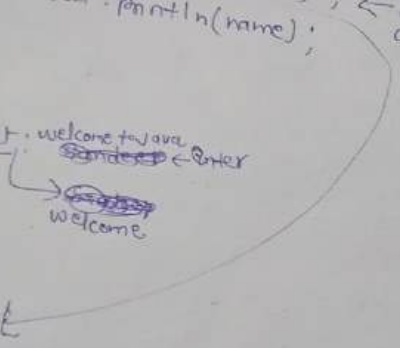
output: 10

Use of Input stdt

```
import java.util.Scanner;
Public class first {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
String name = sc.next();
System.out.println(name);
}
}
```

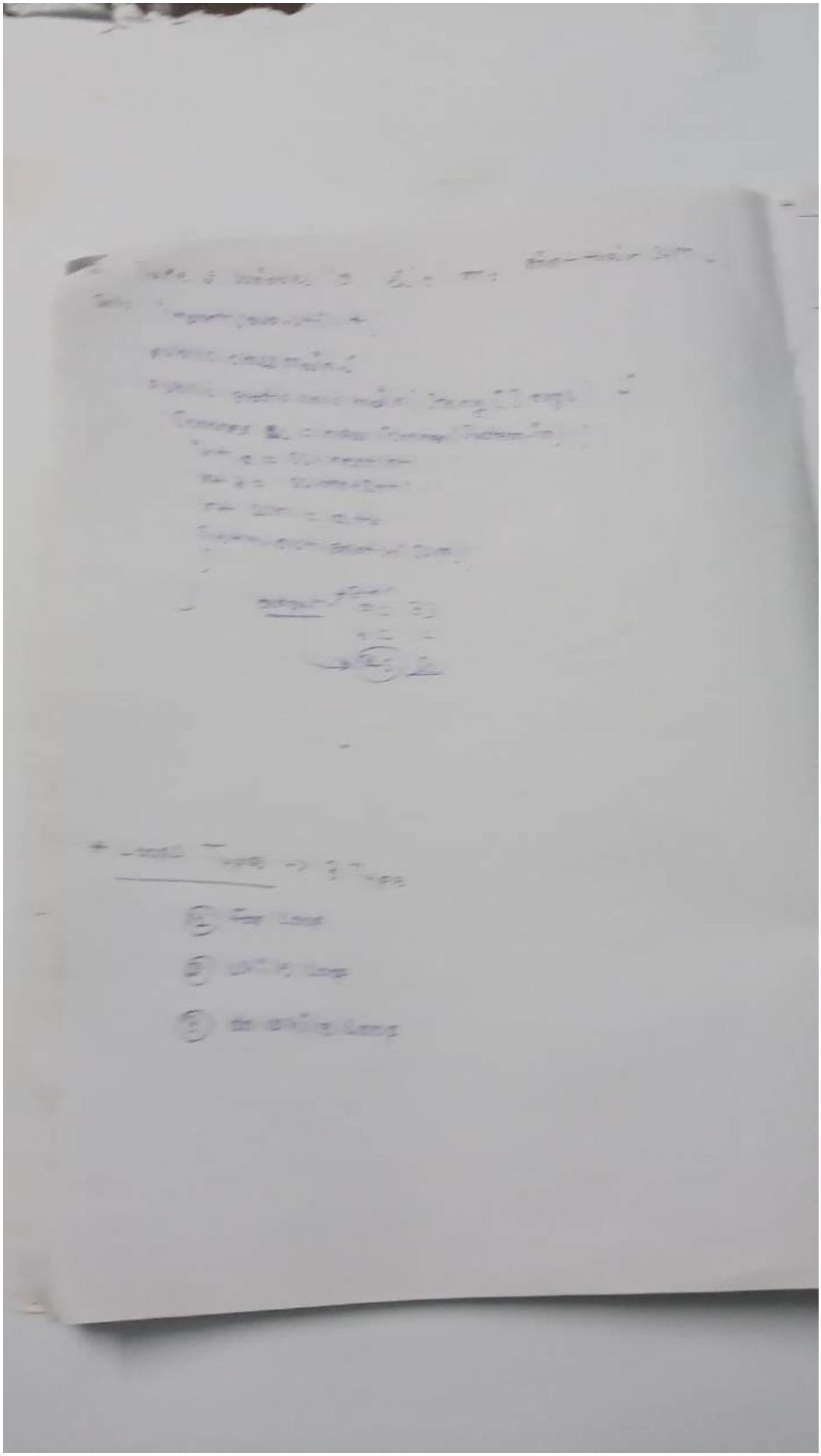
output: welcome to java ← Enter

welcome



// nextInt() → Integer value

// nextFloat() → floating value



* Loops in Java :

for → int i=0 ; i<11 ; i++

Syntax :

```
for (initialisation; condition; updation) {
    // do something
}
```

Program :

```
Public class Loops {
public static void main(String args[]) {
for(int counter = 0; counter <100 ; counter=counter+1)
                                  or counter++
{
System.out.println(" Hello World ");
}
}
}
```

Q. Print two number 0 to 10.

Program:

```
Public Class first {
Public static void main(String[] args) {
for(int counter = 0; counter < 11; counter++ ) {
System.out.println(counter);
}
}
}
```

Output

0 1 2 3 4 5 6 7 8 9 10

Note:
change System.out.print(counter+" ");
output 0 1 2 3 4 5 6 7 8 9 10

Or

```
Public class first {
public static void main(String[] args) {
for(int i=0; i<11; i++) {
System.out.println(i);
}
}
}
```

Loops in Java while (जबतक)

```
while (condition) {
    // do something
}
```

Program:

```
public class first {
public static void main( String [] args )
{
int i=0 ;
while (i<11) {
System.out.println(i++);
}
}
}
```

output

0
1
2
3
4
5
6
7
8
9
10

Loops in Java do (करना)

```
do {
// do something
} while (condition);
```

Program:

```
public class first {
Public static void main(String[]args){
int i=0;
do {
System.out.println(i);
i++;
} while (i<11);
}
}
```

output

0
1
2
3
4
5
6
7
8
9
10

Q. Print the Sum of First n Natural Numbers.

given by n = 4 (number)

Sol= 1+2+3+4 = 10

```
import java.util.Scanner;
public class first {
public static void main(String[] args) {
Scanner sc = new scanner(System.in);
int n = sc.nextInt();
int sum = 0;
for(int i=0; i<=n; i++) {
sum = sum+i;
}
System.out.println(sum);
}
}
```

Run: javac first.java
java first
Or
javac first.java && java first

Output Enter ⇒ 4
→ 10 &

Qus: Print the table of a number input by the user

given n=2

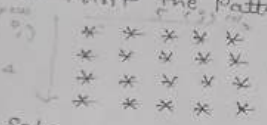
Sol=

```
import java.util.Scanner;
Public class first {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
for(int i=1; i<11; i++) {
System.out.println(i*n);
}
}
}
```

Note


```
for(int i=1; i<=9; i++)S
  sout("* ");
  }
```

Q. Print the Pattern (Solid Rectangle)



```
* * * * *
* * * * *
* * * * *
* * * * *
```

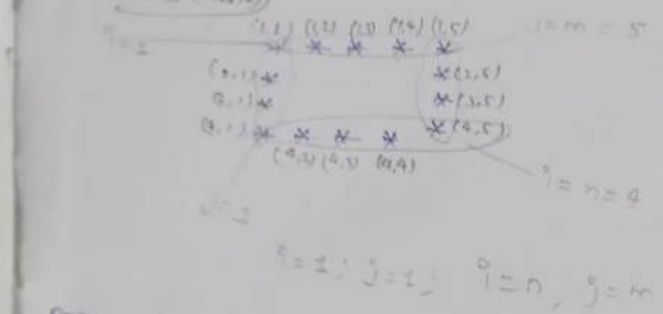
Sol=

```
public class first {
  public static void main(String[] args) {
    int n = 4;
    int m = 5;
    for(int i=1; i<=n; i++) {
      for(int j=1; j<=m; j++) {
        System.out.print("* ");
      }
      System.out.println();
    }
  }
}
```

Print the pattern (Hollow Rectangle)

```
* * * * *
*       *
*       *
* * * * *
```

Sol= [4 X 5 (matrix)]



Program:

```
public class First {
Public static void main (String [] args ) {
  int n = 4
  int m = 5
for(int i=1; i<=n; i++ ) {
  for( int j=1 ; j <=m ; j++ ) {
if ( i==1 || j==1 || i==n || j==m ) {
  System.out.print(" * ");
  } else {
  System.out.print (" ");
  }
  }
  System.out.println( ) ;
}
}
```

3. Print the Pattern (Half Pyramid)

```
*
* *
* * *
* * * *
```

Sol:

```
Public class first {
Public static void main(String[] args) {
for(int i=1; i<=4; i++) {
for(int j=1; j<=i; j++) {
System.out.print("* ");
}
System.out.println();
}
}
}
```

Qus 4. change

```
for(int i=4; i>=1; i--) {
```

Output

```
* * * *
* * *
* *
*
```

```
   *
  * *
 * * *
* * * *
```

program:

```
public class first {
public static void main(String [] args) {
for(int i=1; i<=4; i++) {
for( int j= 1; j<=4-i; j++) {
System.out.print(" ");
}
for(int j=1; j<=i; j++) {
System.out.print("* ");
}
System.out.println( );
}
}
```

Qus 6.

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

program:

```
public class first {
public static void main(String[] args) {
for( int i=1; i<=5; i++) {
for(int j=1; j<=i; j++ ) {
System.out.print(j+" ");
}
System.out.println( );
}
}
```

7.

```
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

Sol=

Program:

```
public class First{
public static void main(String[]args){
for(int i=1; i<=5; i++){
for(int j=1; j<=5-i+1; j++){
System.out.print(j);
}
System.out.println();
}
}
```

8.

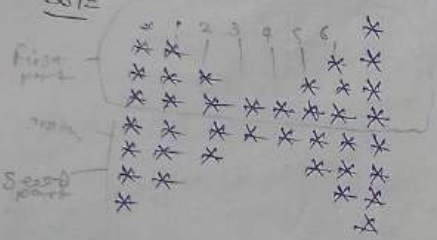
```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

Program:

```
public class first{
public static void main(String[]args){
int number=1;
for(int i=1; i<=5; i++){
for(int j=1; j<=i; j++){
System.out.print(number+" ");
number++;
}
System.out.println();
}
}
```

10. Butterfly Pattern

Sol:-



n=4

	first part		second part
i=1	1 star	6 space	1 star
i=2	2 star	4 space	2 star
i=3	3 star	2 space	3 star
i=4	4 star	0 space	4 star

i=1 → 2*3 → n-i (4-1)
i=2 → 2*2 → n-i (4-2)
i=3 → 2*1 → n-i (4-3)
i=4 → 2*0 → n-i (4-4)

Spaces = 2*(n-i)

Program:

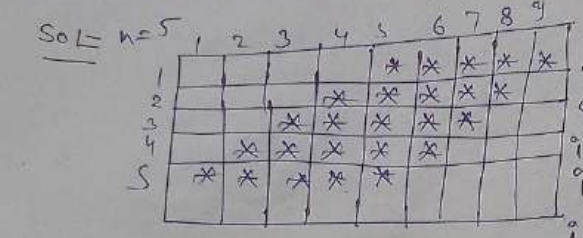
```
Public class first {
Public static void main(String[] args) {
int n=4;
for(int i=1; i<=n; i++) {
for(int j=1; j<=i; j++) {
System.out.print("*");
}
int spaces = 2*(n-i);
for(int j=1; j<=space; j++) {
System.out.print(" ");
}
```

```
for(int j=1; j<=i; j++) {
    System.out.print(" *");
}
System.out.println();
}
for(int i=n; i>=1; i--){
 for(int j=1; j<=i; j++) {
  System.out.print(" *");
 }
int space =2 *(n-i);
  for(int j=1; j<=space; j++){
   System.out.print(" ");
  }
  for(int j=1; j<=i; j++) {
     System.out.print(" *");
     }

     System.out.println();
```

11. Solid Rhombus

```
    * * * * *
  * * * * *
* * * * *
  * * * * *
    * * * * *
```

Sol:- n=5



i=1 4 spaces 5 stars
i=2 3 spaces 5 stars
i=3 2 spaces 5 stars
i=4 1 spaces 5 stars
i=5 0 spaces 5 stars

i=1 4 spaces = n-i
i=2 3 spaces = n-i

spaces = (n-i)

Program:

```
public class First {
public static void main(String [] args) {
for(int i=1; i<=5; i++){
for (int j=1; j<=5-i; j++) {
System.out.print(" ");
}
for(int j=1; j<=5; j++){
System.out.print("* ");
}
System.out.println();
}
}
}
```

12.

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

Sol:-

Program:

```
Public class first {
    Public static void main(String[] args) {
        for(int i=1; i<=5; i++) {
            for(int j=1; j<=5-i; j++) {
                System.out.print(" ");
            }
            for(int j=1; j<=i; j++) {
                System.out.print(i + " ");
            }
            System.out.println();
        }
    }
```

13.

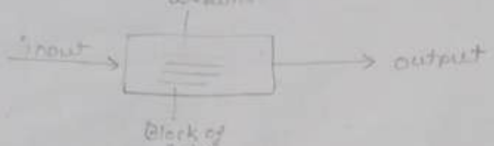
Program:

```
public class first {
    public static void main(String[] args) {
        for(int i=1; i<=4; i++) {
            for(int j=1; j<=4-i; j++) {
                System.out.print(" ");
            }
            for(int j=1; j<=2*i-1; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
// Second half
        for(int i=n; i>=1; i--) {
            for(int j=1; j<=n-i; j++) {
                System.out.print(" ");
            }
            for(int j=1; j<=2*i-1; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
```

1 07

Functions

(function java में एक ऐसा Block of code होता है जो कुछ "input लेता है कुछ operation करता है और कुछ output produce करके देता है)



Example: TV remote

Syntax:

```
returnType functionName(type arg1, type arg2..) {
// operations

}
```

type → int, float, String, boolean — Example: int age, float temp

functionName → Example: printMyName, calculateSum

int, float, string ... }
void → no return
public static → keywords

Note:

printmyname
or
print_my_name → used Python
or
printMyName → Camelcase (Upper, Lower) → used java

Q. Print a given name in a function.

Program:

```
import java.util.Scanner;
public class function {
Public static void printMyName(String name) {
System.out.println(name);
return;
}
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
String name = sc.next();
printMyName(name);
}
}
```

Output: Enter → Hello
→ Hello

Q. Make a function to add 2 numbers and return the sum.

Program:

```
import java.util.Scanner;
public class functions {
public static int calculateSum(int a, int b) {
int sum = a+b;
return sum;
}
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
int a = sc.nextInt();
int b = sc.nextInt();
int sum = calculateSum(a,b);
System.out.println("Sum of 2 number is: "+sum
}
}
```

Output: 9
Enter: 4
5

3. Make a function to multiply 2 numbers and return the product.

Program:

```
import java.util.Scanner;
public class function {
public static int calculateProduct(int a, int b) {
return a * b;
}
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
int a = sc.nextInt();
int b = sc.nextInt();
System.out.println("Calculate product of 2 numbers:
" + calculateProduct(a, b));
}
}
```

$0! = 1$

4. find the factorial of a number:

program:

Note: public static void printfactorial(int n) {

means

कोई void function बनाते हैं तो

return :

लिखो या ना लिखो

पर function के अंदर कुछ तो पाओगे ही

Program:

```
import java.util.Scanner;
Public class function {
public static void printfactorial(int n) {
if(n<0) {
System.out.println("Invalid Number");
return;
}
int factorial = 1;
for (int i= n; i>=1; i--) {
   factorial = factorial * i;
}
System.out.println(factorial);
return;
}
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
printfactorial(n);
   }
}
```

Output:

enter: 5

↳ 120

Difference

Note:
function और method दोनों ही block of code होते हैं जो कोई operation perform करते हैं।
Input को लेते हैं और output को देते हैं

function (function को हम call करते हैं directly)	Methods (methods को हम class के object के through)

Lecture. 9:

Topic : Time & Space Complexity

- Time Complexity: Relation between Input size & Running Time (operations)

Example:

```
Public stattic void main(String args[]) {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    for(int i=0; i<n; i++) {
        System.out.println("hello");
    }
}
```

→ n time

Note: time complexity ∝ input n
(जैसे-जैसे input n बढ़ाते जायेंगे वैसे वैसे operation बढ़ते रहे होंगे
वैसे Time Complexity बढ़ रही होगी)

* 2Time Complexity : (Time complexcity हम तीन तरीके से लिखते हैं)

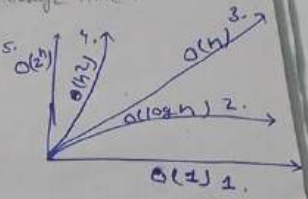
1. BEST CASE (or Big Omega Ω Timecomplexity) → $n^2, n^3, n^4 \ldots$
2. AVERAGE CASE (Θ थीटा)
3. WORST CASE (O(Big ओ)(n))

- BEST CASE (वह minimum time जिसमें अगर Best case लिया तो सारे Code को तो हमारा कोड चल जाए)
- AVERAGE CASE (वह Average time जिसमें वे quality करते अगर को देखेंगे तो हमारे पूरे उतने Average time में चल सका होगा।)

Example :

Number: {1, 2, 3, 4, 5}

search for '1'



Lecture-10 :

* ARRAYS *

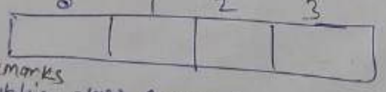
- Arrays : List of items of the same type.

Syntax :

```
type[] arrayName = new type[size];
```

(new → Keywords)

Note: new (new एक Keywords जिसका use memory के अन्दर space लेने के लिये किया जाता है) (non primitive type)



box size = 1 byte
1 byte = 8 bit

Que → integer array of marks

Program :

```
public class Arrays {
    public static void main(String[] args) {
        int[] marks = new int[4];   Or  int marks[] = new int[4];
        marks[0] = 67;
        marks[1] = 77;
        marks[2] = 87;
        marks[3] = 97;
            Or Change → int marks[] = {67, 77, 87, 97};
        System.out.println(marks[0]);
        System.out.println(marks[1]);
        System.out.println(marks[2]);
        System.out.println(marks[3]);
            Or Change → for(int i=0; i<4; i++) {
                            System.out.println(marks[i]);
                        }
    }
}
```

output :

67
77
87
97

Defining an array(2)

type [] arrayName = {1, 2, 3, 4, 5, 6};

for ex

Program:

```
public class Arrays {
public static void main(String[] args) {
int marks [] = {67, 77, 87, 97};
for(int i=0; i<4; i++) {
System.out.println(marks[i]);
}
}
```

→ User ☞ Input ☜ Arrays

Program:

```
public class Arrays {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
int size = sc.nextInt();
int number[] = new int[size];
for(int i=0; i<size; i++) {
System.out.println(number[i]);
}
}
}
```

Output

Enter 6

0 0 0 0 0 0

Int size by Null value

Note: int year = 0

float = 0.0

boolean = false

String = "" Empty String

Store memory location

Program:

```
import java.util.Scanner;
public class Arrays {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
int size = sc.nextInt();
int number[] = new int[size];
//input
for(int i=0; i<size; i++) {
number[i] = sc.nextInt();
//output
for(int i=0; i<size; i++) {
System.out.println(number[i]);
}
}
```

Run Program:

Enter: 5

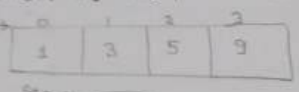
1 2 3 4 5

output:

1 2 3 4 5

Q. Take an array as input from the user. Search for a given number x and print the index at which it occurs.

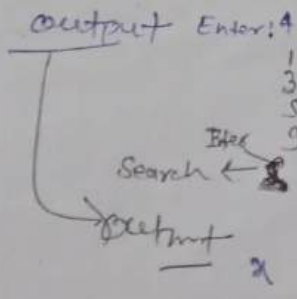
Sol:



Search array is for linear Search array

Program:

```
import java.util.Scanner;
Public class Arrays {
public static void main(String[]args) {
Scanner sc = new Scanner(System.in);
int size = sc.nextInt();
int number[] = new int[size];
// input
for(int i=0; i<size; i++) {
number[i] = sc.nextInt();
}
int x = sc.nextInt();
// output
for(int i=0; i<number.length; i++) {
if(number[i]==x) {
System.out.println("x found at index:"+i);
}
}
}
```



Q. find the maximum & minimum number of an array of integers,

Sol:

```
import java.util.*;
Public class Array {
Public static void main (String args[]) {
Scanner sc = new Scanner (System.in);
int size = sc.nextInt();
int numbers[] = new int[size];
for(int i=0; i<size; i++) {
numbers[i] = sc.nextInt();
}
int max = Integer.MINVALUE;
int min = Integer.MAX_VALUE;
for(int i=0; i<number.length;i++) {
if (numbers[i] < min) {
min = numbers[i];
}
if (numbers[i] > max) {
max = numbers[i];
}
}
```

Q. Take an array of number as input and check if it is an array sorted in assending order.

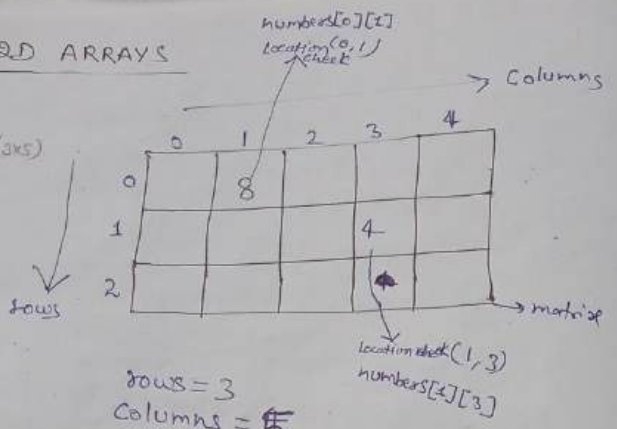
Ans:

```
import java.util.*;
public class Arrays {
public static void main(String args[]) {
Scanner sc = new Scanner(System.in);
int size = sc.nextInt();
int numbers[] = new int[size];
for (int i=0; i<size; i++) {
numbers[i] = sc.nextInt();
}
```

Lecture 11 :

2D ARRAYS

2D Arrays : (3x5)



rows = 3
Columns = 5

Note: एक array का अंदर सब different - different box का collection होता, Rectangle के form में मिलता है जिसके अंदर rows होती हैं जिसके अंदर columns होते हैं इसी structure को हम 2D Arrays कहते हैं।
यह structure memory के अंदर linearly store होता है।)

Declaration :

Syntax: `type[][] arrayName = new type[rows][columns];`

Example:
```
int[][] numbers = new int[3][5];
```

Program:
```
import java.util.*;
public class 2DArray {
public static void main(String args[]) {
Scanner sc = new Scanner(System.in);
int rows = sc.nextInt();
int cols = sc.nextInt();
int[][] number = new int[row][cols];
// input
// rows
for(int i=0; i<rows; i++) {
  // columns
  for(int j=0; j<cols; j++) {
    numbers[i][j] = sc.nextInt();
  }
}
```

```
//output
for(int i=0; i<rows; i++) {
for(int j=0; j<cols; j++) {
System.out.println(numbers[i][j] + " ");
}
System.out.println();
}
}
```

Output

Enter 3
5

```
1 2 3 4 5
5 4 3 2 1
1 2 3 4 5
1 2 3 4 5
5 4 3 2 1
1 2 3 4 5
```

Q. Take a matrix as input from the user.
Search for a given number 'x' and print the indices at which it occurs.

Program: (3x4) matrix

	0	1	2	3
0	1	2	3	4
1	4	3	2	11
2	1	2	3	4

x = 11

```
import java.util.*;
public class 2DArray {
public static void main(String args[]) {
Scanner sc = new Scanner(System.in);
int rows = sc.nextInt();
int cols = sc.nextInt();
int[][] number = new int[rows][cols];
// input
// row
for(int i=0; i<rows; i++){
    // colums
    for(int j=0; j<cols; j++) {
    number[i][j] = sc.nextInt();
    }
}
int x = sc.nextInt();
for(int i=0; i<rows; i++) {
for(int j=0; j<cols; j++) {
// compare with x
if(number[i][j] == x) {
System.out.println("x found at locations(" +i+ ", " +j+ ")");
}}}}
```

Search

Output print

Output

```
3
4
1 2 3 4
4 3 2 11
1 2 3 4
Search x = 11
x found at locations
(1, 3)
```

Topic : Strings

* Strings is a non primitive data type.

Program:

```
import java.util.Scanner;
public class Strings {

    public static void main(String args[]) {
    // String Declaration
        String name = "Tony";
        String fullName = "Tony Stark";
    }
}
```

Program:

```
import java.util.Scanner;
public class String {
    public static void main (String args[]) {
    // User Input
        Scanner sc = new Scanner(System.in);
        String name = sc.nextLine();
        System.out.println("your name is: " + name);
    }
}
```

output: your name is: Enter

Program: import java.util.Scanner;

public class Strings {

public static void main (String args[]) {

//Concatenation

String firstName = "tony";

String lastName = "stark";

String fullName = firstName + lastName;

System.out.println (fullName);

}

}

Output: tonystark

Program: (length) find

import java.util.Scanner;

Public class String {

Public static void main (String args[]) {

String firstName = "tony";

String lastName = "Stark";

String fullName = firstName + "@" + lastName;

output ← // tony@Stark

System.Out.println (fullName.length());

}

}

Output: 1 2 3 4 5 6 7 8 9 10

tony@stark

10 ans

Program: charAtMethod

output:

```
import java.util.Scanner;
public class String {
public static void main (String [] args) {
String firstName = "tony";
String lastName = "Stark";
String fullName = firstName + lastName;
//charAt → used
for(int i=0; i<fullName.length(); i++) {
System.out.println(fullName.charAt(i));
}
}
```

output → tony@stark

Program: Compare (String ko Compare karna)

```
import java.util.*;
public class String {
public static void main(String []args){
// Compare
String name1 = "Tony";
String name2 = "Tony";
//1 s1>s2 : +ve value  Return value
//2 s1==s2 : 0
//3 s1<s2 : -ve value
if(name1.compareTo(name2)==0){
System.out.println("Strings are equal");
}else{
System.out.println("String are not equal");
}
}
```

output

String are equal.

Or if (name1==name2)

substring(beg index, end index)

Program:

```
import java.util.Scanner;
public class String {
public static void main(String args[]) {
String sentence = "My name is Tony";
String name = sentence.substring(11, sentence.length());
System.out.println(name);
}
}
```

Output: Tony

Lecture 13: String Builder

Strings in Java are Immutable.

Program:

```
import java.util.Scanner;
public class String {
public static void main(String args[]) {
StringBuilder sb = new StringBuilder("Tony");
System.out.println(sb);
//char at index 0
System.out.println(sb.charAt(0));
```

```
// set char at index 0
sb.setCharAt(0, 'p');
System.out.println(sb);
}
}
```

output: Tony
Pony

Program: (insert)

```
public class String {
public static void main(String args[]) {
StringBuilder sb = new StringBuilder("Tony");
System.out.println(sb);
sb.insert(0,'s');
System.out.println(sb);
}
}
```

output: Tony
STony

or change

```
sb.insert(2,'n');
```

output: Tony
Tonny

change

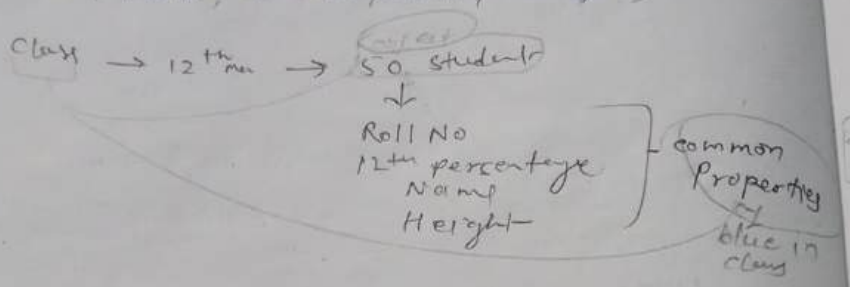
// delete the extra 'n' => sb.delete(start, end);

```
sb.delete(2, 3);
System.out.println(sb);
}
}
```

output: Tony
Tonny
Tony

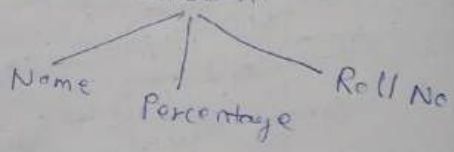
Topic : OOPS in Java

* Java is a oop programming language.
* User defined data type
  Ex: int, float, double, char, string



* Classes - Creation:

New data type → student



(classes-creation d2ir)

program:

```
package copsinJAVA;
public class StudentClass {
    // creating a new data type
    public static class Student {
        String name;
        int rno;
        double percent;
    }
    public static void main (String args[]) {
        Student X = new student();
        x.name = "Raghav";
        x.rno = 83;
        X.percent = 92.5;
        System.out.println(x.percent);
    }
}
```

class — data type creation — object — change

output: 83

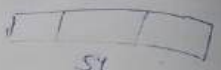
name	rno	percent
Raghav	83	92.5

* Objects — Creation:

ClassName objectname = new ClassName ();

Ex

Student S1 = new Student();



Note:

* objects are real life entities
* Classes are blueprints.
(ex: calobject create karna ho jisme : → multiple attributes)

User-Input ~~Program~~:

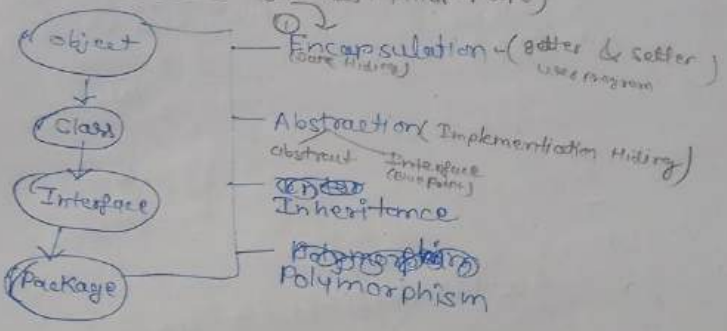
```
import java.util.Scanner;
public class StudentClass{
public static class Student{
    String name;
    int rollno;
    double percent;
}
public static void main (String[] args){
Scanner sc = new Scanner(System.in);
int sc = sc.nextInt();
Student S1 = new Student();
    S1.name = "Sandeep";
    S1.rollno = 3;
    S1.percent = 92.5;
Student S2 = new Student();
S2.name = "Akshit";
S2.percent = 97.2;
S2.rollno = 3;
}}
```

Note:

* OOP(S) → (Independent language)

S = System or Structure

* Java OOPs concepts : (Java has 4 pillar of OOPs)



Object : Real life entity (Physical entity)
(or Any real world entity)

Class : Virtual entity (Logical entity)
or (Class is a factory Producing object
or (A class is the blueprint of object)

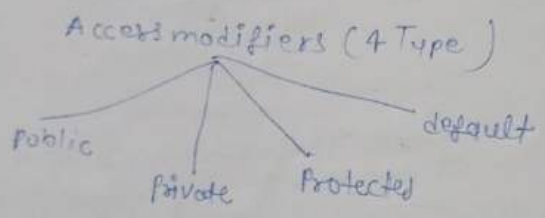
Note : Access Modifier has 4 Type

Modifier	Class	Package	Subclass	Global
Public	Yes	Yes	Yes	Yes
Protected	Yes	Yes	Yes	No
Default	Yes	Yes	No	No
Private	Yes	No	No	No

* Create a class:

- Access modifiers
- Class keyword
- Class name
- Body name

```
public class Car {
    // class Body
}
```



Q. Create a class named "PERSON" with a variable age.

Sol-

```
public class Person {
    int age=20;
public static void main(String[] args){

}
}
```

* Create an object:

Syntax:

```
ClassName ObjectName = new ClassName();
```

Example:

```
Car BMW = new Car();
```

Car() — Constructor

Q. Create an object of person class

Sol:

```
Public class Person {
    int age = 20;
    public static void main(String[] args) {
        Person Rohan = new Person();
        System.out.println(Rohan.age);
    }
}
```

Que: Largest element in the array

Program:

```
import java.util.Scanner;
public class ArrayLargestElement {
    public static void main(String args[]) {
        int arr[] = {2, 5, 1, 3, 0, 9};
        SOP("Largest element array is: " + sort(arr));
    }
    static int sort(int arr[]) {
        Arrays.sort(arr);
        return arr[arr.length - 1];
    }
}
```

Output: Largest elmt array is: 9

Time Complexity: O(log n²)

Topic : Operators in JAVA

OPERATORS : Symbols that tell compiler to perform some operation.

1. Arithmetic Operators A=10, B=5

Binary
+
-
*
/
%

Unary
(++a / a++) ++ (Increement)
(--a / a--) -- (Decrement)

2 Unary operator!

Pre Increement
++a (using step
1. Change value
2. Use value)

```
int a=10;
int b=0;
b=++a;
System.out.println(a);
System.out.println(b);
}
```

output: b=10, a=11

Post Increement
a++ (using step)
1. Used value
2. Change value

```
int a=10;
int b=0;
b=a++;
System.out.println(a);
System.out.println(b);
}
}
```

output: a=11
b=11

PreDecrement

`--a`

using step
1. Change value
2. Use value

```
int a=10;
int b=0;
b= --a;
System.out.println(a);
System.out.println(b);
}
```

output: a=9, b=9

Post Decrement

`a--`

using step
1. Used value
2. Change value

```
int a=10;
int b=0;
b= a--;
System.out.println(a);
System.out.println(b);
}
```

output: b=10, a=9

(3) **Relational Operators :** → result = Boolean

1. == ⇒ equal to
2. != ⇒ not equal to
3. > ⇒ greater than
4. < ⇒ less than
5. >= ⇒ Greater than equal to
6. <= ⇒ less than equal to

Logical operators : Table

OR	Statement 1	Statement 2	AND
T	T	T	T
T	T	F	F
T	F	T	F
F	F	F	F

(4) **Logical Operators :**

1. && ⇒ AND [a=10, b=20, c=30] [a<b (True) && b<c (true) = result True Ans]
2. || ⇒ OR [1st Statement True, 2nd Statement False/True = result True] [a<b || c<a (10<20 True) || 30<10 (False) = result True Ans]
3. ! ⇒ NOT

(5) Bitwise Operators :

Note: True = 1
False = 0

A = 0101
B = 0110

i) & (AND) Binary
0101
0110
0100 → result

ii) | (OR) Binary
0101
0110
0111 → result

iii) ^ (XOR) Binary → Similar value = False result
different value = true
Ex. 00 → 0, 01 → 0, (01 / 10 = 1)

iv) ~ (complement) Binary ones → 1 → 0, 0 → 1

v) << (Binary Left Shift)

vi) >> (Binary Right Shift)

(6) Assignment Operators :

=

+=

-=

*=

/=

* Bit Manipulation ← Topic

left shift

N << i

Ex. 2 << 1

010 << 1

Ans 100

right shift

N >> i

2 >> 1

Ex 010 >> 1

001

i) Get Bit : (position ko check karna hai ya 1 or 0)

Note:

0 1 0 1

3 2 1 0 (position)

Q. Get the 3rd bit (Position=2) of a number. (n=0101) (5)

Sol:- Steps:

① Bit Mask : 1 << i

② Operation : AND

1 << 2

0001 << 2

0100 ←

0100 AND 0101

Bit Mask ⇒ 0100

Number

Program:

```
public class Bits {
    public static void main(String args[]){
        int n=5; //0101
        int pos=2;
        int bitMask=1<<pos;
        if((bitMask&n)==0){
            System.out.println("bit was zero");
        } else {
            System.out.println("bit was one");
        }
    }
}
```

output: bit was one Ans

non-zero ⇒ 1 (Value) (Position) Ans

Zero ⇒ 0 (Value)

Ex 0000

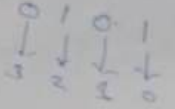
(3)

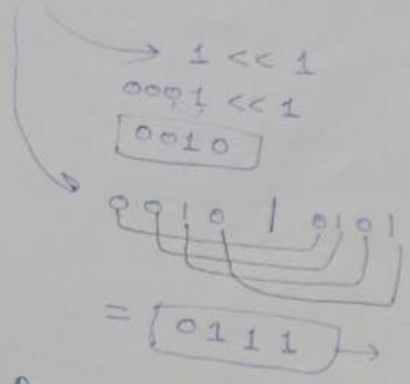
(2) Set Bit :-

Q. Set the 2nd Bit (position = 1) of a number n. (n = 0101)

Sol= Step1: Bit Mask : 1 << i

Step2: Operation : OR





Ans

Program:

```
import java.util.Scanner;
public class Bits {
public static void main (String args[]) {
int n = 5;
int pos = 1
int bitMask = 1 << pos;
int newNumber = bitMask | n;
System.out.println(newNumber);
}
}
```

Output: 7

③ Clear Bit (position ki वाली bit value को clear करना)

Q. Clear the 3rd bit (position = 2) of a number n. (n=0101)

Sol= Step1 : Bit Mask : 1<<i

Step2 : Operation : AND with NOT

1 << 2

0001 << 2

⇒ 0100

~(0100) ⇒ 1011

(1011) & (0101)

⇒ 0001 ⇒ $(1)_{10}$

Program:

```
public class Bits {
public static void main (String args[]) {
int n=5;
int pos=2;
int bitMask = 1<<pos;
int notBitMask = ~(BitMask);
int newNumber = notBitMask &n;
System.out.println(newNumber);
}
}
```

Output: 1

(20) Update Bit: (1 change karna 0 / 0 change karna 1)

Q. Update the 2nd bit (position = 1) of a number n to 1. (n = 1010)

Sol: Step 1. For 1

① Bit Mask: 1<<i

② Operation: AND with NOT

Step 2. For 0 1

① Bit Mask: 1<<i

② Operation: OR

↓

① 1 << 1

0001 << 1

0010

② (0010) | (0101)

↓

0111

Topic : Sorting in Java

1. Bubble Sort
2. Selection Sort
3. Insertion Sort

1. Bubble Sort : (Bubble Sort के अन्दर hum element को उठाते है और जो सबसे भारी element होता है उसको end में push कर देते है उसके बाद उससे जो Next भारी element होता है उसको Second last पर push कर देते है। तो इस तरीके से हम भारी-भारी element को पीछे की तरफ push कर देते है। वह Array को Sort करेंगे asending Order में अब रखते है।

(n-1) -loop चलता है।

Example:

Note: largest element end में save करते है।

① | 7 | 8 | 3 | 1 | 2 |

7, 8 3, 8 1, 8 2, 8

② | 7 | 3 | 1 | 2 | 8 |

3, 7 1, 7 2, 7, 8

③ | 3 | 1 | 2 | 7 | 8 |

1, 3 2, 3, 7, 8

③ | 1 | 2 | 3 | 7 | 8 |

④ 1, 2, 3, 7, 8

| 1 | 2 | 3 | 7 | 8 |

(n-1) asending Order

Order

Sorting (Bubble Program)

```
import java.util.Scanner;
public class Sorting {
    public static void printArray(int arr[]) {
        for(int i=0; i<arr.length; i++) {
            System.out.println(arr[i]+" ");
        }
        System.out.println();
    }
    public static void main(String args[]) {
        int arr[] = {7, 8, 3, 1, 2};
        // time complexity = O(n2)
        // bubble sort
        for(int i=0; i<arr.length-1; i++) {
            for(int j=0; j<arr.length-i-1; j++) {
                if(arr[j] > arr[j+1]) {
                    // swap
                    int temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                }
            }
        printArray(arr);
    }
}
```

Output:

1, 2, 3, 7, 8

2. Selection Sort : (Selection Sort में हम सबसे Small element निकालते हैं और उसे सबसे आगे रख देते हैं।

0	1	2	3	4
7	8	3	1	2

Sort:

1	2	3	7	8

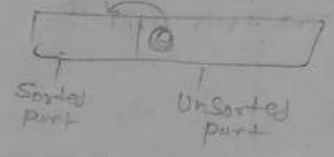
Program:

```
import java.util.Scanner;
class Sorting {
public static void printArray(int arr[]) {
for(int i=0; i<arr.length; i++) {
System.out.println(arr[i]+" ");
}
System.out.println();
}
public static void main(String args[]) {
int arr[] = {7, 8, 3, 1, 2};
//time complexity = O(n^2)
//selection sort
for(int i=0; i<arr.length-1; i++) {
int smallest = i;
for(int j=i+1; j<arr.length; j++) {
if(arr[smallest] > arr[j]) {
smallest = j;
}
}
int temp = arr[smallest];
arr[smallest] = arr[i];
arr[i] = temp;
}
printArray(arr);
}
}
```

Output

1,2,3,7,8

3. Insertion Sort : (Insertion Sorting में हम array को पहले array को 2 part में divide करते हैं।



7 | 8 3 1 2

① 7, 8 | 3, 1, 2

② 3 7 8 | 1, 2

③ 1, 3, 7, 8 | 2

④

1	2	3	7	8

Insertion Sort:

Program:

```
import java.util.Scanner;
class Sorting {
public static void printArray(int arr[]){
for(int i=0; i<arr.length; i++){
System.out.println(arr[i]+" ");
}
System.out.println();
}
public static void main(String args[]){
int arr[]={7,0,3,1,2};
//insertion sort
for(int i=1; i<arr.length; i++){
int current = arr[i];
int j=i-1;
while(j>=0 && current<arr[j]){
arr[j+1]=arr[j];
j--;
}
// placement
arr[j+1]=current;
}
printArray(arr);
}
}
```

output

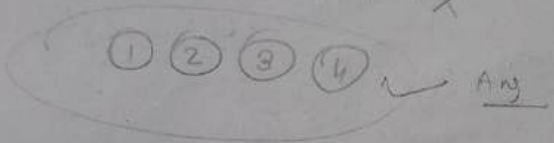
1,2,3,7,8

Topic : Hashset — II nd

Hashset : Set ek aisha structure hota jo duplicate (unique) value ko allow nahi karta hai.

Ex

1, 2, 3, 4, 4 ✗



Note: Hashset ek important data structure hai because as a time complexcity.

Hashset in Java

1. Insert / Add — O(1)
2. Search / contains — O(1)
3. Delete / Remove — O(1)

Syntax:

angular brackets

Hashset < dataType > set = new Hashset <>();

Parenthesis

Iterator :

Package : import java.util.Iterator

Syntax : Iterator it = set.iterator();

Working of Iterator:

(1, 2, 3)

return => 1, 3, 2 Or 2, 3, 1

it = Null → (1) → (2) → (3)
↓
Next

it.next() → 1 (result)
it.next() → 2
it.next() → 3

Note: No Order of the Set.

it.hasNext() → true / false (result)

// next value present true
// not present = false

HashSet Program:

```
import java.util.Scanner;
import java.util.HashSet;
import java.util.Iterator;
public class Hashing {
public static void main(String args[]) {
// Creating Hashset
HashSet<Integer> set = new HashSet<>();
set.add(1);
set.add(2);
set.add(3);
set.add(1);
set.add(5);
if(set.contains(1)) { sop("set contains 1"); }
if(!set.contains(6)) { sop("set does not contains 6"); }
```

```
// Delete
set.remove(1);
if(!set.contains(1)){
    SOP("does not contains 1 we deleted 1");
}
// size
SOP("size of set is : "+set.size());
// print all elements
SOP(set);

// Iterator

Iterator it = set.iterator();
while(it.hasNext()){
SOP(it.next());
}
}
}
```

Output:

set contains 1
does not containing
does not contains 1 - we deleted 1
set of set is : 3
[2, 3, 5]
2
3
5

Topic : HashMAP — I

HashMAP : Two (2) value ka pair ~~at~~ Bhi store karna ho java ke ordues (BIFOR) waha hum HashMap ka used karte hai.

ex : key ⟷ value

Student rollno

RollNo	name
64	"Shiv"
65	"Rajat"
66	"Aman"
71	"Sunny"

Fuel

key, fuel	Value price
CNG	70
diesel	80
Petrol	90

HashMap : Package Name

import java.util.HashMap ;

HashMap creation Syntax ;

HashMap < String , Integer > map = new HashMap <> ();
(key) (Value)

Note : HashMap is a UnOrder Map hoti hai.

Insert in HashMap :

map.put() → Are two case

- Key exist → Ans → Update value print
- Key doesn't exist → new pair is inserted print

Search Operation / lookup Operation (ढूंढना)

Search in HashMap

- get
  - Key exist → Ans → Value print
  - Key doesn't exist → Ans → NULL print
- containsKey
  - Key exist → true
  - Key doesn't exist → false

Program HashMap :

```
import java.util.*;        // All package include
public class HashMaping {
public static void main (String args [] ) {
// country (key) , population (value)
HashMap < String , Integer > map = new HashMap <>();
// Insertion (add)
map.put ("India", 120);
map.put ("china", 130);
map.put ("us", 30);
System.out.println (map);

// Search
if (map.containkey ("India") ) {
  System.out.println ("key is present in the map");
} else {
SOP ("key is not present in the map");
  }
// search key
 SOP (map.get ("china") ); // key exists
 SOP (map.get ("Indonesia") ); // key doesnot exist
```

Notes HashMap loop Syntax

```
Map.Entry<Integer, Integer> e : Map.entrySet()
```

```
// HashMap loop
for(Map.Entry<String, Integer> e : map.entrySet()){
    System.out.println(e.getKey());
    System.out.println(e.getValue());
}

// Second method (result pair print value)
Set<String> keys = map.keySet();
for (String key : keys) {
    System.out.println(key + " " + map.get(key));
}
// Remove
map.remove("China");
System.out.println(map);
}
```

output :

{China=130, US=30, India=120}

key is present in the map

130

null

China
130

US
30

India
120

China 130
US 30
India 120

{China=130, US=30, India=120}

Implementation HashMap function

O(1) ← used →

1. put() = HashMap के अंदर data कुछ add करने के लिए
2. get() = data retrive करने के लिए
3. containsKey() = HashMap के अंदर कोई key exist करती है या नहीं करती
4. remove() = key value pair को remove करने के लिए
5. size() = Total कितने pairs है उसका size
6. keyset() = सारे के सारे keyHashMap में के देता है

Note:

Hashing Means ⇒ Data के form को change करना

Ex: "abc" → 2586
or 1234 → 354 or "lu5cd"

Used → password

BinaryToDecimal

```
public static void binToDec(int binNum) {
    int pow = 0;
    int decNum = 0;
    while (binNum > 0) {
    int lastDigit = binNum % 10;
    decNum = decNum + lastDigit * (int) Math.pow(2, pow);
    pow++;
    binNum = binNum / 10;
    }
    SOP("decimal of " + binNum + " = " + decNum);
    }
    PSVM (String args[]) {
        binToDec(101);
    }
}
```

Output : decimal of 0 = 5

Date: 06-06-2024 "Sorting"

Topic: 10 Basic Sorting Algorithms in

Sorting: Arranging in order.

Bubble Sort:

{5, 4, 1, 3, 2} unsorted array

{1, 2, 3, 4, 5} increasing Order

{5, 4, 3, 2, 1} decreasing Order

Bubble Sort: n = 5

Process:

5 4 1 3 2
4 5 1 3 2
4 1 5 3 2
4 1 3 5 2
4 1 3 2 5

0th Step

4 1 3 2 5
1 4 3 2 5
1 3 4 2 5
1 3 2 4 5

1st Step

1 3 2 4 5
1 3 2 4 5
1 2 3 4 5

2nd Step

3rd step

1 2 3 4 5

1	2	3	4	5

final step

filename save ⇒ BubbleSort.java

Program:

```
import java.util.Scanner;
public class BasicSorting {
public static void bubbleSort(int arr[]) {
for(int turn=0; turn<arr.length-1; turn++) {
for(int j=0; j<arr.length-1-turn; j++) {
if(arr[j] > arr[j+1]) {
// swap
int temp = arr[j];
arr[j] = arr[j+1];
arr[j+1] = temp;
}
}
}
public static void PrintArr(int arr[]) {
for(int i=0; i<arr.length; i++) {
System.out.print(arr[i]+" ");
}
System.out.println();
}
public static void main(String args[]) {
int arr[] = {5,4,1,3,2};
bubbleSort(arr);
PrintArr(arr);
}
}
```

step1: javac BubbleSort.java
step2: java BubbleSort
output: 12345

output: 12345

Selection Sort :

Selection Sort में हम Smallest digit value को हम 0th index या सबसे पहले Push करते हैं।

Time Complexcity = O(n²)

Example

5	4	1	3	2
1	5	4	3	2
1	2	5	4	3
1	2	3	5	4
1	2	3	4	5

— final output

Program:

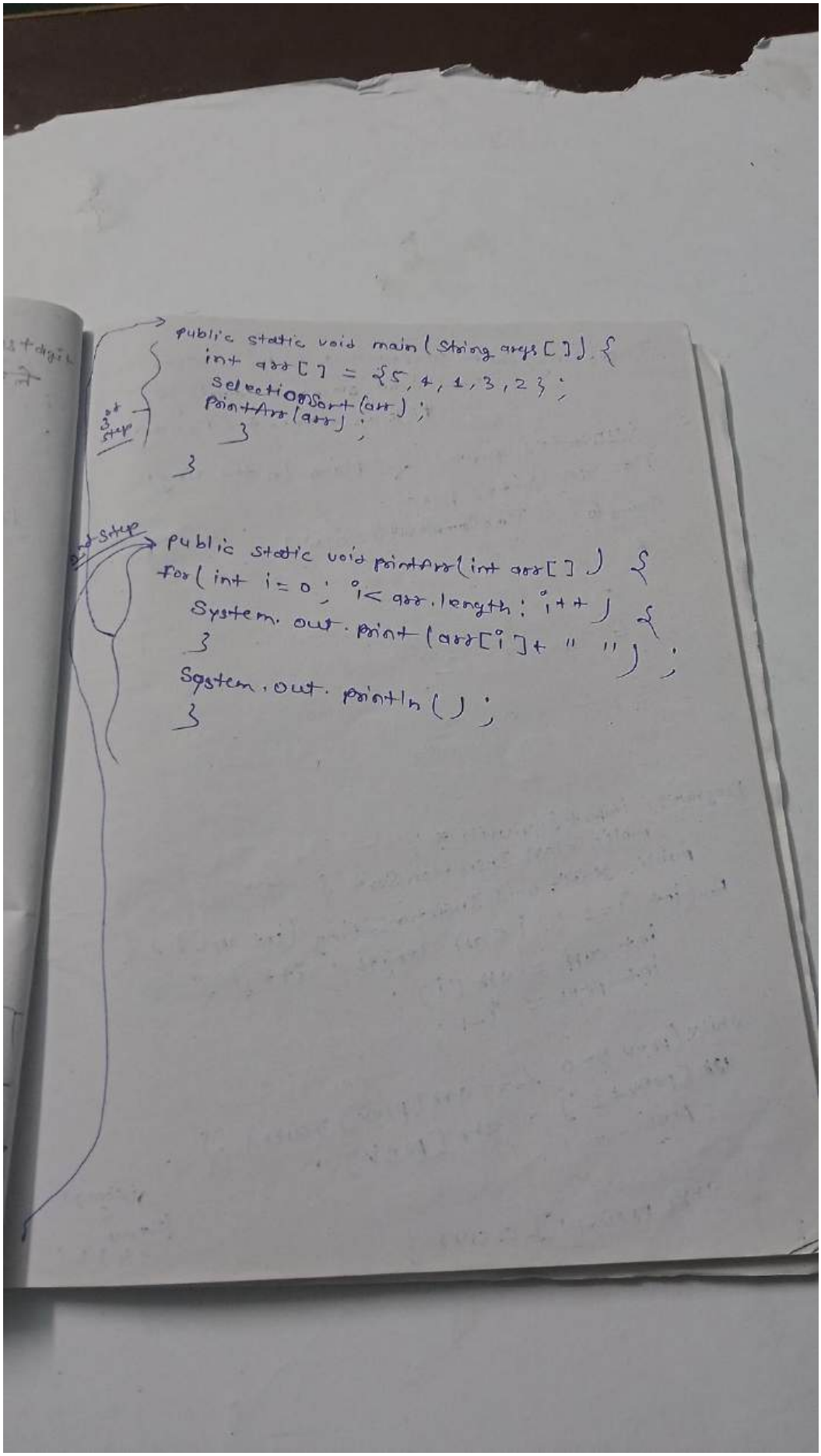
```
import java.util.*;
public class SelectionSort {
public static void selectionSorting (int arr[])
for(int i=0; i<arr.length-1; i++) {
    int minPos = i;                    //current posting
    for(int j = i+1; j<arr.length; j++) {
    if(arr[minPos] > arr[j]) {
        minPos = j;
    }
}
// Swap
int temp = arr[minPos];
arr[minPos] = arr[i];
arr[i] = temp;
}
}
```

change output 5 4 3 2 1

output 1 2 3 4 5

3rd step

3rd step


Public static void main (String args[]) {
int arr[] = {5, 4, 1, 3, 2};
SelectionSort(arr);
PrintArr(arr);
}
3rd step
2nd step
Public static void printArr(int arr[]) {
for(int i=0; i< arr.length; i++) {
System.out.print(arr[i] + " ");
}
System.out.println();
}

Inserting Sort :

(Inserting Sort में हम unsort element को उठाते हैं और उसे compare करके small element को first position में रख देते हैं।)

Pick an element (from unsorted part) & place in the right pos in sorted part.

Example: Time Complexity $= O(n^2)$

5 (4) 1 3 2

4 5 (1) 3 2 — sorted element / unsorted part

1 4 5 3 2

1 3 4 5 2

1 2 3 4 5

Program:

```
import java.util.*;
public class InsertionSort {
public static void InsertionSorting (int arr[]) {
for(int i=1 ; i<arr.length ; i++) {
    int curr = arr[i];
    int prev = i-1;
// finding out the correct pos to insert
while(prev >=0 && arr[prev] >curr) {
arr[prev+1] = arr[prev];
    prev--;
} //insertion
    arr[prev+1] = curr;
}
}
```

(change > to < → output 5 4 3 2 1)

```
public static void printArr(int arr[]){
for(int i=0; i<arr.length; i++){
System.out.print(arr[i]+" ");
}
System.out.println();
}
public static void main(String args[]){
int arr[]={5,4,1,3,2};
insertionSort(arr);
printArr(arr);
}
}
```

Output: 1 2 3 4 5

Inbuilt Sort :

```
import java.util.Arrays;
Arrays.sort(arr)
```

Time complexity = $O(n \log n)$

(11)

```
Arrays.sort(arr, si, ei)
```

si → Starting index

ei → ending index

Note :

$O(n^2)$ → ज्यादा time complexity होती है

$O(n \log n)$ → $O(n^2)$ की तुलना में कम time complexity होती है

Programs

```
import java.util.Arrays;
Public class InbuitSort {
  Public static void printArr(int arr[]) {
  for(int i=0 ; i<arr.length ; i++) {
  System.out.print(arr[i]+" ");
    }
  System.out.println();
  }
  public static void main(String[] args){
  int arr[] = {5, 4, 1, 3, 2};
  Arrays.sort(arr);
  printArr(arr);
    }
  }
```

Output: 1 2 3 4 5

index ← 0 1 2 3 4

change; {5, 4, 1, 3, 2}

```
Arrays.sort(arr, 0, 3);
printArr(arr);
}
}
```

output: 1 4 5 3 2

Counting Sort : (CountingSort में हम कौन-कौन number कितनी बार आयेगा उसको count karaye ; उसके बाद sort kar dege .)

1, 4, 1, 3, 2, 4, 3, 7

1	1	2	3	3	4	4	7

→ Output

Time Complexity = O(n + range)

Date 7/06/2024

15.

APNA COLLEGE

Topic : 2D Arrays

2D Arrays :

MATRIX (ROWS & COLUMNS)

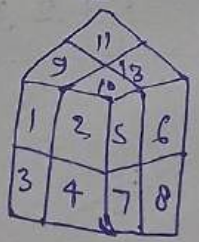
1D

1	2	3

2D

1	2	3
4	5	6

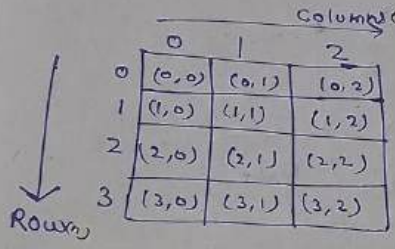
3D



Syntax

int matrix[][] = int int [row size] [column size] ;

Representation



rows = 4
columns = 3

4 X 3 → matrix

cells = rows X columns
cells = 4 X 3
Cells = 12.

Program: Create a simple matrix program.

```
import java.util.Scanner ;
public class Matrices {
public static void main ( String args [] ) {
int matrix [][] = new int [3] [3] ;
int n = matrix.length , m = matrix[0].length ;
                  (Row)              (Columns)
Scanner sc = new Scanner ( System.in ) ;
for (int i=0 ; i<n ; i++ ) {
for ( int j=0 ; j<m ; j++ ) {
matrix [i][j] = sc.nextInt() ;
} }
//output
for (int i=0 ; i<n ; i++ ) {
for ( int j=0 ; j<m ; j++ ) {
System.out.print(matrix [i][j] + " " ) ;
}
System.out.println() ;
} } }
```

Output
Enter: 1 2 3 4 5 6 7 8 9

1 2 3
4 5 6
7 8 9

3.

Qus: Spiral Matrix print

Sol= Approach

	1st	2nd	
Start Row	0	1	++
end Row	3(n-1)	2	--
Start col	0	1	++
end col	3(m-1)	2	--

while (—)

{

① top

② Right

③ bottom

④ left

}

3. Q.

```
import java.util.*;
public class SpiralMatrix {
public static void printSpiral(int matrix[][]){
    int startRow = 0;
    int startCol = 0;
    int endRow = matrix.length-1;
    int endCol = matrix[0].length-1;
while(startRow <= endRow && startCol <= endCol){
  // top
for(int j = startCol; j<=endCol; j++){
    System.out.print(matrix[startRow][j]+" ");
  }
// right

for(int i = startRow+1; i<=endRow; i++){
  System.out.print(matrix[i][endCol]+" ");
  }
// bottom
for(int j = endCol-1; j>=startCol; j--){
  System.out.print(matrix[endRow][j]+" ");
  }
// left
for(int i = endRow-1; i>=startRow+1; i--){
  System.out.print(matrix[i][startCol]+" ");
  }
  startCol++;
  startRow++;
  endCol--;
  endRow--;
} System.out.println();
}
```

Q. what is Transpose

Sol:- Swapping the rows to columns.

Ques 1. Print the number of 7's that are in the 2D array

Example {4, 7, 8} {8, 8, 7}

output = (2)

Program:

```
public class Solution {
    public static void main ( String args[] ) {
int [][] array = { {4,7,8},{8,8,7} };
 int countOf7= 0;
for (int i=0; i<array.length ; i++ ) {
for(int j=0; j<array[0].length ; j++ ) {
if (array[i][i] == 7 ) {
    countOf7++;
    } } }
System.out.println ( +countOf7 ) ;
    }
}
```

Output 2

Qus 2: Print out the sum of the number in the second row of the "nums" array.

Ex: { {1,4,9}, {11,4,3}, {2,2,3} }

output = 18

Program:

```
public class qus {
    public static void main (String[] args) {
int [][] nums = { {1,4,9}, {11,4,3}, {2,2,3} };
int sum=0;
// sum of 2nd row elements
for (int j=0; j<nums[0].length; j++) {
  sum += nums[i][j];
  }
System.out.println (+sum);
    }

}
```

Output:

18

Q3. Transpose matrix

Matrix				Transposed matrix	
a11	a12	a13	⇒	a11	a21
a21	a22	a23		a12	a22
				a13	a23

Program:

```
public class solution {
public static void main (String args[]) {
int row=2, colum = 3;
int[][] matrix = {{2,3,7}, {5,6,7}};
printMatrix(matrix);
int[][] transpose = new int[colum][row];
for (int i=0; i<row; i++) {
for (int j=0; j<column; j++) {
transpose[j][i] = matrix[i][j];
} }
printMatrix (transpose);
}
public static void printMatrix (int[][]matrix) {
System.out.println ("The matrix is:");
for(int i=0; i<matrix.length; i++) {
for(int j=0; j<matrix[0].length; j++) {
System.out.print (matrix[i][j] + " ");
}
System.out.println();
} }
```

output

```
2 3 7
5 6 7
```

Date 8/6/24 Topic: 16. Strings

Q. what are Strings

Ans: String are Immutable.

program:

```
public class Strings {
    public static void main(String args[]) {
        char arr[] = {'a', 'b', 'c', 'd'};
        String str = "abcd";
        String str2 = new String("xyz");
        //Strings declaration
        Scanner sc = new Scanner(System.in);
        String name;
        name = sc.nextLine();
        System.out.println(name);
    }
}
```

1 method OR 2 method

output: Tony ← Enter

Tony

Q. write a program Palindrome or not..

Program:

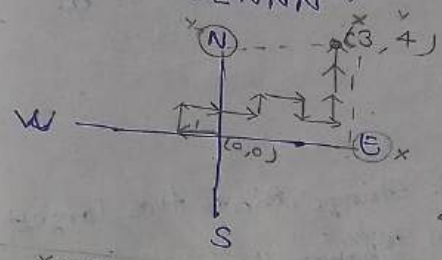
```
import java.util.Scanner;
public class Palindrome {
public static boolean isPalindrome (String str) {
for(int i=0; i< str.length()/2; i++) {
int n = str.length();
if (str.charAt(i) != str.charAt(n-i-1)) {
return false;
}
}
return true;
}
public static void main (String args [ ]) {
String str = "racecar";
System.out.println (isPalindrome);
}
}
```

Output: true

O(n)

Q.: given a route containing 4 directions (E, W, N, S) find the shortest path to reach destination.
"WNEENESENNN"

Sol:



$Path = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$

$= \sqrt{(4-0)^2+(3-0)^2}$

Path => 5 Ans.

O(n) → Time Complexity

X=0	N ↑ →	Y+1	W ← X-1
Y=0	S ↓ →	Y-1	E ⇒ X+1

Program:

```
import java.util.*;
Public class direction {
public static float getShortestPath(String path) {
int X=0, Y=0;
for (int i=0; i<Path.length(); i++) {
char dir = path.charAt(i);
//South
if(dir=='S') {
Y--;
}
//North
else if (dir=='N') {
Y++;
}
//west
else if (dir=='W') {
X--;
}
//East
else {
X++;
}
}
int X2 = X * X;
int Y2 = Y * Y;
return (float)Math.Sqrt(X2+Y2);
}
Public static void main(String args[]) {
String path = "WNEENESENNN";
SOP(getShortestPath(Path));
}
```

output = 5

Note:

Compare to IgnoreCase → 'A' or a
दोनों same मान्य

Q.3

For a given set of Strings, Print the largest String.
"apple", "mango", "banana"

Output = mango

Program:

```
import java.util.*;
public class LargestString {
public static void main (String args[]) {
String fruits[] = {"apple", "mango", "banana"};
String largest = fruits[0];
for(int i=1; i<fruits.length; i++) {
if(largest.compareTo(fruits[i]) <0) {
largest = fruits[i];
}
}
System.out.println(largest);
}
}
```

StringBuilder:

define declaration: StringBuiler sb = newStringBuiler("Hellow");

Program

```
import java.util.*;
public class StringBuilder {
public static void main(String args[]) {
StringBuilder sb = new StringBuilder("");
for(char ch='a'; ch<='z'; ch++) {
    sb.append(ch);
```

append → रखे के पीछे रखे चुड़ते चले जाना

```
}
System.out.println(sb);
Sop(sb.length());
}
}
```

← lengthfind

output:

abcb - - - z

Q. For a given String convert each the first letter of each word to uppercase.
"hi, i, am, sandeep"

Output: "Hi, I Am Sandeep.

Program:

```
import java.util.*;
public class String {
public static String toUpperCase(String str) {
StringBuilder sb = new StringBuilder("");
char ch = Character.toUpperCase(str.charAt(0));
sb.append(ch);
for(int i=1; i<str.length(); i++) {
if(str.charAt(i) == ' ' && i<str.length()-1) {
sb.append(str.charAt(i));
i++;
sb.append(Character.toUpperCase(str.charAt(i)));
} else {
sb.append(str.charAt(i));
}
}
return sb.toString();
}
public static void main(String args[]) {
String str = "hi, i am Sandeep";
System.out.println(toUpperCase(str));
}}
```

Output: Hi I Am Sandeep

Q String Compression

input → "aaabbcccdd"

output: a3b2c3d2

program:

```
import java.util.*;
public class StringCompression {
public static String compress(String str) {
    String newStr = "";
    for (int i=0; i<str.length; i++) {
        Integer count = 1;
while (i< str.length()-1 && str.charAt(i)==
                            str.charAt(i+1)) {
        count ++;
        i++;
    }
    newStr. += str.charAt(i);
    if (count > 1) {
        newStr. += count.toString();
    }
    }
    return newStr;
}
public static void main(String args[]) {
    String str = "aaabbcccdd");
    System.out.println();
    }
}
```

Date 09/6/24

Q. N to 1 print Decresing

```
Public class Recursion {
public static void printDec(int n) {
    if (n == 1) {
        System.out.println(1);
        return;
    }
    System.out.print(n + " ");
    printDec(n-1);
}
public static void main(String args[]){
    int n=10;
    printDec(n);
}}
```

Output - Decsing Order 10 9 8 7 6 5 4 3 2 1

output Incresing ordering print

1 2 3 4 5 6 7 8 9 10

Q. factorial number print

```
public class Recursion {
    public static int printfact(int n) {
        if (n == 0) {
            return 1;
        }
        int fn = n * fact(n-1);
        return fn;
    }
    public static void main(String args[]) {
        int n = 5;
        Sop(printfact(n));
    }
```

Output = 120

Q. Sum of n number:

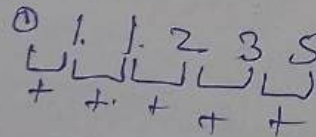
```
public class sum {
    public static int calculateSum(int n) {
        if (n == 1) {
            return 1;
        }
        int sum1 = calculateSum(n-1) + n;
        return sum1;
    }
    public static void main(String args[]) {
        int n = 5
        SOP(calculateSum(n));
    }
}
```

Out = 15

Q. fibonacci number print

```
Public class Recursion {
Public static int calfib (int n ) {
if (n==0 || n==1) {
    return n;
  }
int fn = calfib(n-1) + calfib(n-2);
   return fn;
  }
public static void main(String args[]) {
  int n=25;
System.out.println(calfib(n));
  }
}
```

Output: 75025



Q. check if array is sorted or not

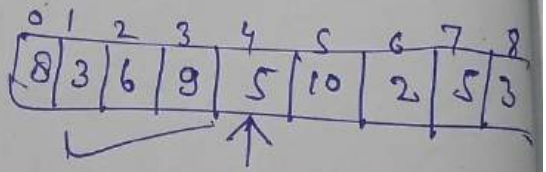
```
public class Recursion {
public static boolean SortedArray(int arr[], int i) {
if(i == arr.length-1) {
return true;
}
if(arr[i] > arr[i+1]) {
return false;
}
return SortedArray(arr, i+1);
}
public static void main(String args[]) {
int arr[] = {1,2,3,4,5};
SOP(SortedArray(arr, 0));
}
}
```

Q. WAP to find the first occurence of an element in an array.

Sol:

```
public class Recursion {
    public static int firstOccurence(int arr[], int key, int i) {
        if (i == arr.length) {
            return -1;
        }
        if (arr[i] == key) {
            return i;
        }
        return firstOccurence(arr, key, i+1);
    }
    public static void main(String args[]) {
        int arr[] = {8, 3, 6, 9, 5, 10, 2, 5, 3};
        System.out.println(firstOccurence(arr, 5, 0));
    }
}
```

output = 4



Q. WAP to find the last Occurence of an element in an array.

Sol:

```
public class Recursion {
public static int lastOccurence(int arr[], int key,
                                int i) {
    if(i == arr.length) {
        return -1;
    }
    int isFound = lastOccurence(arr, key, i+1);
    if(isFound == -1 && arr[i] == key) {
        return i;
    }
    return isFound;
}
public static void main(String args[]) {
    int arr[] = {8, 3, 6, 9, 5, 10, 2, 5, 3};
    SOP(lastOccurence(arr, 5, 0));
}
}
```

Output = 7

Q: print x to the power n Ex $x^n \Rightarrow 2^{10} = 1024$

```
public class Recursion {
public static int power(int x, int n) {
   if(n==0) {
        return 1;
   }
return x * power(x, n-1);
}
public static void main(String args[]) {
   SOP(power(2,10));
   }
}
```

Output:) 1024

(11) Q. Tiling Problem:

Given a "2 X n" board and tiles of size "2 X 1", count the number of ways to tile the given board using the 2 X 1 tiles..
(A tile can either be placed horizontally or vertically.)

Sol:

```
public class Recursion {
public static int TilingProblem(int n) {
    if(n==0 || n==1) {
        return 1;
    }
    int totways = TilingProblem(n-1)+TilingProblem(n-2);
    return totways;
}
public static void main(String args[]) {
    int n=3;
    SOP(TilingProblem(n)
    SOP(totways);
}
}
```

Output = 3

Q12. Remove Duplicates in a string

"appnnacollege"

Program:

```
public class Recursion {
    public static int RemoveDup (String str, int idx,
                                 StringBuilder newStr, boolean map[]) {
        if (idx == str.length()) {
            System.out.println(newStr);
            return;
        }
        char currChar = str.charAt(idx);
        if (map[currChar - 'a'] == true) {
            removeDuplicates(str, idx + 1, newStr, map);
        } else {
            map[currChar - 'a'] = true;
            removeDuplicates(str, idx + 1, newStr.append(currChar),
                             map);
        }
    }
    psvm(String args[]) {
        String str = "appnnacollege";
        removeDuplicates(str, 0, new StringBuilder(""),
                         new boolean[26]);
    }
}
```

output: apncoleg

Topic : ~~Divide & Conquer~~

Date 11-06-24

Topic : Time & SPACE Complexity

Space Complexity

⇒ memory / space → heap → objects
                 → stack → functions &

① input space → (n)
② auxiliary space (n) + add (n2)

Date 12/06/24 Topic : BACKTRACKING

Types of Backtracking

1. Decision
2. Optimization
3. Enumeration

Ex : Backtracking

```
public class Backtracking {
public static void changeArr(int arr[], int i, int val) {
// base case
if (i == arr.length) {
    printArr(arr);
    return;
}
// recursion
arr[i] = val;
changeArr(arr, i+1, val+1);
arr[i] = arr[i] - 2;
}
public static void printArr(int arr[]) {
for(int i=0; i< arr.length; i++) {
System.out.print(arr[i] + " ");
}
System.out.println();
}
```

```
public static void main (String args[]){
  int arr[] = new int[5];
  changeArr(arr, 0, 1);
  printArr(arr);
}
```

output:

1 2 3 4 5 → Base

-1 0 1 2 3 changeArray

Q. findSubset "abc"

output: abc, ab, ac, a, bc, b, c, null

```
public class Subset{
public static void findSubset (String str, String ans, int i)
{
// base case
if(i == str.length()){
  if(ans.length() == 0){
    System.out.println("null");
  }else{
    System.out.println(ans);
  }
  return;
}
//yes choice
findSubset(str, ans+str.charAt(i), i+1);
// No choice
findSubset(str, ans, i+1);
}
public sum(String args[]){
  String str = "abc";
  findSubset(str, "", 0);
}}
```

times $O(2^n * n)$

Q. Find Permutations all of a string "abc"

Output: abc, acb, bac, bca, cab, cba

```
public class Permutations {
public static void FindPermutation (String str,
    String ans ) {
  // base case
 if ( str.length() == 0 ){
      System.out.println (ans);
      return ;
    }
  // recursion
 for(int i=0 ; i<str.length(); i++ ) {
    char curr = str.charAt(i) ;
    String NewStr = str.substring(0, i) + str.substring
                                          (i+1);
    FindPermutation(NewStr, ans+curr) ;
  }
 }
public static void main (String args[]) {
    String str = "abc";
    FindPermutation(str, "");
  }
}
```

Time: O(n*n)

Grid ways: Find number of ways to reach from (0,0) to (n-1, m-1) in a N X M Grid, Allowed moves - right or down.

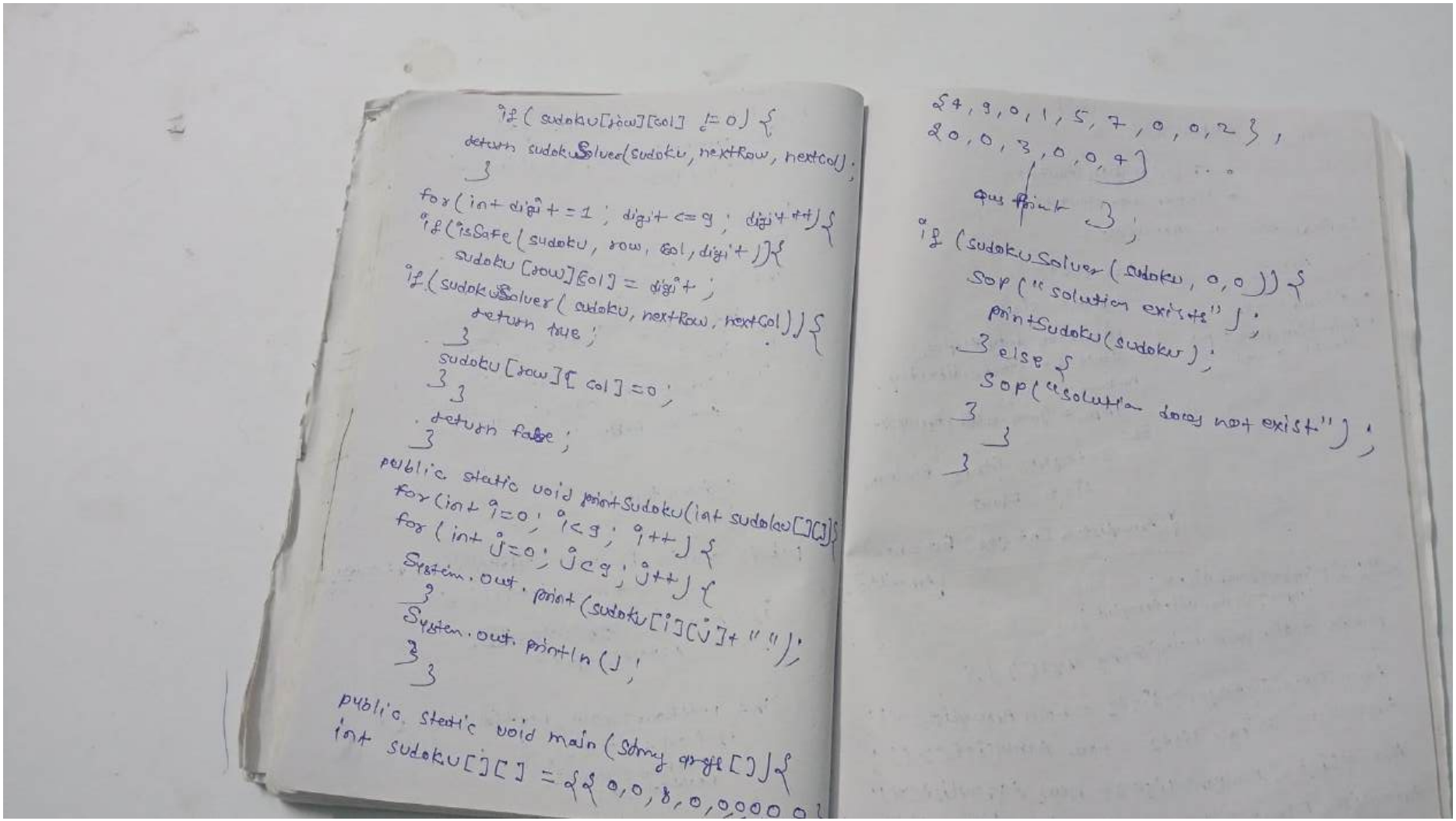
Sol:

```
public class Gridways {
public static int gridways(int i, int j, int n, int m)
{
if (i==n-1 && j==m-1){
    return 1;
}else if (i==n || j==n){
    return 0;
}
int w1 = gridways(i+1, j, n, m);
int w2 = gridways(i, j+1, n, m);
return w1 + w2;
}
public static void main(String args[]){
    int n=3, m=3;
    System.out.println(gridways(0,0,n,m));
}
}
```

Que. Write a function to complete a Sudoku.

Program:

```
public class Classroom {
public static boolean isSafe(int Sudoku[][], int
row, int col, int digit) {
//column
for(int i=0; i<=8; i++) {
if (sudoku[i][col] == digit) {
return false;
} }

// row
for (int j=0; j<=8; j++) {
if (sudoku[row][j] == digit) {
return false;
} }
// grid
int sr = (row/3) * 3;
int sc = (col/3) * 3;
for(int i=sr; i<sr+3; i++) {
for(int j=sc; j<sc+3; j++) {
if (sudoku[i][j] == digit) {
return false;
} }
return true;
}
public static boolean sudokuSolver(int sudoku
[][],
int row, int col) {
if (row == 9) {
return true;
}
int nextRow = row, nextCol = col+1;
if (col+1 == 9) {
nextRow = row+1;
nextCol = 0;
}
```



```
if (sudoku[row][col] != 0) {
    return sudokuSolver(sudoku, nextRow, nextCol);
}
for (int digit = 1; digit <= 9; digit++) {
    if (isSafe(sudoku, row, col, digit)) {
        sudoku[row][col] = digit;
        if (sudokuSolver(sudoku, nextRow, nextCol)) {
            return true;
        }
        sudoku[row][col] = 0;
    }
}
return false;
}

public static void printSudoku(int sudoku[][]) {
    for (int i=0; i<9; i++) {
        for (int j=0; j<9; j++) {
            System.out.print(sudoku[i][j] + " ");
        }
        System.out.println();
    }
}

public static void main(String args[]) {
    int sudoku[][] = {{0,0,8,0,0,0,0,
```

```
{4,9,0,1,5,7,0,0,2},
{0,0,3,0,0,4}...
    Sys Print
}
if (sudokuSolver(sudoku, 0, 0)) {
    Sop("solution exists");
    printSudoku(sudoku);
} else {
    Sop("solution does not exist");
}
}
```

Topic : ARRAYLIST

- inbuilt data structure
- linear data structure

Introduction to Arraylists :

Array	Arraylist
• Fixed size	dynamic size
• Primitive data types can be stored	Primitive data type e can't be stored directly.

Package :
import java.util.ArrayList

Ex :
Integer, String, Boolean, List, Float

```
ArrayList <Integer> list = new ArrayList
```

define Program :

```
import java.util.*;  <- All properties include
import java.util.ArrayList;
public static void main(String args[]) {
ArrayList<Integer> list1 = new ArrayList<>();
ArrayList<String> list2 = new ArrayList<>();
ArrayList<Boolean> list3 = new ArrayList<>();
ArrayList<Float> list4 = new ArrayList<>();
```

Operations

	Timecomplexity	Example	output
Add Element	O(1)	list.add(1); list.add(2);	→ [1,2]
Get Element	O(1)	int element = list.get(2); SOP(element);	Ex [1,2,3,4] ③ output
Remove Element	O(n)	list.remove(2); SOP(list);	[1,2,3,4,5] output : [1,2,4,5]
Set Element at Index	O(n)	list.set(2, 10);	Ex [1,2,3,4,5] out : [1,2,10,4,5]
Contains Element	O(n)	Ex [1,2,3,4,5]	

```
// Add Element  O(1)
list.add(1);
list.add(2);
list.add(3);
list.add(4);
list.add(5);
System.out.println(list);   output [1,2,3,4,5]

// Get Operation - O(1)
int element = list.get(2);
SOP(element);   output 3

// Delete / Remove
list.remove(2);
SOP(list);
output: [1,2,3,4,5]
 -> [1,2,4,5]

// set
list.set(2, 10);
S.O.P(list);
output: [1,2,3,4,5]
 -> [1,2,10,4,5]

// contains
SOP(list.contains(1));
SOP(list.contains(11));
output [1,2,3,4,5]
 -> true
 -> false
}
}
```

Size [1, 2, 3, 4, 5]

```
System.out.println(list.size());
```

output → 5 4

Arraylist Loop Used

```
for(int i=0; i<list.size(); i++){
System.out.print(list.get(i)+" ");
}
System.out.println();
}
```

Ques Print Reverse of an Arraylist

list = 1, 2, 3, 4, 5

output = 5, 4, 3, 2, 1

O(n)

Time

```
import java.util.*;
public class Arraylist{
psvm(String args[]){
ArrayList<Integer> list = new Arraylist<>();
list.add(1);
list.add(2);
list.add(3);
list.add(4);
list.add(5);
```

// Reverse Print

```
for(int i=list.size()-1; i>=0; i--){
System.out.print(list.get(i)+" ");
}
System.out.println();
}
```

output: 5 4 3 2 1

Q. Find Maximum in An Arraylist

input: 2, 5, 9, 3, 6

```
max = Integer.MIN_VALUE
import java.util.*;
public class Arraylist{
psvm(String args[]){
ArrayList<Integer> list = new Arraylist<>();
list.add(2);
list.add(5);
list.add(9);
list.add(3);
list.add(6);
int max = Integer.MIN_VALUE;
for(int i=0; i<list.size(); i++){
if(max < list.get(i)){
max = list.get(i);
}
sop("max element is "+max);
}}
```

output → max element: 9

or max = Math.max(max, list.get(i));

Swap 2 numbers

list : 2, 5, 9, 3, 6 output : 2, 3, 9, 5, 6

idx : idx1 = 1
idx2 = 3

```
public class Classroom{
public static void swap(ArrayList<Integer> list,
                        int idx1, int idx2){
    int temp = list.get(idx1);
    list.set(idx1, list.get(idx2));
    list.set(idx2, temp);
}
public static void main(String args[]){
    ArrayList<Integer> list = new ArrayList<>();
    list.add(2);
    list.add(5);
    list.add(9);
    list.add(3);
    list.add(6);
    int idx1 = 1, idx2 = 3;
    sop(list);
    swap(list, idx1, idx2);
    sop(list);
}
}
```

[2, 5, 9, 3, 6]

output: [2, 3, 9, 5, 6]

Sorting an ArrayList

Collections.sort(list) → ascending order output print.

Package: import java.util.Collections

descending Order print ArrayList → based

Collections.sort(list, Collections.reverseOrder());

Same code

```
sop(list);
Collections.sort(list);
sop(list);
Collections.sort(list, Collections.reverseOrder());
sop(list)
```

output [2, 3, 5, 6, 9]

→ output [9, 6, 5, 3, 2]

Multi-dimension Arraylist

define

```
ArrayList<ArrayList<Integer>> mainlist = new ArrayList<>();
```

Program:

```
public class Classroom {
psvm (String args[]) {
ArrayList<ArrayList<Integer>> mainlist=new
                    ArrayList<>();
ArrayList<Integer> list =new ArrayList<>();
list.add(1);
list.add(2);
mainlist.add(list);
ArrayList<Integer> list2 = new ArrayList<>();
list2.add(3);
list2.add(4);
mainlist.add(list2);
for(int i=0; i< mainlist.size(); i++) {
ArrayList<Integer> currlist = mainlist.get(i);
for(int j=0; j< currlist.size(); j++) {
SOPint(currlist.get(j)+" ");
}
SOP();
} }
```

output

```
1 2
3 4
```

Q. Store the value

list 1 = 1 2 3 4 5
list 2 = 2 4 6 8 10
list 3 = 3 6 9 12 15

Program:

```
public class Classroom {
psvm(String args[]) {
ArrayList<ArrayList<Integer>> mainlist=
                    new ArrayList<>();
ArrayList<Integer> list1 = new ArrayList<>();
ArrayList<Integer> list2 =new ArrayList<>();
ArrayList<Integer> list3 = new ArrayList<>();
for(int i=1; i<=5; i++) {
list1.add(i*1);   // 1 2 3 4 5
list2.add(i*2);   // 2 4 6 8 10
list3.add(i*3);   // 3 6 9 12 15
}
mainlist.add(list1);
mainlist.add(list2);
mainlist.add(list3);
SOP(mainlist);
for(int i=0; i<mainlist.size(); i++) {
Arraylist<Integer> currlist=mainlist.get(i);
for(int j=0; j<currlist.size(); j++) {
SOPint(currlist.get(j)+" ");
}SOP();
} }
```

output [1,2,3,4,5] [2,4,6,8,10] [3,6,9,12,15]

```
1 2 3 4 5
2 4 6 8 10
3 6 9 12 15
```

Container with most water

for given n lines on x-axis, use 2 lines to form a container such that it holds maximum water.

height = [1, 8, 6, 2, 5, 4, 8, 3, 7]

Program:

```
class classroom {
public static int storeWater(ArrayList<Integer> height) {
    int maxWater = 0;
    for(int i=0; i<height.size(); i++) {
        for (int j=i+1; j<height.size(); j++) {
            int ht = Math.min(height.get(i), height.get(j));
            int width = j-i;
            int currWater = ht * width;
            maxWater = Math.max(maxWater, currWater);
        }
    }
    return maxWater;
}
psvm() {
    ArrayList<Integer> height = new ArrayList<>();
    height.add(1);
    height.add(8);
    height.add(6);
    height.add(2);
    height.add(5);
    height.add(4);
    height.add(8);
    height.add(3);
    height.add(7);
}
```

Time complexity = O(n²)

output = 49

Pair Sum - 1

Q. Find if any pair in a sorted ArrayList has a target sum

list = [1, 2, 3, 4, 5, 6], target = 5

Brute Force → method

program:

```
import java.util.*;
public class Classroom {
    public static boolean pairSum(ArrayList<Integer> list, int target) {
        for(int i=0; i<list.size(); i++) {
            for(int j=i+1; j<list.size(); j++) {
                if(list.get(i) + list.get(j) == target) {
                    return true;
                }
            }
        }
        return false;
    }
    psvm(String args[]) {
        ArrayList<Integer> list = new ArrayList<>();
        list.add(1);
        list.add(2);
        list.add(3);
        list.add(4);
        list.add(5);
        list.add(6);
        int target = 5;
        SOP(pairSum(list, target));
    }
}
```

output = true;

Same qus

method - 2 Solution — 2 Pointer Approach

list [1, 2, 3, 4, 5, 6]

0 … n-1

program:

```
import java.util.*;
public class Classroom {
    public static boolean pairSum(ArrayList<Integer> list, int target) {
        int lp = 0;
        int rp = list.size() - 1;
        while(lp != rp) {
            // case 1
            if(list.get(lp) + list.get(rp) == target) {
                return true;
            }
            // case 2
            if(list.get(lp) + list.get(rp) < target) {
                lp++;
            } else {
                rp--;
            }
        }
        return false;
    }
    psvm() {
        ArrayList<Integer> list = new ArrayList<>();
        list.add(1);
        list.add(2);
        list.add(3);
        list.add(4);
        list.add(5);
        list.add(6);
        int target = 5;
        SOP(pairSum(list, target));
    }
}
```

output: true

Ques 2 Sorted & Rotated pair Sum

list = [11, 15, 6, 8, 9, 10],
target = 16

O(n)

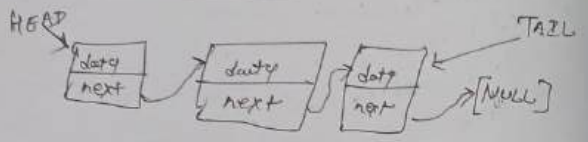
Sol
```
import java.util.*;
public class classroom{
public static boolean pairSum2 (ArrayList<
Integer> list, int target) {
int bp = -1;
int n = list.size();
for(int i=0; i<list.size(); i++) {
if(list.get(i) > list.get(i+1)) {
bp = i;
break;
}}
int lp = bp+1;
int rp = bp;
while (lp != rp ) {
if(list.get(lp) + list.get(rp) == target) {
return true;
}
if (list.get(lp) + list.get(rp) < target) {
lp = (lp+1) % n;
} else {
rp = (n+rp-1) % n;
}} return false;
}
```

```
psum() {
ArrayList<Integer> list = new ArrayList<>();
list.add(11);
list.add(15);
list.add(6);
list.add(8);
list.add(9);
list.add(10);
int target = 16;
Sop (pairSum2(list, target));
}
}
```

output = true

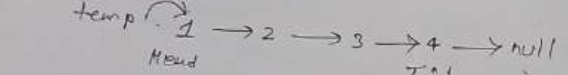
Topic : Linked List

what's a linked list




```
class Node {
  int data;
  Node next;
  public Node (int data) {
    this.data = data;
    this.next = null;
  }
}
```

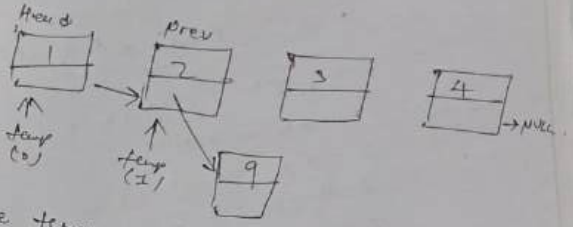
Loop Linked List




```
Node temp = head;
while (temp != null) {
  print(temp.data)
  temp = temp.next;
}
```

Add in the middle

add (index, data)




```
Node temp = head
i = 0
while (i < idx-1) {
  temp → next
  i++
}
```

temp (prev)

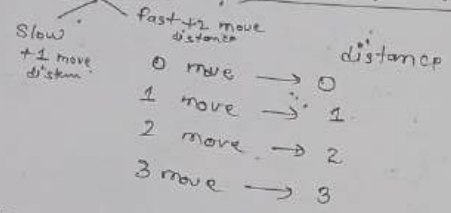
① newNode.next = temp.next

② temp.next = new Node

LinkList-2

Detect a Loop / Cycle in a LL

Or Floyd's Cycle Finding Algorithm



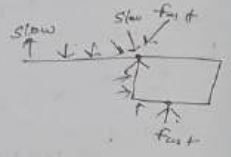
Program:

```
public class classroom {
    public static boolean isCycle() {
        Node slow = head;
        Node fast = head;
        while (fast != null && fast.next != null)
        {
            slow = slow.next;          // +1 move
            fast = fast.next.next;     // +2 move
            if (slow == fast) {
                return true;           // cycle exists
            }
        }
        return false                   // cycle does not exist
    }
```

```
public static void main(String args[]){
    head = new Node(1);
    head.next = new Node(2);
    head.next.next = new Node(3);
    SOP(isCycle());
}
}
```

Output = false

Remov a Loop / Cycle in a LL



(1) detect cycle

(2) slow = head
prev = null

(3)
```
while (slow == fast) {
    slow = slow.next;
    fast = fast.next;
}
```

prev.next = null // last node

Program:

```
public class Classroom {
    public static void removeCycle() {
        Node slow = head;
        Node fast = head;
        boolean cycle = false;
        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;
            if (fast == slow) {
                cycle = true;
                break;
            }
        }
        if (cycle == false) {
            return;
        }
        slow = head;
        Node prev = null;
        while (slow != fast) {
            prev = fast;
            slow = slow.next;
            fast = fast.next;
        }
        prev.next = null;
    }
    psvm() {
        head = new Node(1);
        Node temp = new Node(2);
        head.next = temp;
        head.next.next = new Node(3);
        head.next.next.next = temp;
        SOP(isCycle());
        removeCycle();
        SOP(isCycle());
    }
}
```

Output
true
false

LinkList Java Collections Framework (JCF)

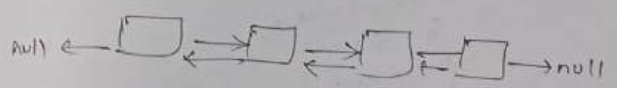
Package : import java.util.LinkedList ;

Use : Integer, Float, character
add
delete

Program : import jav.util.LinkedList ;
public class Classroom {
psvm ( ) {
LinkedList <Integer> ll = new LinkedList<>();
//add
ll.addlast (1) ;
ll.addlast (2) ;
ll.addfirst (0) ;
Sop (ll) ; //0->1->2
//remove
ll.removelast ( ) ;
ll.removefirst ( ) ;
Sop (ll) ;
}
}

output
[0, 1, 2]
[1]

Doubly Linked List



Node {
int data ;
Node next ;
Node prev ;
public node (int data) {
this.data = data ;
this.next = null ;
this.prev = null ;
}
}

Program : Doubly Linked List

```
import java.util.*;
public class Node {
    int data;
    Node next;
    Node prev;
    public Node(int data) {
        this.data = data;
        this.next = null;
        this.prev = null;
    }
}
public static Node head;
public static Node tail;
public static int size;
// add
public void addFirst(int data) {
    Node newNode = new Node(data);
    size++;
    if (head == null) {
        head = tail = newNode;
        return;
    }
    newNode.next = head;
    head.prev = newNode;
    head = newNode;
}
// print
public void print() {
    Node temp = head;
    while (temp != null) {
        System.out.print(temp.data + "<->");
        temp = temp.next;
    }
    System.out.print("null");
}
// remove
psvm() {
    DoubleLL dll = new DoubleLL();
    dll.addFirst(3);
    dll.addFirst(2);
    dll.addFirst(1);
    dll.print();
    SOP(dll.size);
}
```

next
prev
com

Output 1<->2<->3<->null
Size = 3

```
//
public int removeFirst() {
    if(head == null ) {
        SOP ("DLL is empty!!");
        return Integer.MIN_VALUE;
    }
    if (size == 1) {
        int val = head.data;
        head = tail = null;
        size--;
        return val;
    }
    int val = head.data;
    head = head.next;
    head.prev = null;
    size--;
    return val;
}

psvm ()
{
    same last 3 line write
    dll.removeFirst();
    dll.print();
    SOP(dll.size);
}
```

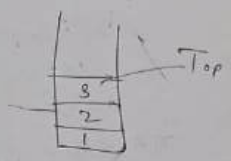
output
1 ⟷ 2 ⟷ 3 ⟷ null
3
2 ⟷ 3 ⟷ null
2

Stack ← Topic

Note: Last In First Out (LIFO)

Operations

1. Push O(1) → add
2. Pop O(1) → delete
3. Peek O(1)
4. isEmpty



Create a stack program → method → used ArrayList

```
import java.util.ArrayList;
public class StackB {
    static class Stack {
        static ArrayList<Integer> list = new ArrayList<>();
        public static boolean isEmpty() {
            return list.size() == 0;
        }
        // push
        public static void push(int data) {
            list.add(data);
        }
        // pop
        public static void push(int data) {
            list.add(data);
        }
    }
```

// pop

```
public static int pop() {
    if(isEmpty()) {
        return -1;
    }
    int top = list.get(list.size()-1);
    list.remove(list.size()-1);
    return top;
}
```

// peek

```
public static int peek() {
    if (isEmpty()) {
        return -1;
    }
    return list.get(list.size()-1);
}
```

```
p.sum() {
Stack s = new stack();
s.push(1);
s.push(2);
s.push(3);
```

```
while(!s.isEmpty()) {
    sop(s.peek());
    s.pop();
}
```

3
3
3

Output: 3
2
1

Implementation StackList+LinkList → Program

```
import java.util.ArrayList;
public class StackB{
    static class Node {
        int data;
        Node next;
        Node(int data) {
            this.data = data;
            this.next = null;
        }
    }
    static class Stack {
        static Node head = null;
        public static boolean isEmpty(){
            return head == null;
        }
        // push
        public static void push(int data){
            Node newNode = new Node(data);
            if(isEmpty()){
                head = newNode;
                return;
            }
            newNode.next = head;
            head = newNode;
        }
        // pop
        public static int pop(){
            if(isEmpty()){
                return -1;
            }
            int top = head.data;
            head = head.next;
            return top;
        }
        // peek
        public static int peek(){
            if(isEmpty()){
                return -1;
            }
            return head.data;
        }
    }
    psvm(){
        Stack s = new Stack();
        s.push(1);
        s.push(2);
        s.push(3);
        while(!s.isEmpty()){
            sop(s.peek());
            s.pop();
        }
    }
}
```

Output
3
2
1

Qus 1 → Recursion O(n)

Push at the Bottom of the stack

```
import java.util.*;
public class StackB {
public static void pushAtBottom(Stack<Integer> s,
                                int data) {
if(s.isEmpty()){
    s.push(data);
    return;
}
int top = s.pop();
pushAtBottom(s, data);
s.push(top);
}
psvm(){
Stack<Integer> s = new Stack<>();
    s.push(1);
    s.push(2);
    s.push(3);
pushAtBottom(s, 4);
while(!s.isEmpty()){
    sop(s.pop());
}}}
```

output

3
2
1
4 ←

Qu 2

Reverse a String using a Stack

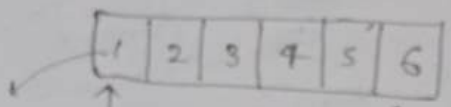
"abc" → output → "cba"

Program:

```
public import java.util.*;
    class Classroom {
public static String reverseString(String str) {
Stack<Character> s = new Stack<>();
long idx = 0;
while (idx < str.length()) {
    s.push(str.charAt(idx));
    idx++;
}
StringBuilder res = new StringBuilder("");
while (!s.isEmpty()) {
    char curr = s.pop();
    res.append(curr);
}
return res.toString();
}
psvm() {
String str = "abc";
String res = reverseString(str);
sop(res);
}
}
```

Topic : QUEUE

* Queue → FIFO



Front (exit)

Rear (entry)

Operations

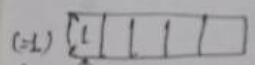
Add O(1) (Enque)

Remove O(1) (Dequeue)

Peek O(1) (Front)

Implementation

Array
X
remove O(n) / add O(1)
fixed-size



LinkedList
✓

Stack
✓

Array Implementation program:

```
public class QueueB {
    static class Queue {
    static int arr[];
    static int size;
    static int rear;
    Queue(int n) {
        arr = new int[n];
        size = n;
        rear = -1;
    }
    public static boolean isEmpty() {
        return rear == -1;
    }
// add
    public static void add(int data) {
        if(rear == size-1) {
            sop("queue is full");
            return;
        }
        rear = rear + 1;
        arr[rear] = data;
    }
//remove
    public static int remove() {
        if(isEmpty()) {
            sop("empty queue");
            return -1;
    }
```

```
int front = arr[0];
for(int i=0; i<rear; i++) {
    arr[i] = arr[i+1];
}
rear = rear - 1;
return front;
}
//peek
public static int peek() {
if (isEmpty()) {
    sop("empty queue");
    return -1;
}
return arr[0];
}}
psvm() {
Queue q = new Queue(5);
    q.add(1);
    q.add(2);
    q.add(3);
while(!q.isEmpty()) {
    sop(q.peek());
    q.remove();
}}
```

output
1
2
3

Circular Queue Implementation:

program:

```
public class QueueB {
static class Queue {
static int arr[];
static int size;
static int rear;
static int front;
Queue(int n) {
arr = new int[n];
size = n;
rear = -1;
front = -1;
}
public static boolean isEmpty() {
return rear == -1 && front == -1;
}
public static boolean isFull() {
return (rear+1) % size == front;
}
//add
public static void add(int data) {
if (isFull()){
System.out.println("queue is full");
return;
}
//add 1st element
if (front == -1) {
front = 0;
}
```

```
        rear = (rear + 1) % size;
        arr[rear] = data;
    }

// remove
public static int remove() {
    if (isEmpty()) {
        sop("empty queue");
        return -1;
    }
int result = arr[front];
// last element delete
if (rear == front) {
    rear = front = -1;
} else {
    front = (front + 1) % size;
}
return result;
}
// Peek
public static int peek() {
    if (isEmpty()) {
        sop("empty queue");
        return -1;
    }
    return arr[front];
}
```

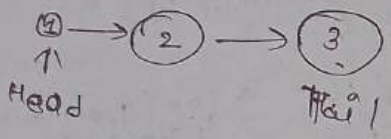
rear idx front idx

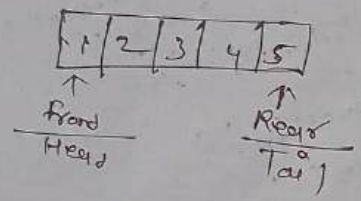
```
psum() {
Queue q = new Queue(5);
  q.add(1);
  q.add(2);
  q.add(3);
  sop(q.remove());
  q.add(4);
  sop(q.remove());
  q.add(5);
while(!q.isEmpty()){
  sop(q.peek());
  q.remove();
  }
  }
```

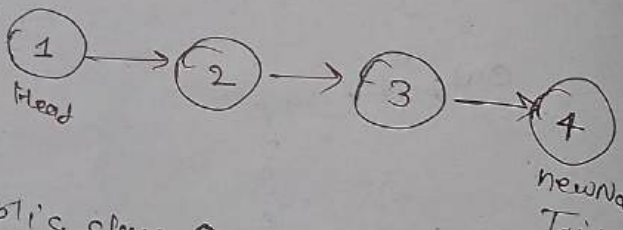
Output:

1
2
3
4
5

Ques Queue using Linked List





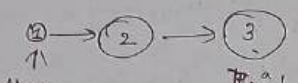


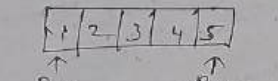
Program:

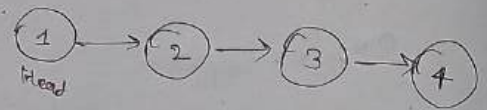
```
public class Queue {
    static class Node {
        int data;
        Node next;
        Node(int data) {
            this.data = data;
            this.next = null;
        }
    }
    static class Queue {
        static Node head = null;
        static Node tail = null;
    }
```

```
// add
public static void add(int data) {
    Node newNode = new Node(data);
    if(head == null) {
        head = tail = newNode;
        return;
    }
    tail.next = newNode;
    tail = newNode;
}
// remove
public static int remove() {
    if(isEmpty()) {
        System.out.println("empty queue");
        return -1;
    }
    int front = head.data;
    // single element
    if(tail == head) {
        tail = head = null;
    } else {
        head = head.next;
    }
    return front;
}
// peek
public static int peek() {
    if(isEmpty()) {
        System.out.println("empty queue");
        return -1;
    }
}
```

Que: Queue using Linked List







Program:

```
public class Queue {
    static class Node {
        int data;
        Node next;
        Node(int data) {
            this.data = data;
            this.next = null;
        }
    }
    static class Queue {
        static Node head = null;
        static Node tail = null;
    }
```

```
// add
public static void add(int data) {
    Node newNode = new Node(data);
    if (head == null) {
        head = tail = newNode;
        return;
    }
    tail.next = newNode;
    tail = newNode;
}
// remove
public static int remove() {
    if (isEmpty()) {
        sop("empty queue");
        return -1;
    }
    int front = head.data;
    // single element
    if (tail == head) {
        tail = head = null;
    } else {
        head = head.next;
    }
    return front;
}
// peek
public static int peek() {
    if (isEmpty()) {
        sop("empty queue");
        return -1;
    }
}
```

```
return head.data;
}}
psvm(String args[]){
Queue q = new Queue(8);
q.add(1);
q.add(2);
q.add(3);
while(!q.isEmpty()){
    sop(q.peek());
    q.remove();
}
}
```

Output = 1
2
3

Implementation of Queue

time = O(1)

```
import java.util.*;
public class QueueB {
    public static void main(String args[]){
Or  Queue<Integer> q = new ArrayDeque<>();
    Queue<Integer> q = new LinkedList<>();
        q.add(1);
        q.add(2);
        q.add(3);
    while(!q.isEmpty()){
        sop(q.peek());
        q.remove();
    }
    }
}
```

Output 1
2
3

Queue

Qus 2. Queue using 2 stacks

Program: import java.util.*;

```
public class QueueUs {
    static class Queue {
        static Stack<Integer> s1 = new Stack<>();
        static Stack<Integer> s1 = new Stack<>();
        public static boolean isEmpty() {
            return s1.isEmpty();
        }
```

Add → O(n)

```
        public static void add(int data) {
            while(!s1.isEmpty()) {
                s2.push(s1.pop());
            }
            s1.push(data);
            while(!s2.isEmpty()){
                s1.push(s2.pop());
            }
        }
```

```
// remove O(1)
public static int remove() {
    if(isEmpty()) {
        sop("queue empty");
        return -1;
    }
    return s1.pop();
}
// peek O(1)
public static int peek() {
    if(isEmpty()) {
        sop("queue empty");
        return -1;
    }
    return s1.peek();
}
psvm() {
    Queue q = new Queue();
    q.add(1);
    q.add(2);
    q.add(3);
    while(!q.isEmpty()) {
        sop(q.peek());
        q.remove();
    }
}
```

output

1
2
3

Ques 4.

First non-repeating Letter in a stream of characters

Input = a a b c c x b

a ——— a
a a ——— -1
a a b ——— b
a a b c ——— b
a a b c c ——— b
a a b c c x ——— b
a a b c c x b → x

Program

```
import java.util.*;
public class Classroom {
public static void printNonRepeating(String str){
int freq[] = new int[26];
Queue<Character> q = new LinkedList<>();
for(int i=0; i<str.length(); i++){
char ch = str.charAt(i);
q.add(ch);
freq[ch-'a']++;
while(!q.isEmpty() && freq[q.peek()-'a']>1)
q.remove();
}
if(q.isEmpty()){
sop(-1+" ");
}else{
sop(q.peek()+" ");
}
}
sop();
}
psvm(){
String str="aabccxb";
printNonRepeating(str);
}
}
```

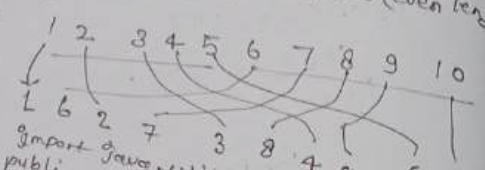
Output

a -1 b b b b x

Que 5

Interleave 2 Halves of a Queue (even length)

Input:



Output 1 6 2 7 3 8 4 9 5 10

Programe

```
import java.util.*;
public class Classroom{
public static void interLeave(Queue<Integer>q){
Queue<Integer> firstHalf = new LinkedList<>();
int size = q.size();
for(int i=0; i<size/2; i++){
firstHalf.add(q.remove());
}
while(!firstHalf.isEmpty()){
q.add(firstHalf.remove());
q.add(q.remove());
}}
psvm(){
Queue<Integer> q = new LinkedList<>();
q.add(1);
q.add(2);
q.add(3);
q.add(4);
q.add(5);
q.add(6);
q.add(7);
q.add(8);
q.add(9);
q.add(10);
interLeave(q);
while(!q.isEmpty()){
sop(q.remove()+" ");
}
sop();
}
}
```

Question 6

Queue Reversal 1 2 3 4 5

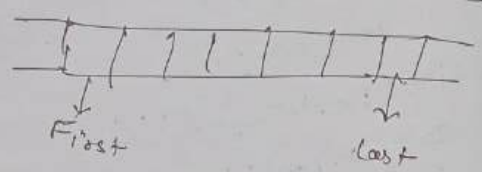
output : 5 4 3 2 1

Program:

```
import java.util.*;
public class Classroom{
public static void reverse(Queue<Integer>q)
{
Stack<Integer> s = new stack<>();
while(!q.isEmpty()){
s.push(q.remove());
}
while(!s.isEmpty()){
q.add(s.pop());
}}
psvm(){
Queue<Integer> q = new LinkedList<>();
q.add(1);
q.add(2);
q.add(3);
q.add(4);
q.add(5);
reverse(q);
while(!q.isEmpty()){
Sop(q.remove()+" ");
}
Sop();
}
}
```

output : 5 4 3 2 1

* **Deque** (Double ended queue)



Operation

addFirst ()
addLast () .
removeFirst()
removeLast()
getFirst()
getLast()

Program:

```
import java.util.*;
public class Classroom {
  psvm ( ) {
    Deque <Integer> deque = new LinkedList<>();
    deque.addFirst (1);
    deque.addFirst (2);
    SOP(deque);                 // output [2, 1]
    deque.removeFirst();
    SOP(deque);          ---> [2, 1] output [1]
    → Comment out karni hai
    deque.addLast (3);
    deque.addLast (4);
  }
}
```

Output : 2 1 3 4

Question 7

Stack ~~& Queue~~
using Deque

Program:

```
import java.util.*;
public class Classroom {
    static class Stack {
        Deque<Integer> deque = new LinkedList<>();
        public void push(int data) {
            deque.addLast(data);
        }
        public int pop() {
            return deque.removeLast();
        }
        public int peek() {
            return deque.getLast();
        }
    }
    psvm() {
        Stack S = new Stack();
        S.push(1);
        S.push(2);
        S.push(3);
        Sop("peek" + S.peek());
        Sop(S.pop());
        Sop(S.pop());
        Sop(S.pop());
    }
}
```

Output

```
peek = 3
3
2
1
```

Topic : Greedy Approach

19/06/24

Q. Fractional Knapsack

Given the weights and value of N items, put these items in a knapsack of capacity W to get the maximum total value in the knapsack

value = [60, 100, 120]
weight = [10, 20, 30]
W = 50

answer output = 240

Sol:

20+20+10 => 50

60+100+80 = 240

120/30 x 20 => 80

Program:

```
import java.util.*;
public class Classroom {
psvm( ) {
int val[] = {60, 100, 120};
int weight[] = {10, 20, 30};
int W = 50;
double ratio[][] = new double[val.length][2];

for(int i=0; i<val.length; i++) {
ratio[i][0] = i;
ratio[i][1] = val[i]/(double)weight[i];
}

//ascending order
Arrays.sort(ratio, Comparator.comparingDouble(o -> o[1]));

int capacity = W;
int finalval = 0;
for(int i=ratio.length-1; i>=0; i--){
int idx = (int) ratio[i][0];
if(capacity >= weight[idx]){
finalVal += val[idx];
capacity -= weight[idx];
}else{
finalVal += (ratio[i][1] * capacity);
capacity = 0;
break;
}
}
sop("final value=" + finalval);
}
}
```

Q.2

Min Absolute Difference Pairs

Given two arrays A and B of equal length. Pair each element of array A to an element in array B, such that sum S of absolute difference of all the pair is minimum.

Ex 1

A = [1, 2, 3]
B = [2, 1, 3]

|1-2| = 1
|2-1| = 1
|3-3| = 0

ans = 0
output

1+1+0 = 2

Case 1: |1-2| + |2-1| + |3-3| = 1+1+0 = 2
Case 2: |1-3| + |2-1| + |3-2| = 2+1+1 = 4
Case 3: |1-1| + |2-2| + |3-3| = 0+0+0 = 0

minimum value = 0

Ex 2

A = [4, 1, 8, 7]
B = [2, 3, 6, 5]

Step 1 → Sort

A = [1, 4, 7, 8] B = [2, 3, 5, 6]

|1-2| = 1
|4-3| = 1
|7-5| = 2
|8-6| = 2

⇒ 1+1+2+2 = 6 min value

Program:

```
import java.util.*;
public class Classroom {
    public static void main(String args[]) {
        int A[] = {1, 2, 3};
        int B[] = {2, 1, 3};
        Arrays.sort(A);
        Arrays.sort(B);
        int minDiff = 0;
        for(int i=0; i<A.length; i++) {
            minDiff += Math.abs(A[i] - B[i]);
        }
        System.out.println("min absolute diff. of pairs" + minDiff);
    }
}
```

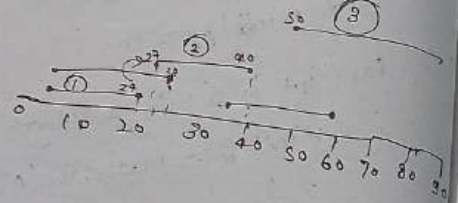
Output = 0

Max Length Chain of Pairs

You are given n pairs of number. In every pair, the first number is always smaller than the second number. A pair (c,d) can come after pair (a,b) if b < c. Find the longest chain which can be formed from a given set of pairs.

pair
(5, 24)
(39, 60)
(5, 28)
(27, 40)
(50, 90)

Output = 3



Program:

```
import java.util.*;
public class Classroom {
    P sum ( ){
int pairs[][] = {{5,24},{39,60},{5,28},
                 {27,40},{50,90}};
Arrays.sort(pairs, Comparator.comparing
                    Double(o -> o[1]));
int chainLen = 1;
int chainEnd = pairs[0][1];
for (int i=1; i<pairs.length; i++){
if (pairs[i][0] > chainEnd){
    chainLen++;
    chainEnd = pairs[i][1];
    }}
sop ("max length of chain :" +chainLen);
    }
}
```

Ques 4 Indian Coins

We are given an infinite supply of denominations [1, 2, 5, 10, 20, 50, 100, 500, 2000]

Find min no of coins/notes to make change for a value V.

V = 121

ans = 3 (100 + 20 + 1)

output

V = 590

ans = 4 (500 + 50 + 20 + 20)

Program:

```
import java.util.*;
public class Classroom {
    psvm () {
        Integer coins[] = {1, 2, 5, 10, 20, 50, 100, 500, 2000};
        Arrays.sort(coins, Comparator.reverseOrder());
        int countOfCoins = 0;
        int amount = 590;
        ArrayList<Integer> ans = new ArrayList<>();
        for(int i=0; i<coins.length; i++) {
            if (coins[i] <= amount) {
                while (coins[i] <= amount) {
                    countOfCoins++;
                    ans.add(coins[i]);
                    amount -= coins[i];
                }
            }
        sop ("total (min) coins used =" + countOfCoins);
        for (int i=0; i<ans.size(); i++) {
            sop(ans.get(i) + " ");
        }
        sop();
    }
}
```

Ques 4 Indian Coins

We are given an infinite supply of denomination [1, 2, 5, 10, 20, 50, 100, 500, 2000]

Find min no of coins/notes to make change for a value V.

V = 121

ans = 3 (100 + 20 + 1) output

V = 590

ans = 4 (500 + 50 + 20 + 20)

Program:

```
import java.util.*;
public class Classroom {
    psvm () {
        Integer coins[] = {1, 2, 5, 10, 20, 50, 100, 500, 2000};
        Arrays.sort(coins, Comparator.reverseOrder());
        int countOfCoins = 0;
        int amount = 590;
        ArrayList<Integer> ans = new ArrayList<>();
        for(int i=0; i<coins.length; i++) {
            if (coins[i] <= amount) {
                while(coins[i] <= amount) {
                    countOfCoins++;
                    ans.add(coins[i]);
                    amount -= coins[i];
                }
            }
        }
        SOP("total (min) coins used =" + countOfCoins);
        for (int i=0; i<ans.size(); i++) {
            SOP(ans.get(i) + " ");
        }
        SOP();
    }
}
```

Q. Job Sequencing Problem

Given an array of jobs where every job has a deadline and profit if the job is finished before the deadline. It is also given that every job takes a single unit of time, so the minimum possible deadline for any job is 1. Maximize the total profit if only one job can be scheduled at a time.

Job A = 4, 20
Job B = 1, 10
Job C = 1, 40
Job D = 1, 30

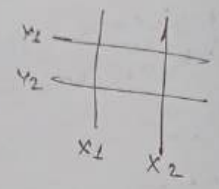
ans = C, A

Program:

```
import java.util.*;
public class Classroom {
    static class Job {
        int deadline;
        int profit;
        int id;
        public Job(int i, int d, int p) {
            id = i;
            deadline = d;
            profit = p;
        }
    }
    public static void main(String[] args) {
        int jobsInfo[][] = {{4,20},{1,10},{1,40},{1,30}};
        ArrayList<Job> jobs = new ArrayList<>();
        for (int i = 0; i < jobsInfo.length; i++) {
            jobs.add(new Job(i, jobsInfo[i][0], jobsInfo[i][1]));
        }
        Collections.sort(jobs, (obj1, obj2) -> obj2.profit - obj1.profit);
        ArrayList<Integer> seq = new ArrayList<>();
        int time = 0;
        for (int i = 0; i < jobs.size(); i++) {
            Job curr = jobs.get(i);
            if (curr.deadline > time) {
                seq.add(curr.id);
                time++;
            }
        }
    }
}
```

```
sop("max jobs = " + seq.size());
for(int i=0; i<seq.size(); i++){
    soPrint(seq.get(i) + " ");
}
sop();
}
```

Q. chocola Problem:



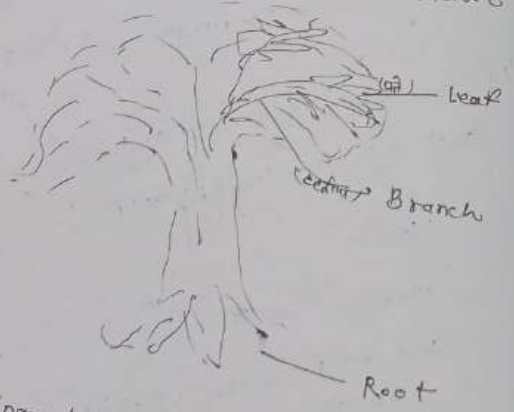
Output = 42

Program:

```
import java.util.*;
public class Classroom{
psvm(){
    int n=4, m=6;
    Integer costVer[] = {2,1,3,1,4};
    Integer costHor[] = {4,1,2};
Arrays.sort(costVer, Collections.reverseOrder());
Arrays.sort(costHor, Collections.reverseOrder());
int h=0, v=0;
int hp=1, vp=1;
int cost=0;
while(h<costHor.length && v<costVer.length){
if(costVer[v] <= costHor[h]){
    cost += (costHor[h] * vp);
    hp++;
    h++;
}else{
    cost += (costVer[v] * hp);
    vp++;
    v++;
}}
while(h<costHor.length){
    cost += hp++ * (costHor[h] * vp);
    h++;
}
while(v<costVer.length){
    cost += (costVer[v] * hp);
    vp++;
    v++;
}sop("min cost of cuts = " + cost);}}
```

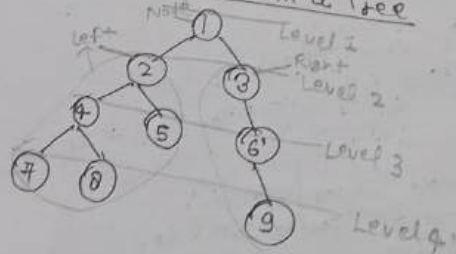
Topic : Binary Tree

* Binary tree Hierarchical Data Structure



* Binary tree at least 2 children

• Levels & Subtree in a Tree



Pop Quiz

Ans

(a) Children of 4 : 7, 8

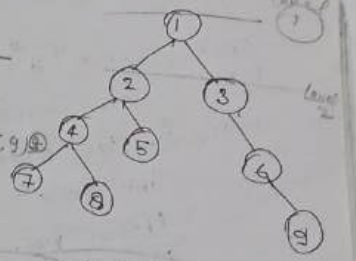
(b) No of Leaves : (7,8,5,9) 4

(c) Parent of 6 : 3

(d) Level of 2 : 2

(e) Subtrees of 1 & 2 : (2,4,5,7,8) (3,6,9)
(4,7,8) (5)
(4,2,1)

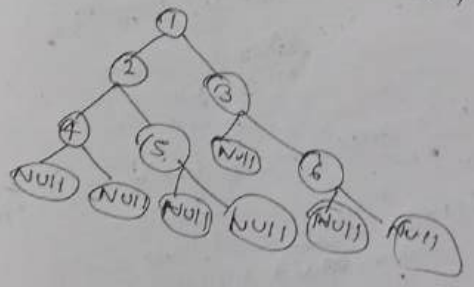
(f) Ancestors of 8 : 1 (Left) 1 (Right)



Build Tree PreOrder :

-1 = NULL

1, 2, 4, -1, -1, 5, -1, -1, 3, -1, 6, -1, -1



Build tree Node program

O(n)

= preOrder Sequence

```
import java.util.*;
public class ClassRoom
static class Node{
    int data;
    Node left;
    Node right;
    Node(int data){
        this.data = data;
        this.left = null;
        this.right = null;
    }
}
static class BinaryTree{
    static int idx = -1;
    public static Node buildTree(int nodes[]){
        idx++;
        if(nodes[idx] == -1){
            return null;
        }
        Node newNode = new Node(nodes[idx]);
        newNode.left = buildTree(nodes);
        newNode.right = buildRoom(nodes);
        return newNode;
    }
}
psvm(){
int node[] = {1,2,4,-1,-1,5,-1,-1,3,
              -1,6};
BinaryTree tree = new BinaryTree();
Node root = tree.buildTree(nodes);
SOP(root.data);
}
}
```

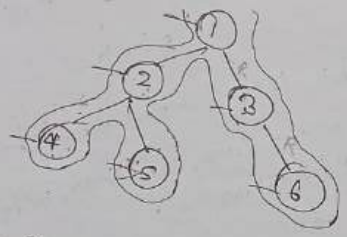
Output =

Tree Traversals

a. PreOrder

Recursive Print

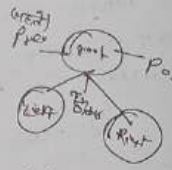
1. Root
2. Left Subtree
3. Right Subtree



Output: 1 2 4 5 3 6

```
public static void preOrder(Node root){
    if (root == null){
        return;
    }
    System.out.print(root.data + " ");
    preOrder(root.left);
    preOrder(root.right);
}
```

last line print tree.preOrder(root)



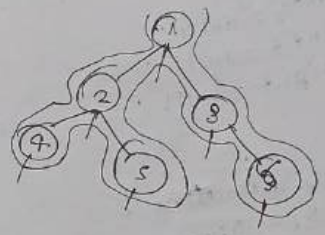
O(n)

Tree Traversals

b. InOrder

1st — Left Subtree
2nd — Root
3rd — Right Subtree

O(n)



Output: 4 2 5 1 3 6

```
public static void inOrder(Node root){
    if (root == null){
        return;
    }
    inOrder(root.left);
    System.out.print(root.data + " ");
    inOrder(root.right);
}
tree.inOrder(root)
```

Programs

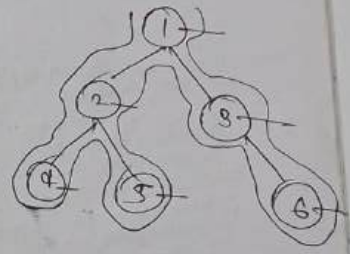
```
public class BinaryTreeB {
    static class Node {
        int data;
        Node left;
        Node right;
        Node(int data) {
            this.data = data;
            this.left = null;
            this.right = null;
        }
    }
    static class BinaryTree {
        static int idx = -1;
        public static Node buildTree(int nodes[]) {
            idx++;
            if (nodes[idx] == -1) {
                return null;
            }
            Node newNode = new Node(nodes[idx]);
            newNode.left = buildTree(nodes);
            newNode.right = buildTree(nodes);
            return newNode;
        }
    }
    psvm() {
        int nodes[] = {1, 2, 4, -1, -1, 5, -1, -1, 3, -1, 6, -1, -1};
        BinaryTree tree = new BinaryTree();
        Node root = tree.buildTree(nodes);
        SOP(root.data);
    }
}
```

output = 1

Tree Traversals

c. Postorder

Left Subtree
Right Subtree
Root



output : 4 5 2 6 3 1

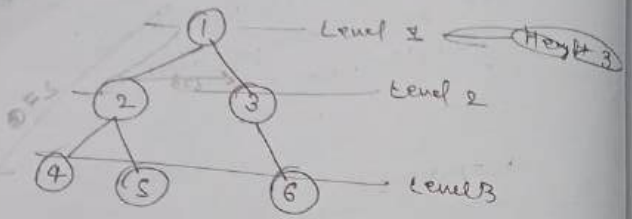
```
public static void postOrder(Node root) {
    if (root == null) {
        return;
    }
    postOrder(root.left);
    postOrder(root.right);
    SOP(root.data + " ");
}
```

tree.printPostOrder(root);

Tree Traversals

d. Level Order

O(n)

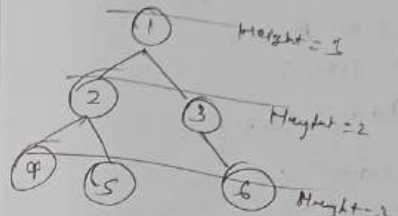


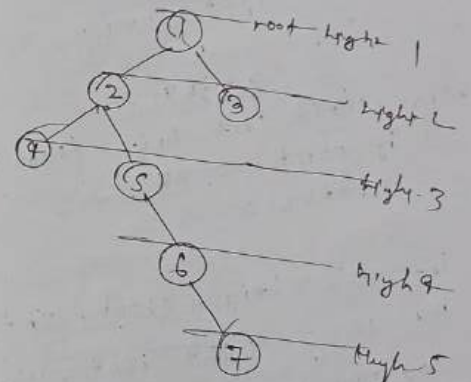
1 2 3 4 5 6

next = null; see degree storage

```
public static void levelOrder(Node root){
  if(root == null){ return; }
  Queue<Node> q = new LinkedList<>();
  q.add(root);
  q.add(null);
  while(!q.isEmpty()){
    Node currNode = q.remove();
    if(currNode == null){
      System.out.println();
      if(q.isEmpty()){
        break;
      }else{
        q.add(null);
      }
    }else{
      System.out.print(currNode.data+" ");
      if(currNode.left != null){
        q.add(currNode.left);
      }
      if(currNode.right != null){
        q.add(currNode.right);
      }
    }
  }
}
```

Height of a Tree :

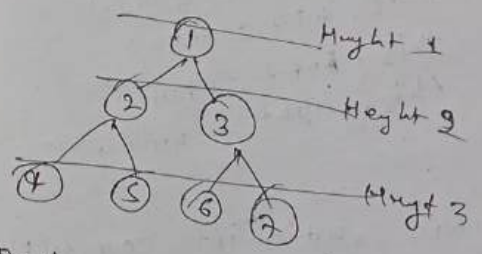




Program:

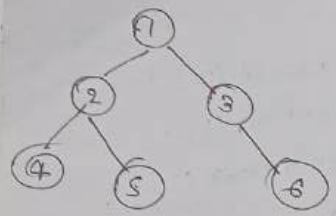
```
import java.util.*;
public class Classroom {
    static class Node {
        int data;
        Node left, right;
        public Node(int data) {
            this.data = data;
            this.left = null;
            this.right = null;
        }
    }
    // Height find
    public static int height(Node root) {
        if(root == null) {
            return 0;
        }
        int lh = height(root.left);
        int rh = height(root.right);
        return Math.max(lh, rh)+1;
    }
```

```
// Print statement
PSVM() {
    Node root = new Node(1);
    root.left = new Node(2);
    root.right = new Node(3);
    root.left.left = new Node(4);
    root.left.right = new Node(5);
    root.right.left = new Node(6);
    root.right.right = new Node(7);
    SOP(height(root));
}
```



Output = 3

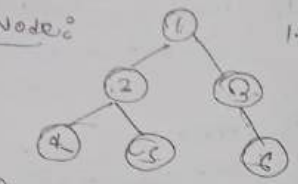
9. Count of Nodes



Program:

```
import java.util.*;
public class Classroom {
    static class Node {
        int data;
        Node left, right;
        public Node(int data) {
            this.data = data;
            this.left = null;
            this.right = null;
        }
    }
    public static int count(Node root) {
        if (root == null) {
            return 0;
        }
        int leftCount = count(root.left);
        int rightCount = count(root.right);
        return leftCount + rightCount + 1;
    }
    psvm() {
        Node root = new Node(1);
        root.left = new Node(2);
        root.right = new Node(3);
        root.left.left = new Node(4);
        root.left.right = new Node(5);
        root.right.left = new Node(6);
        root.right.right = new Node(7);
        sop(count(root));
    }
}
```

Sum of Nodes:



1+2+3+4+5+6 = 21

Output

O(n)

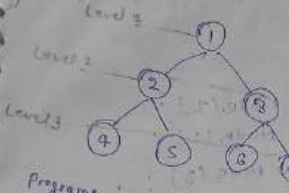
Program:

```
import java.util.*;
public class classroom {
    static class Node {
        int data;
        Node left, right;
        public Node(int data) {
            this.data = data;
            this.left = null;
            this.right = null;
        }
    }
    public static int Sum(Node root) {
        if(root == null) {
            return 0;
        }
        int leftSum = Sum(root.left);
        int rightSum = Sum(root.right);
        return leftSum + rightSum + root.data;
    }
    psvm() {
    }
    Node root = new Node(1);
    root.left = new Node(2);
    root.right = new Node(3);
    root.left.left = new Node(4);
    root.left.right = new Node(5);
    root.right.left = new Node(6);
    root.right.right = new Node(7);
    sop(sum(root));
}
```

Output = 28

* 21. Topic: Binary Tree *

Q. Kth Level of a Tree



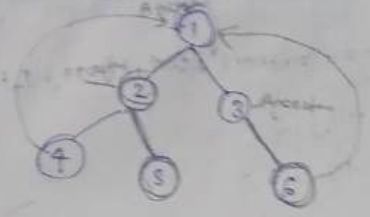
k = 3

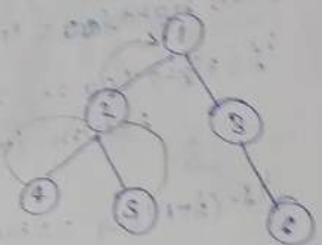
output => 4 5 6 7

Program:

```
import java.util.*;
public class Classroom5 {
    static class Node {
        int data;
        Node left, right;
        public Node(int data) {
            this.data = data;
            this.left = null;
            this.right = null;
        }
    }
    public static void kLevel(Node root, int level, int k) {
        if (root == null) {
            return;
        }
        if (level == k) {
            SOPln(root.data + " ");
            return;
        }
```

Lowest Common Ancestor





n1 = 4, n2 = 6

output = 1

n1 = 4, n2 = 5

output = 2

Path 1(n1) =

1	2	4		

Path 2(n2) =

1	3	6		

output = 1

Programme

```
import java.util.*;
public class Classroom {
    static class Node {
        int data;
        Node left, right;
        public Node(int data) {
            this.data = data;
            this.left = null;
            this.right = null;
        }
    }
    public static Node lowestCommon(Node root, int n1,
                                    int n2)
    {
        if(root == null || root.data == n1 || root.data == n2)
        {
            return root;
        }
```

```
Node left_lca = lowestCommon(root.left, n1, n2);
Node right_lca = lowestCommon(root.right, n2, n2);
if (right_lca == null) {
    return left_lca;
}
if (left_lca == null) {
    return right_lca;
}
return root;
}
P sum() {
Node root = new Node(1);
root.left = new Node(2);
root.right = new Node(3);
root.left.left = new Node(4);
root.left.right = new Node(5);
root.right.left = new Node(6);
root.right.right = new Node(7);
int n1 = 4, n2 = 7;
SOP(lowestCommon(root, n1, n2).data);
}
```

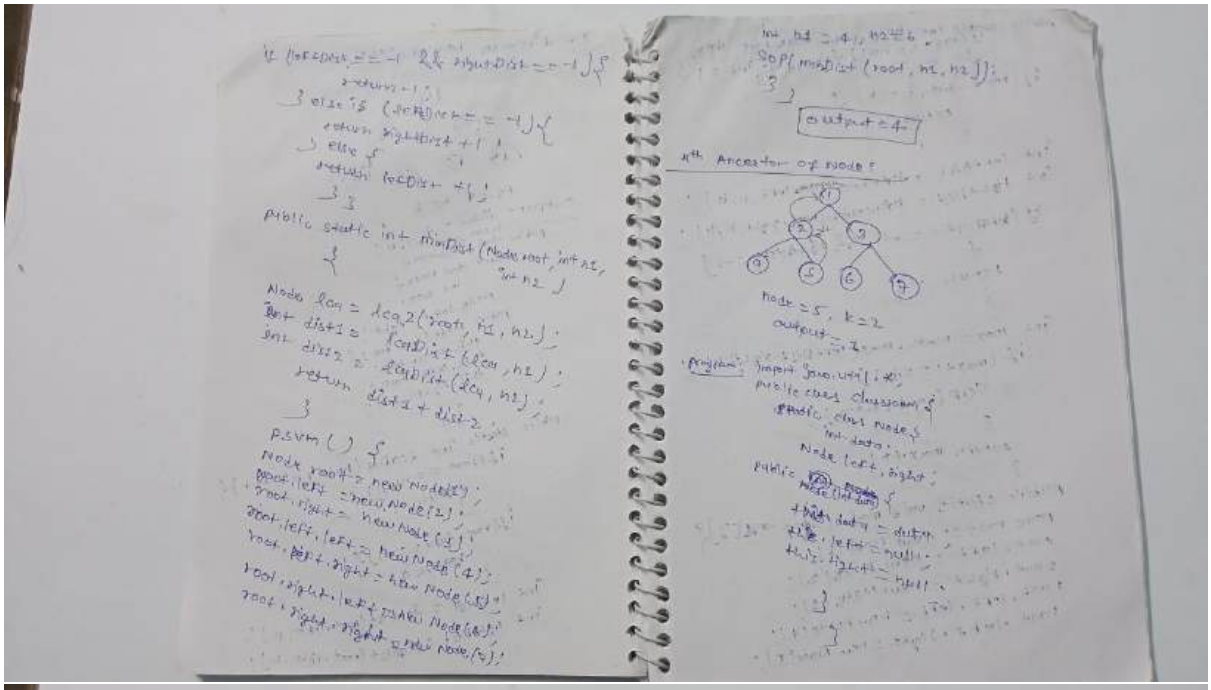
Output = 1

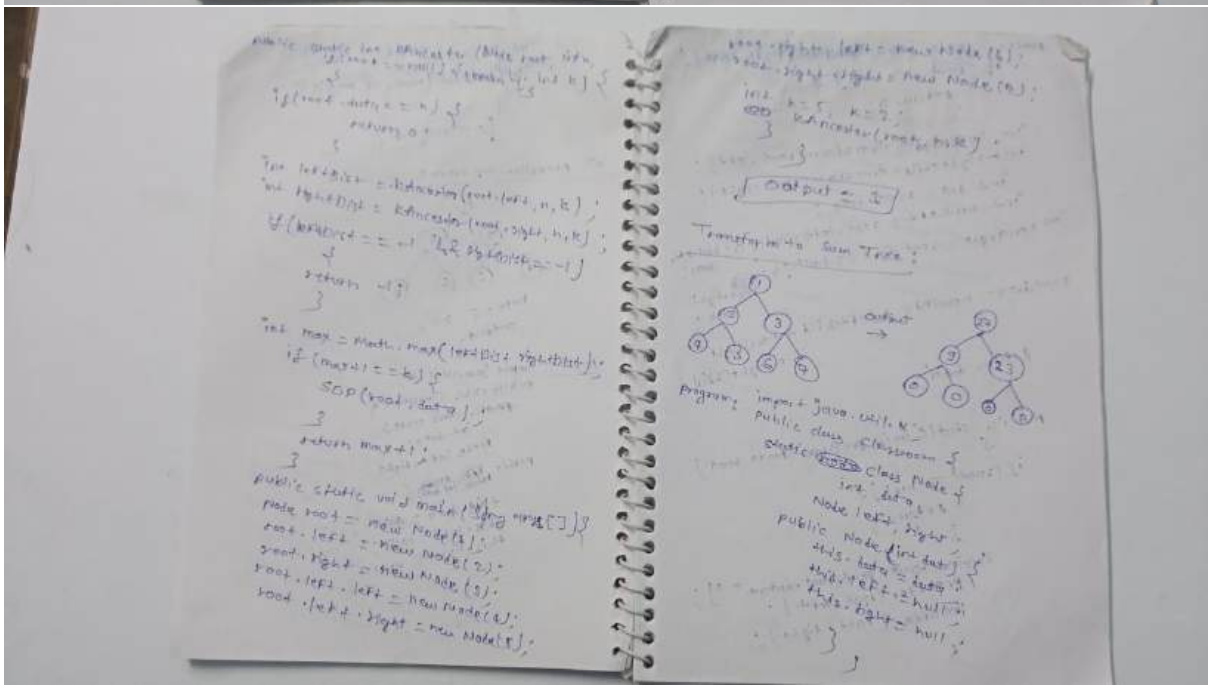
Min Distance between nodes

n1 = 4 n2 = 6

Output: distem = 4

```
import java.util.*;
public class Classroom {
    static class Node {
        int data;
        Node left, right;
        public Node(int data) {
            this.data = data;
            this.left = null;
            this.right = null;
        }
    }
    public static int lcaDis(Node root, int n) {
        if(root == null) {
            return -1;
        }
        if(root.data == n) {
            return 0;
        }
        int leftDist = lcaDist(root.left, n);
        int rightDist = lcaDist(root.right, n);
```





```
public static int transform(Node root) {
    if (root == null) {
        return 0;
    }
    int leftChild = transform(root.left);
    int rightChild = transform(root.right);
    int data = root.data;
    int newLeft = root.left == null ? 0 : root.left.data;
    int newRight = root.right == null ? 0 : root.right.data;
    root.data = newLeft + leftChild + newRight + rightChild;
    return data;
}
public static void preOrder(Node root) {
    if (root == null) {
        return;
    }
    System.out.print(root.data + " ");
    preOrder(root.left);
    preOrder(root.right);
}
```

```
public static void main(String args[]) {
    Node root = new Node(1);
    root.left = new Node(2);
    root.right = new Node(3);
    root.left.left = new Node(4);
    root.left.right = new Node(5);
    root.right.right = new Node(6);
    root.right.left = new Node(7);
    transform(root);
    preOrder(root);
}
```

Output:

22 5 20 13 0 0

Ques 1 : Check if a binary tree is univalued or not

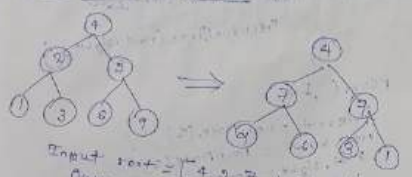
Yes, No

Program: Import java.util.*;

```
class Solution {
    static class Node {
        int data;
        Node left;
        Node right;
    }
    static Node newNode(int data) {
        Node temp = new Node();
        temp.data = data;
        temp.left = temp.right = null;
        return(temp);
    }
    static boolean isUnivalTree(Node root) {
        if (root == null) {
            return true;
        }
        if (root.left != null && root.data != root.left.data)
            return false;
        return isUnivalTree(root.left) && isUnivalTree(root.right);
    }
    public static void main(String[] args) {
        Node root = newNode(1);
        root.left = newNode(1);
        root.right = newNode(1);
        root.left.left = newNode(1);
        root.left.right = newNode(1);
        root.right.right = newNode(1);
        if (isUnivalTree(root)) {
            System.out.println("Yes");
        }
        else {
            System.out.println("No");
        }
    }
}
```

Output: Yes

226. Invert Binary Tree



Input: root = [4,2,7,1,3,6,9]
Output = [4,7,2,9,6,3,1]

```
Class Solution {
public TreeNode invertTree(TreeNode root){
if(root == null) return root;
TreeNode ln = invertTree(root.left);
TreeNode rn = invertTree(root.right);
root.left = rn;
root.right = ln;
return root;
}
}
```

94. Binary Tree Inorder Traversal

Input: root = [1,null,2,3]
Output: [1,3,2]

Program:

```
Class Solution {
public List<Integer> inOrderTraversal
(TreeNode root) {
List<Integer> inOrder = new ArrayList<>();
if(root == null) return root;
inOrder.addAll(inOrderTraversal(root.left));
inOrder.add(root.val);
inOrder.addAll(inOrderTraversal(root.right));
return inOrder;
}
```

144. Binary Tree Preorder Traversal

Input: root = [1,null,2,3]
Output: [1,2,3]

Program:

```
Class Solution {
public List<Integer> preorderTraversal
(TreeNode root)
{
List<Integer> preOrder = new ArrayList<>();
if(root == null) return preOrder;
preOrder.add(root.val);
preOrder.addAll(preOrderTraversal(root.left));
preOrder.addAll(preOrderTraversal(root.right));
return preOrder;
}
}
```

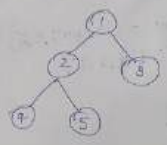
145. Binary Tree Postorder Traversal

Input: root = [1,null,2,3]
Output: [3,2,1]

Program:

```
Class Solution {
public List<Integer> postorderTraversal
(TreeNode root){
List<Integer> postOrder = new ArrayList<>();
if(root == null)
return postOrder;
postOrder.addAll(postorderTraversal(root.left));
postOrder.addAll(postorderTraversal(root.right));
postOrder.add(root.val);
return postOrder;
}
}
```

#43. Diameter of Binary Tree



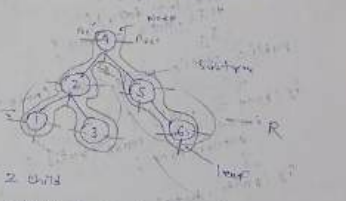
Input: root = [1,2,3,4,5]
Output: 3

Program:

```
class Solution {
    int max = 0;
    public int diameterOfBinaryTree(TreeNode root) {
        helper(root);
        return max-1;
    }
    public int helper(TreeNode root) {
        if(root == null) return 0;
        int left = helper(root.left);
        int right = helper(root.right);
        max = Math.max(max, 1+left+right);
        return 1 + Math.max(left, right);
    }
}
```

* Binary Search Tree → Topic *

* Time Complexity → O(H)



- At least 2 child
- a. Left subtree Nodes < Root
  b. Right subtree Nodes > Root
  c. Left, Right subtrees are also BST with no duplicates.

Note: Inorder Traversal of BST gives a sorted sequence.

Inorder output: 1 2 3 4 5 6

Build a BST

Values[] = {5, 1, 3, 4, 2, 7}

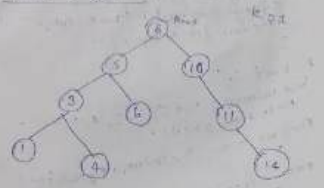
Program:

```
import java.util.*;
public class BST {
    static class Node {
        int data;
        Node left;
        Node right;
        Node(int data) {
            this.data = data;
        }
    }
    public static Node insert(Node root, int val) {
        if(root == null) {
            root = new Node(val);
            return root;
        }
        if(root.data > val) {
            // left subtree
            root.left = insert(root.left, val);
        } else {
            // right subtree
            root.right = insert(root.right, val);
        }
        return root;
    }
    public static void inOrder(Node root) {
        if(root == null) {
            return;
        }
        inOrder(root.left);
        System.out.print(root.data + " ");
        inOrder(root.right);
    }
    public static void main(String[] args) {
        int values[] = {5, 1, 3, 4, 2, 7};
        Node root = null;
        for(int i=0; i<values.length; i++) {
            root = insert(root, values[i]);
        }
        inOrder(root);
        System.out.println();
    }
}
```

Output: 1 2 3 4 5 7

Note: Search a BST O(H)
Root > key → Left
Root < key → Right
Root = null → Return false
Root = key → Return true

Search a BST



Root > key ⇒ left
root < key ⇒ Right
Root = Key ⇒ true
Root = null ⇒ False

Program:

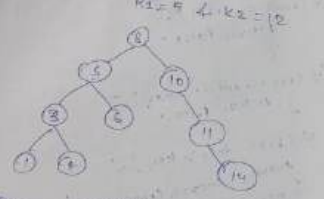
```
import java.util.*;
public class Classroom {
    static class Node {
        int data;
        Node left, right;
        public Node(int data) {
            this.data = data;
            this.left = null;
            this.right = null;
        }
    }
```

```
public static boolean search(Node root, int key)
{
if(root == null) {
    return false;
}
if(root.data == key) {
    return true;
}
if(root.data > key) {
    return search(root.left, key);
} else {
    return search(root.right, key);
}}
public Sum() {
int values[] = {5, 1, 3, 4, 2, 7};
Node root = null;
for(int i=0; i<values.length; i++)
{
root = insert(root, values[i]);
}
if(search(root, 1)) {
    sop("found");
} else {
    sop("not found");
}}}
```

output: found exist

Print in Range:

K1 = 5 & K2 = 12

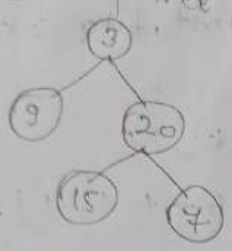


Program:

```
import java.util.*;
public class Classroom {
    static class BST {
        int data;
        Node left, right;
        public Node(int data) {
            this.data = data;
        }
    }
public static void printInRange(Node root, int k1,
                                int k2)
{
if(root == null) {
    return;
}
if(root.data >= k1 && root.data <= k2)
{
printInRange(root.left, k1, k2);
System.out.print(root.data + " ");
printInRange(root.right, k1, k2);
}
else if(root.data < k1) {
    printInRange(root.left, k1, k2);
} else {
    printInRange(root.right, k1, k2);
}
Psum() {
int value[] = {8, 5, 3, 1, 4, 6, 10, 11, 14};
Node root = null;
for(int i=0; i<values.length; i++) {
root = insert(root, values[i]);
}
printInRange(root, k1, k2);
}}
```

output: 5 6 8 10 11

102 Binary Tree Level Order Traversal

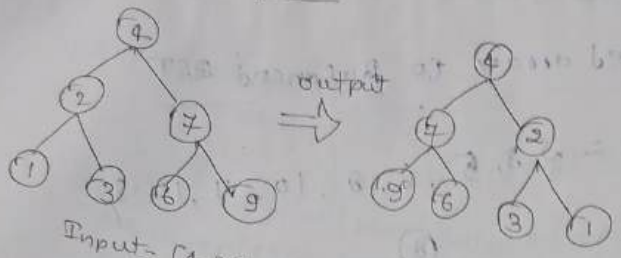


Input: root = [3, 9, 20, null, null, 15, 7]
Output: [[3], [9, 20], [15, 7]]

Sol:

```
class Solution {
public List<List<Integer>> levelOrder(TreeNode root)
{
List<List<Integer>> res = new ArrayList<>();
if (root == null) {
return res;
}
Queue<TreeNode> queue = new LinkedList<>();
queue.add(root);
while (!queue.isEmpty()) {
int levelsize = queue.size();
List<Integer> currlevel = new ArrayList<>();
for (int i=0; i<levelsize; i++) {
TreeNode currentNode = queue.poll();
currlevel.add(currentNode.val);
if (currentNode.left != null) {
queue.add(currentNode.left);
}
if (currentNode.right != null) {
queue.add(currentNode.right);
}
}
res.add(currlevel);
}
return res;
}
```

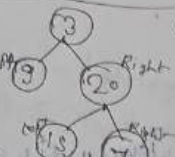
226 Invert Binary Tree :



Input = [4,2,7,1,3,6,9]
output = [4,7,2,9,6,3,1]

```
class solution {
public TreeNode invertTree(TreeNode root) {
 if(root == null) return root;
TreeNode left = invertTree(root.left);
TreeNode Right = invertTree(root.right);
 //swap root
 root.left = Right;
 root.right = Left;
 return root;
 }
}
```

404. Sum of Left Leaves :



only Left sum
9+15 = 24

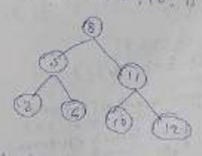
```
class solution {
public int sumofLeftLeaves(TreeNode root) {
 if(root == null) return root;
 int sum = 0;
if(root.left != null && root.left.left == null &&
 root.left.Right == null)
 sum += root.left.val;
sum += sumofleftleaves(root.left);
sum += sumofleftLeaves(root.right);
return sum;
}
}
```

C = Dennis Ritchie
Java = James Gosling
C++ = Bjarne Stroustrup
JS = Brendan Eich
PHP = Rasmus Lerdorf
PYTHON = Guido van Rossum

Topic Binary Search Tree

Sorted array to Balanced BST

arr = {3, 5, 6, 8, 10, 11, 12}



mid = (st + end) / 2

arr = {3, 5, 6, (8) 10, 11, 12}
st … mid … end

3 5 6
st mid end

Programs:

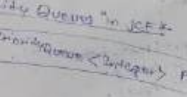
```
import java.util.*;
public class Classroom {
    static class Node {
        int data;
        Node left;
        Node right;
    public Node(int data) {
        this.data = data;
        this.left = this.right = null;
    }
}

public static void preorder(Node root)
{
    if (root == null) {
        return;
    }
    System.out.print(root.data + " ");
    preorder(root.left);
    preorder(root.right);
}
public static Node createBST(int arr[], int st, int end)
{   if (st > end) { return null; }
    int mid = (st + end) / 2;
    Node root = new Node(arr[mid]);
    root.left = createBST(arr, st, mid-1);
    root.right = createBST(arr, mid+1, end);
    return root;
}
public static void main(String args[]) {
    int arr[] = {3, 5, 6, 8, 10, 11, 12};
    Node root = createBST(arr, 0, arr.length-1);
    preorder(root);
    }
}
```

output: 8 5 3 6 11 10 12

Topic: (HEAPS)

* Heaps
* Priority Queues in JCF ← Topic

PriorityQueue<Integer> pq = new PriorityQueue<>();



* Priority Queue properties → Time Complexity
  - add() → O(logn)
  - remove() → O(logn)
  - peek() → O(1)

* Priority Queue हमेशा ascending Order में होती है Queue में

Priority Queues –

Program:

```
import java.util.PriorityQueue;
public class Classroom {
public static void main(String args[]) {
    PriorityQueue<Integer> pq = new PriorityQueue<>();
    pq.add(3);
    pq.add(4);
    pq.add(1);
    pq.add(7);
    while(!pq.isEmpty()) {
        System.out.println(pq.peek());
        pq.remove();
    }
  }
}
```

PriorityQueue<Integer> pq = new PriorityQueue<>(Comparator.reverseOrder());

output: 1 3 4 7 | Output: 7 4 3 1

```
import java.util.*;
PriorityQueue<Object> pq = new PriorityQueue<>();
static class Student implements
        Comparable<Student>
{
    String name;
    int rank;
    public Student(String name, int rank){
        this.name = name;
        this.rank = rank;
    }
    @Override
    public int compareTo(Student s2){
        return this.rank - s2.rank;
    }
}
public static void main(String args[]){
PriorityQueue<Student> pq = new PriorityQueue<>();
pq.add(new Student("A", 4));
pq.add(new Student("B", 5));
pq.add(new Student("C", 2));
pq.add(new Student("D", 12));
while(!pq.isEmpty()){
System.out.println(pq.peek().name+"->"
        +pq.peek().rank);
pq.remove();
}
}
```

Output:

C -> 2
A -> 4
B -> 5
D -> 12

Topic: Heap

Max heap
- more element
- more priority

Min heap
- less element
- high priority

Heap: Binary Tree at most 2 children
Filled -> left to Right

Heap Order Property:

Children >= Parent (minHeap)
Children <= Parent (maxHeap)

Heapify -> (Logn)

root = i
left = 2*i + 1
right = 2*i + 2

Heap Sort

Heapsort -> Sorting Program?

```
import java.util.*;
public class Classroom{
public static void heapify(int arr[], int i,
                           int size){
    int left = 2*i+1;
    int right = 2*i+2;
    int maxIdx = i;
    if(left < size && arr[left] > arr[maxIdx])
    {
        maxIdx = left;
    }
    if(right < size && arr[right] > arr[maxIdx])
    {
        maxIdx = right;
    }
    if(maxIdx != i){
        // swap
        int temp = arr[i];
        arr[i] = arr[maxIdx];
        arr[maxIdx] = temp;
        heapify(arr, maxIdx, size);
    }
}
```

```
public static void heapSort(int arr[]){
int n = arr.length;
for(int i=n/2; i>=0; i--){
  heapify(arr, i, n);
}
// step 2 - push largest at end
for(int i=n-1; i>0; i--){
  // swap largest (first) with last
  int temp = arr[0];
  arr[0] = arr[i];
  arr[i] = temp;
  heapify(arr, 0, i);
}
}
public static void main(String args[]){
int arr[] = {1,2,4,5,3};
heapSort(arr);
// print
for(int i=0; i<arr.length; i++){
  System.out.print(arr[i]+1+" ");
}
}
```

Output: 1 2 3 4 5

Quiz:

Nearby Cars:

We are given N points in a 2D plane where are locations of N cars. If we are at the origin, print the nearest K cars.

K = 2

C0 (3,3)
C1 (5,-1)
C2 (-2,4)

output: C0 & C2

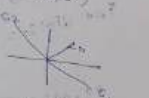
Sol:

C0 = $\sqrt{3^2+3^2}$ = 18

C1 = $\sqrt{(5)^2+(-1)^2}$ = 26

C2 = $\sqrt{(-2)^2+(4)^2}$ = 20

Origin = (0,0)



Program:

```
import java.util.*;
public class ClosestKcars{
  static class Point implements Comparable<Point>{
    int x;
    int y;
    int distSq;
    int idx;
    public Point(int x, int y, int distSq, int idx){
      this.x = x;
      this.y = y;
      this.distSq = distSq;
      this.idx = idx;
    }
```

```
@Override
public int compareTo(Point p2){
  return this.distSq - p2.distSq;
}
}
public static void main(String args[]){
  int pts[][] = {{3,3},{5,-1},{-2,4}};
  int k=2;
  PriorityQueue<Point> pq = new PriorityQueue<>();
  for(int i=0; i<pts.length; i++){
    int distSq = pts[i][0]*pts[i][0] + pts[i][1]*pts[i][1];
    pq.add(new Point(pts[i][0], pts[i][1], distSq, i));
  }
  // nearest k cars
  for(int i=0; i<k; i++){
    System.out.println("C"+pq.remove().idx);
  }
}
```

Quiz: Connect N Ropes

Given an N array of different lengths, the task is to compute these ropes into one rope with minimum cost, such that the cost to connect two ropes is equal to sum of their lengths.

ropes = {4, 3, 2, 6}

Output: 29

2+3 = 5
5+4 = 9
9+6 = 15
5+9+15 = 29

Program:

```
import java.util.*;
public class ClosestKcars{
public static void main(String[] args){
int ropes[] = {4,3,2,6};
PriorityQueue<Integer> pq = new PriorityQueue<>();
for(int i=0; i<ropes.length; i++){
  pq.add(ropes[i]);
}
int cost=0;
while(pq.size()>1){
  int min = pq.remove();
  int min2 = pq.remove();
  cost += min+min2;
  pq.add(min+min2);
}
System.out.println("cost of connecting ropes = "+cost);
}
}
```

Program: Q42

Weakest Soldier

We are given an m x n binary matrix of 1's (soldier) and 0's (civilians). The soldiers are positioned in front of the civilians. That is, all the 1's will appear to the left of all the 0's in each row.

A row i is weaker than a row j if one of the following is true:

- The number of soldiers in row i is less than the number of soldiers in row j.
- Both rows have the same number of soldiers and i < j.

Find the k weakest rows.

m = 4, n = 4, k = 2

1 0 0 0
1 1 1 1
1 0 0 0
1 0 0 0

answer output : row 0, row 2

Sol:

1 0 0 0 → i=0 → Row[ ] idx=0
1 1 1 1 → i=1 → Row[ ] idx=1
1 0 0 0 → i=2 → Row[ ] idx=2
1 0 0 0 → i=3 → Row[ ] idx=3

Solution = equal = (i, j)

Program:

```
import java.util.*;
public class classroom {
    static class Row implements Comparable<Row> {
        int soldiers;
        int idx;
        public Row(int soldiers, int idx) {
            this.soldiers = soldiers;
            this.idx = idx;
        }
        @Override
        public int compareTo(Row r2) {
            if (this.soldiers == r2.soldiers) {
                return this.idx - r2.idx;
            } else {
                return this.soldiers - r2.soldiers;
            }
        }
    }
    public static void main(String[] args) {
        int arr[][] = {{1,0,0,0},{1,1,1,1},{1,0,0,0},{1,0,0,0}};
        int k = 2;
        PriorityQueue<Row> pq = new PriorityQueue<>();
        for (int i = 0; i < arr.length; i++) {
            int count = 0;
            for (int j = 0; j < arr[0].length; j++) {
                count += arr[i][j] == 1 ? 1 : 0;
            }
            pq.add(new Row(count, i));
        }
        for (int i = 0; i < k; i++) {
            System.out.println("R" + pq.remove().idx);
        }
    }
}
```

Ques: Sliding Window Maximum

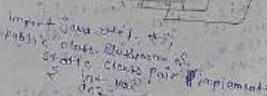
Maximum of all subarrays of size k

1, 2, 3, 4, 5, 6, 7, 8, 9, 10

k = 3

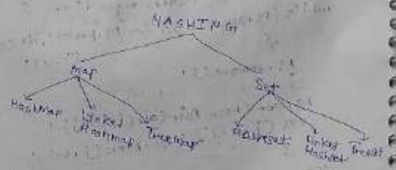
output: 3, 4, 5, 6, 7, 8, 9, 10

Sol:



```
import java.util.*;
public class SlidingWindow {
    static class Pair implements Comparable<Pair> {
        int val;
        int idx;
        public Pair(int val, int idx) {
            this.val = val;
            this.idx = idx;
        }
        @Override
        public int compareTo(Pair r2) {
            return r2.val - this.val;
        }
    }
    public static void main(String[] args) {
        int arr[] = {1,2,3,4,5,6,7,8,9,10};
        int k = 3;
        int res[] = new int[arr.length - k + 1];
        PriorityQueue<Pair> pq = new PriorityQueue<>();
        for (int i = 0; i < k; i++) {
            pq.add(new Pair(arr[i], i));
        }
        res[0] = pq.peek().val;
        for (int i = k; i < arr.length; i++) {
            while (pq.size() > 0 && pq.peek().idx <= (i - k)) {
                pq.remove();
            }
            pq.add(new Pair(arr[i], i));
            res[i - k + 1] = pq.peek().val;
        }
        // Print result
        for (int i = 0; i < res.length; i++) {
            System.out.print(res[i] + " ");
        }
        System.out.println();
    }
}
```

Topic : HASHING



HashMap

(Key, Value) (Hashmap java ke ander ek aisa data structure hota hai jo Unique table of information ko store karne ke kaam aata hai.)

Key	Value
Tea	10
Samosa	13
Pizza	250
Burger	50
Coffee	50

Note: HashMap unorder me output print karta hai.
Note: Key hamisha Unique rahita hai.
Note: Hashmap pair ke form me store karne ke liye used kiya jata hai.

HashMap Operations

Package: import java.util.HashMap

HashMap create:

```
HashMap<keyType, dataValueType> hm = new HashMap<>();
```

put(key, value) → O(1) : put ka used Insert or Addition ke used kiya jata hai.

Get → O(1) : Get ka used value ka search karne ke liye used hota hai.

containKey → O(1) : used check karta hai ki kiya mera map ke andar mein koi key exist karta hai ya nahi karti.
Output: true → exist
false → not exist

Remove(key) → O(1) : delete ke liye used hota hai.

Size → O(1) → (hm.size())

Empty → O(1) → (hm.isEmpty())
→ output false / true

HashMap create program

```
1. import java.util.HashMap;
2. import java.util.*;
3. public class classroom {
4.   public static void main(String args[]) {
     // create
5. HashMap<String, Integer> hm = new HashMap<>();
     // Insert or Add - O(1)
6. hm.put("India", 100);
7. hm.put("China", 150);
8. hm.put("US", 50);
9. System.out.println(hm);
     → output: {China=150, US=50, India=100}

   // Get - O(1) → Key search
10. int population = hm.get("India");
11. System.out.println(population);
     → output: 100

12. System.out.println(hm.get("Indonesia"));
     → output: null
```

Note: jo key exist nahi karti hai to output me null store ho jata hai.

```
   // containsKey - O(1)
13. System.out.println(hm.containsKey("India"));
     → output: true
14. System.out.println(hm.containsKey("Indonesia"));
     → output: false
```

10, 11, 12, 13, 14 → comment out karte hai.

```
   // Remove or Delete - O(1)
   System.out.println(hm.remove("China"));
   System.out.println(hm);
  }
}
   → output
   {China=150, US=50, India=100}
   150
   {US=50, India=100}

   // size
   System.out.println(hm.size());
   → output: 3

   // Empty
   System.out.println(hm.isEmpty());
  }
}
   → output: false
```

Iteration on HashMap

```
Set<Integer> keys = hm.keySet();
```

datatype — only key ko print kr ne ke used

Ex: "India", "china", "US", "Nepal"

Program:

```
import java.util.*;
public class Classroom{
public static void main(String args[]){
HashMap<String, Integer> hm = new HashMap<>();
hm.put("India", 100);
hm.put("china", 150);
hm.put("US", 50);
hm.put("Indonesia", 6);
hm.put("Nepal", 5);
//Iterate
Set<String> keys = hm.keySet();
System.out.println(keys);
```

↳ output: {India, china, US, Indonesia, Nepal}

```
// loop
for(String k : keys){
System.out.println("key"+k+"value="+hm.get(k));
}
}
}
```

output:

[china, US, Nepal, India, Indonesia]

key = china, value = 150
key = US, value = 50
key = Nepal, value = 5
key = India, value = 100
key = Indonesia, value = 6

LinkedHashMap:

Note: 1. keys are insertion ordered
2. output jis order me rakha hota hai usi order me output aata hai.

Package: import java.util.LinkedHashMap;

create:

```
LinkedHashMap<key, value> lhm = new LinkedHashMap<>();
```

put, get, remove, size, isEmpty, keySet — Same properties as HashMap; Same Time Complexity

Program: LinkedHashMap

```
import java.util.*;
public class Classroom{
public static void main(String args[]){
//create
LinkedHashMap<String, Integer> lhm = new
        LinkedHashMap<>();
lhm.put("India", 100);
lhm.put("china", 150);
lhm.put("US", 50);
System.out.println(lhm);
}
```

output:

{India = 100, china = 150, US = 50}

TreeMap:

Note: 1. keys are alphabet (A-Z) ascending sorted order.
2. put, get, remove are O(log n)
3. TreeMap internal implementation Red Black Tree

```
TreeMap<key, value> tm = new TreeMap<>();
```

Program:

```
import java.util.*;
public class Classroom{
public static void main(String args[]){
TreeMap<String, Integer> tm = new TreeMap<>();
tm.put("India", 100);
tm.put("China", 150);
tm.put("US", 50);
System.out.println(tm);
}
}
```

output: {china = 150, India = 100, US = 50}

alphabet order sorted ↳ O(log n)

Qus: Majority Element

Given an integer array of size n, find all elements that appear more than [n/3] times.

nums[] = {1, 3, 2, 5, 1, 3, 1, 5, 1};
output: 1

nums[] = {1, 2};
output = 1, 2

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

→ 4/3 = (1) output

Program:

```
import java.util.*;
public class Classroom {
public static void main (String args[]) {
int arr[] = { 1, 3, 2, 5, 1, 3, 1, 5, 1};
HashMap<Integer, Integer> map = new HashMap<>();
for(int i=0; i<arr.length; i++) {
if (map.containskey(arr[i])) {
map.put(arr[i], map.get(arr[i])+1);
}else {
map.put(arr[i], 1);
}
}
```

or

```
map.put(arr[i], map.getOrDefault(arr[i], 0)+1);
}
Set<Integer> keySet = map.keySet();
for(Integer key : keySet) {
if (map.get(key) > arr.length/3) {
System.out.println(key);
}}}
```

Ans Valid Anagram

Given two strings s and S, return true if t is an anagram of s, and false otherwise.

S = "race" t = "care" = True

S = "heart" t = "earth" = true

S = "tulip" t = "lipid" = false

Sol = S = "Knee" t = "keem"

Program:

```
import java.util.*;
public class Classroom {
public static boolean isAnangram(String S, String t){
HashMap<Character, Integer> map = new HashMap<>();
for(int i=0; i<S.length(); i++){
char ch = S.charAt(i);
map.put(ch, map.getOrDefault(ch, 0)+1);
}
for(int i=0; i<t.length(); i++){
char ch = t.charAt(i);
if(map.get(ch) != null){
if(map.get(ch)==1){
map.remove(ch);
}else{
map.put(ch, map.get(ch)-1);
}
}else{
return false;
}
}
return map.isEmpty();
}
psvm(){
String S = "race";
String t = "care";
sop(isAnangram(S,t));
}}
```

map
k - 1
n - 1
e - 2

O(n)
Output
true

C = Dennis Ritchie
Java = James Gosling
C++ = Bjarne Stroustrup
JS = Brendan Eich
PHP = Rasmus Lerdorf
PYTHON = Guido van Rossum

Topic : Hashset

Note:
- Hashset is collections of unique value store.
- Unordered print
- NULL is allowed
- Hashset is implemented using a HashMap in java.

Hashset Operations:

Package: import java.util.HashSet

Create Hashset:

```
HashSet<Integer> hs = new HashSet<>();
add(key) -> O(1)
contains(key) -> O(1)
remove(key) -> O(1)
```

HashSet create Program:

```
import java.util.*;
public class Classroom {
public static void main(String args[]) {
HashSet<Integer> set = new HashSet<>();
Set.add(1);
Set.add(2);
Set.add(4);
Set.add(2);
Set.add(1);
System.out.println(set);
set.remove(2);
if (set.contains(2)) {
System.out.println("set contains 2");
SOP(set.size());
}
}
}
```

output: [1, 2, 4]

* Iteration on HashSet :

a) using Iterators

Iterator ek aisa pointer hota hai jo kisi collection par ya kisi bhi datastructure pe loop chalane ki liye use kiya jata hai.

Code:

```
Iterator it = set.iterator()
while (it.hasNext()) {
print(it.next())
}
```

Program:

```
import java.util.*;
public class Classroom {
public static void main(String args[]) {
HashSet<String> cities = new HashSet<>();
cities.add("Delhi");
cities.add("Mumbai");
cities.add("Noida");
cities.add("Bengaluru");
Iterator it = cities.iterator();
while (it.hasNext()) {
System.out.println(it.next());
}
}
}
```

or

```
for(String city : cities)
{
SOP(city);
}
```

output:
Delhi
Bengaluru
Noida
Mumbai

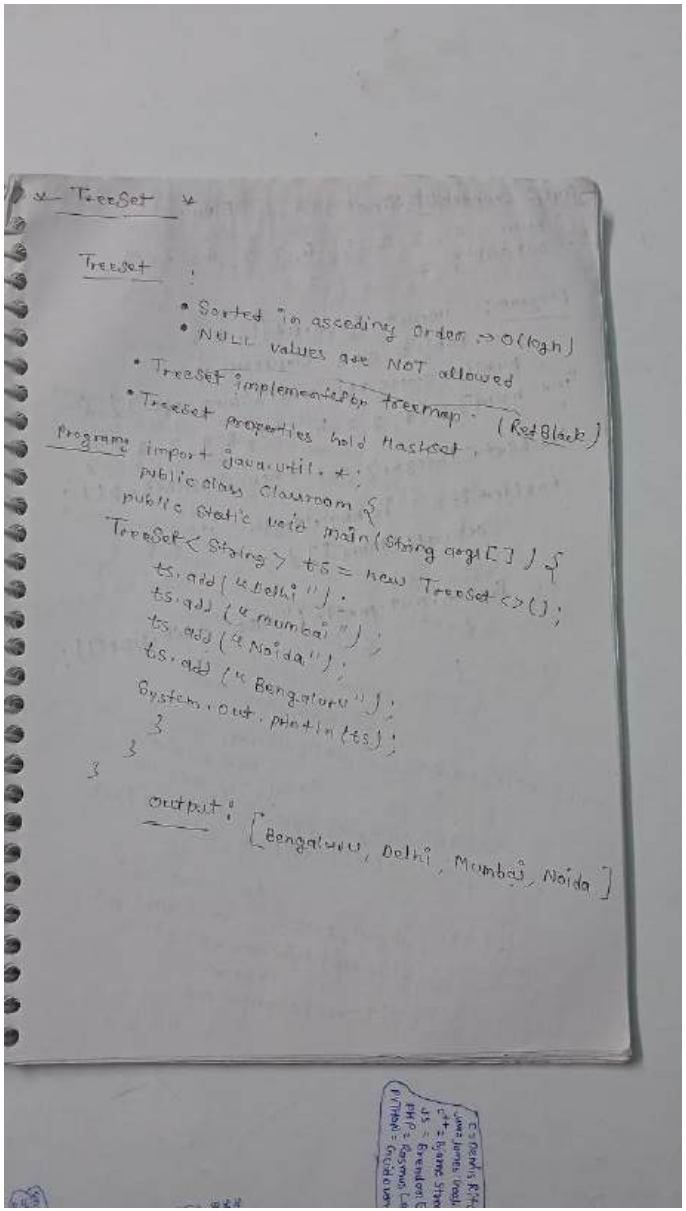
* Linked Hashset *

Note:
- Properties are hold Hashset
- Linked HashSet Ordered using DLL
- Unique value, Serialwise Print Output

Program:

```
import java.util.*;
public class Classroom {
public static void main(String args[]) {
LinkedHashSet<String> lhs = new LinkedHashSet<>();
lhs.add("Delhi");
lhs.add("Mumbai");
lhs.add("Noida");
lhs.add("Bengaluru");
System.out.println(lhs);
lhs.remove("Delhi");
SOP(lhs);
}
```

output:
[Delhi, Mumbai, Noida, Bengaluru]
[Mumbai, Noida, Bengaluru]



TreeSet

TreeSet :

- Sorted in ascending order → O(log n)
- NULL values are NOT allowed
- TreeSet implemented by treemap. (Red Black)
- TreeSet properties hold HashSet.

Program:

```
import java.util.*;
public class Classroom {
    public static void main(String args[]) {
        TreeSet<String> ts = new TreeSet<>();
        ts.add("Delhi");
        ts.add("Mumbai");
        ts.add("Noida");
        ts.add("Bengaluru");
        System.out.println(ts);
    }
}
```

Output: [Bengaluru, Delhi, Mumbai, Noida]

Qus: Count Distinct or Unique Elements

num = 4, 3, 2, 5, 6, 7, 3, 4, 2, 1

output: 7

Program:

```
import java.util.*;
public class Classroom {
    public static void main(String args[]) {
int num[] = {4, 3, 2, 5, 6, 7, 3, 4, 2, 1};
HashSet<Integer> set = new HashSet<>();
for(int i=0; i<num.length; i++) {
    set.add(num[i]);
    }
    System.out.println("Ans=" +set.size());
    }
}
```

Qus 2. Union & Intersection of 2 arrays

arr1 = {7, 3, 9}

arr2 = {6, 3, 9, 2, 9, 4}

union
output = 6 {7, 3, 9, 6, 2, 4}

intersection
output = 2 (3, 9)

Program:

```
import java.util.*;
Public class Classroom {
public static void main (String args[]) {
int arr1 = {7, 3, 9};
int arr2 = {6, 3, 9, 2, 9, 4};
HashSet<Integer> Set = new HashSet<>();
// union
for(int i=0; i<arr1.length; i++) {
    Set.add(arr1[i]);
}
for(int i=0; i<arr2.length; i++) {
    Set.add(arr2[i]);
}
System.out.println("union = " +Set.size());
//intersection
set.clear();
for(int i=0; i<arr1.length; i++) {
    Set.add(arr1[i]);
}
int count=0;
for(int i=0; i<arr2.length; i++) {
    if(Set.contains(arr2[i])) {
        count++;
        Set.remove(arr2[i]);
    }
}
}
```

Find Itinerary from Tickets

"Chennai" → "Bengaluru"
"Mumbai" → "Delhi"
"Goa" → "Chennai"
"Delhi" → "Goa"

Output: "Mumbai" → "Delhi" → "Goa" → "Chennai" → "Bengaluru"

Sol:


```
Program: import java.util.*;
public class Classman {
    public static String getStart(HashMap<String, String>
                                  tickets) {
        HashMap<String, String> revMap = new HashMap<>();
        for(String key : tickets.keySet()) {
            revMap.put(tickets.get(key), key);
        }
        for(String key : tickets.keySet()) {
            if(!revMap.containsKey(key)) {
                return key;
            }
        }
        return null;
    }
    public static void main(String args[]) {
        HashMap<String, String> tickets = new HashMap<>();
        tickets.put("Chennai", "Bengaluru");
        tickets.put("Mumbai", "Delhi");
        tickets.put("Goa", "Chennai");
        tickets.put("Delhi", "Goa");
        String start = getStart(tickets);
        System.out.print(start);
        for(String key : tickets.keySet()) {
            System.out.print(" -> " + tickets.get(start));
            start = tickets.get(start);
        }
    }
}
```

Largest Subarray with 0 sum

arr = {15, -2, 2, -8, 1, 7, 10}

Output: 5

-2+2-8+1+7 = 0

arr = {3, 4, 5}

Output = 0

Program:

```
import java.util.*;
public class Classman {
    public static void main(String args[]) {
        int arr[] = {15, -2, 2, -8, 1, 7, 10, 23};
        HashMap<Integer, Integer> map = new HashMap<>();
        // (sum, idx)
        int sum = 0;
        int len = 0;
        for(int i=0; i<arr.length; i++) {
            sum += arr[i];
            if(map.containsKey(sum)) {
                len = Math.max(len, i-map.get(sum));
            } else {
                map.put(sum, i);
            }
        }
        System.out.println("Largest subarray with sum 0 => " + len);
    }
}
```

Ques: Subarray sum equal to K

arr = {1, 2, 3} K=3 return number of such subarray

i=0 i=1 i=2

Output: 2 (1,2) (3)

Program: import java.util.*;

public class Classroom {

public static void main (String args[]) {

int arr[] = {10, 2, -2, -20, 10};

int K = -10;

HashMap<Integer, Integer> map = new HashMap();

// Sum, count

map.put(0,1);

int sum = 0;

int ans = 0;

for(int j=0; j<arr.length; j++) { // i to j

sum += arr[j];

if(map.containsKey(sum-K)) {

ans += map.get(sum-K);

}

map.put(sum, map.getOrDefault(sum, 0)+1);

}

System.out.println(ans);

}

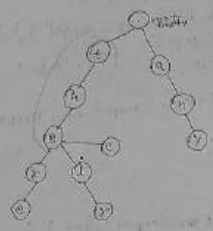
}

Output: 3	O(n)

Topic: Trie Data Structure (Retrieval)

Q. What is a Trie? → K-ary tree

Words[] = "the", "a", "there", "their", "any", "thee"



Class TrieNode {

TrieNode[26] children;

boolean eow;

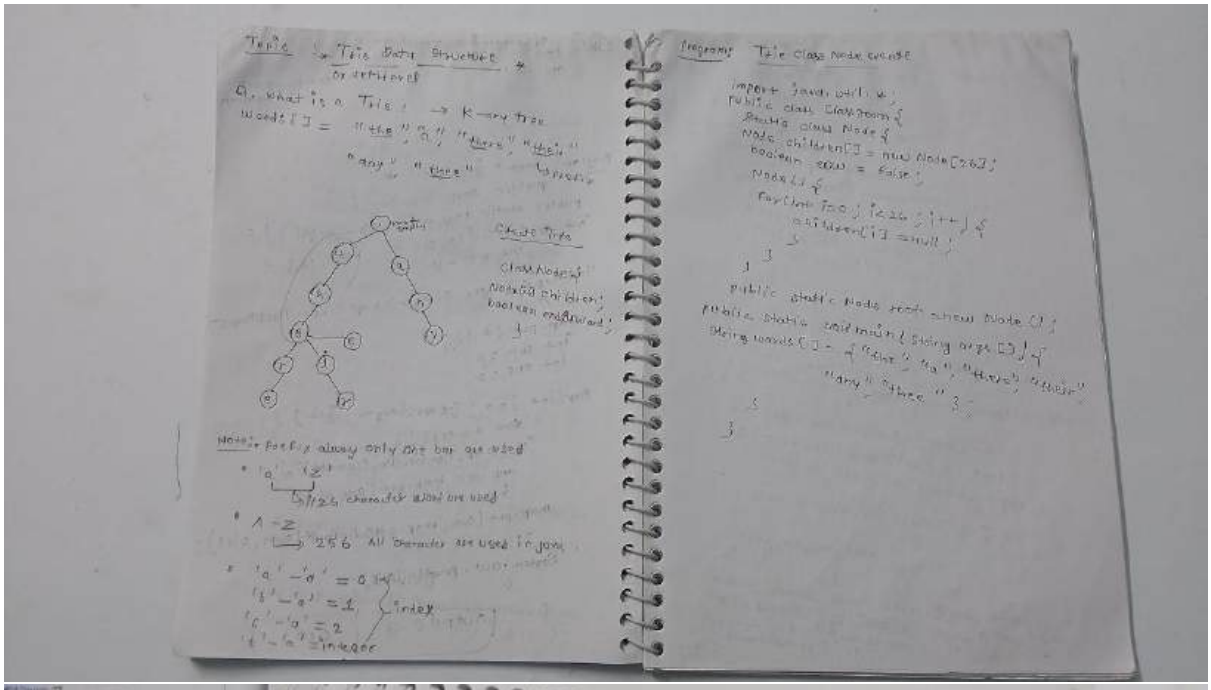
}

Note: Prefix always only one bar use hoti

- 'a' - 'z' → 26 character alone use hoti
- A-Z → 256 All character use hoti in java
- 'a' - 'a' = 0
  'b' - 'a' = 1
  'c' - 'a' = 2 } index

Program: Trie class Node create

```
import java.util.*;
public class Classroom {
  static class Node {
    Node children[] = new Node[26];
    boolean eow = false;
    Node() {
      for(int i=0; i<26; i++) {
        children[i] = null;
      }
    }
  }
  public static Node root = new Node();
  public static void main(String args[]) {
    String words[] = {"the", "a", "there", "their",
                      "any", "thee"};
  }
}
```



Trie → Tree Data Structure
(or retrieval)

Q. what is a Trie? → K-ary Tree

words[] = "the", "a", "there", "their", "any", "thee"

Create Trie

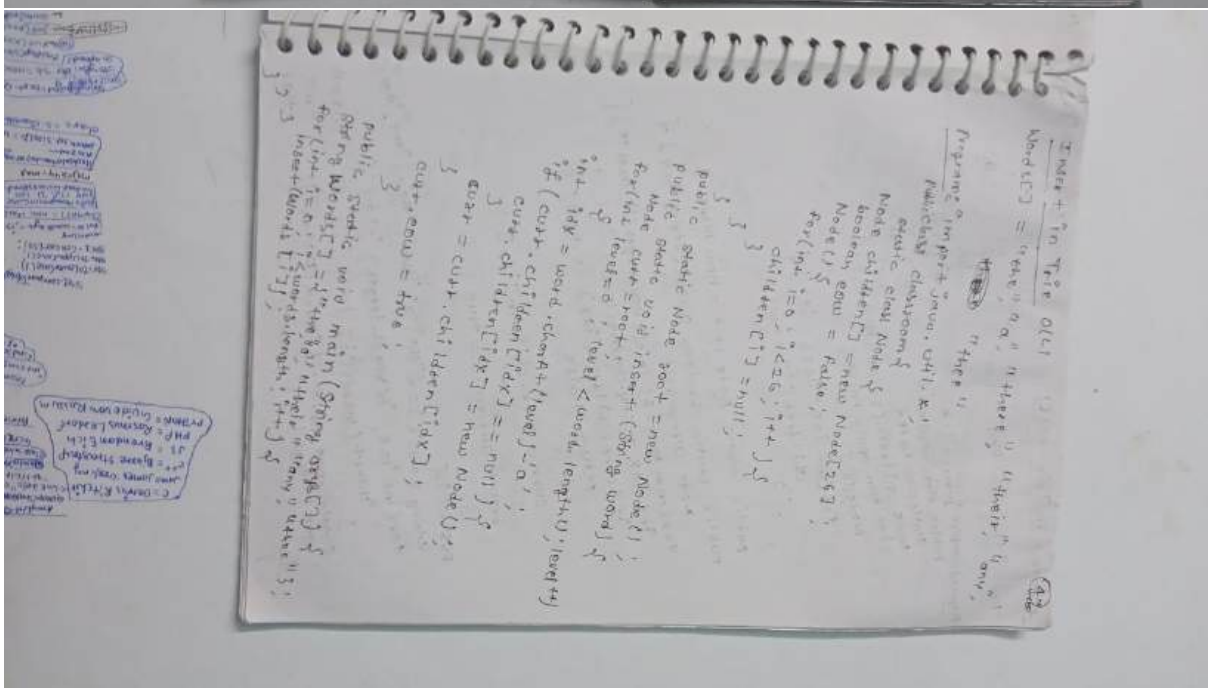
Class Node {
Node[] children;
boolean endOfWord;
}

Note: Prefix always only Dict but are used

- 'a' - 'z' → 26 characters alone are used
- A - Z → 256 All character are used in java
- 'a' - 'a' = 0 index
  'b' - 'a' = 1
  'c' - 'a' = 2
  'z' - 'a' = 25

Program: Trie class Node create

```
import java.util.*;
public class ClassRoom {
  Static class Node {
    Node children[] = new Node[26];
    boolean eow = false;
    Node() {
      for(int i=0; i<26; i++) {
        children[i] = null;
      }
    }
  }
  public static Node root = new Node();
  public static void main(String args[]) {
    String words[] = {"the", "a", "there", "their", "any", "thee"};
  }
}
```

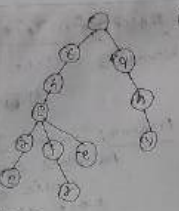


Time + in Trie O(L)

Word[] = "the", "a", "there", "their", "any", "thee"

Program: import java.util.*;
public class ClassRoom {
  static class Node {
    Node children[] = new Node[26];
    boolean eow = false;
    Node() {
      for(int i=0; i<26; i++) {
        children[i] = null;
      }
    }
  }
  public static Node root = new Node();
  public static void insert(String word) {
    Node curr = root;
    for(int level=0; level<word.length(); level++) {
      int idx = word.charAt(level) - 'a';
      if(curr.children[idx] == null) {
        curr.children[idx] = new Node();
      }
      curr = curr.children[idx];
    }
    curr.eow = true;
  }
  public static void main(String args[]) {
    String words[] = {"the", "a", "there", "their", "any", "thee"};
    for(int i=0; i<words.length; i++) {
      insert(words[i]);
    }
  }
}

Search in Trie — O(L)



Key = "thee" Key = "thor"

Program:

```
import java.util.*;
public class Classroom {
    static class Node {
        Node children[] = new Node[26];
        boolean eow = false;
        Node() {
            for(int i=0; i<26; i++) {
                children[i] = null;
            }
        }
    }
    public static Node root = new Node();
    public static boolean search(String key) {
        Node curr = root;
        for(int level=0; level<key.length(); level++) {
            int idx = key.charAt(level) - 'a';
            if(curr.children[idx] == null) {
                return false;
            }
            curr = curr.children[idx];
        }
        return curr.eow == true;
    }
    psvm() {
        String words[] = {"the", "a", "there", "their", "any", "thor"};
        for(int i=0; i<words.length; i++) {
            insert(words[i]);
        }
        System.out.println(search("thee"));
        son("thor");
    }
}
```

Output: true, thor

Word Break Problem : O(L)

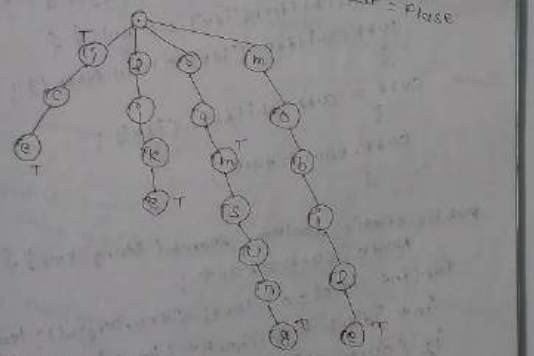
Given an input string and a dictionary of words, find out if the input string can be broken into a space-separated sequence of dictionary words.

words[] = {"i", like, sam, samsung, mobile, ice}

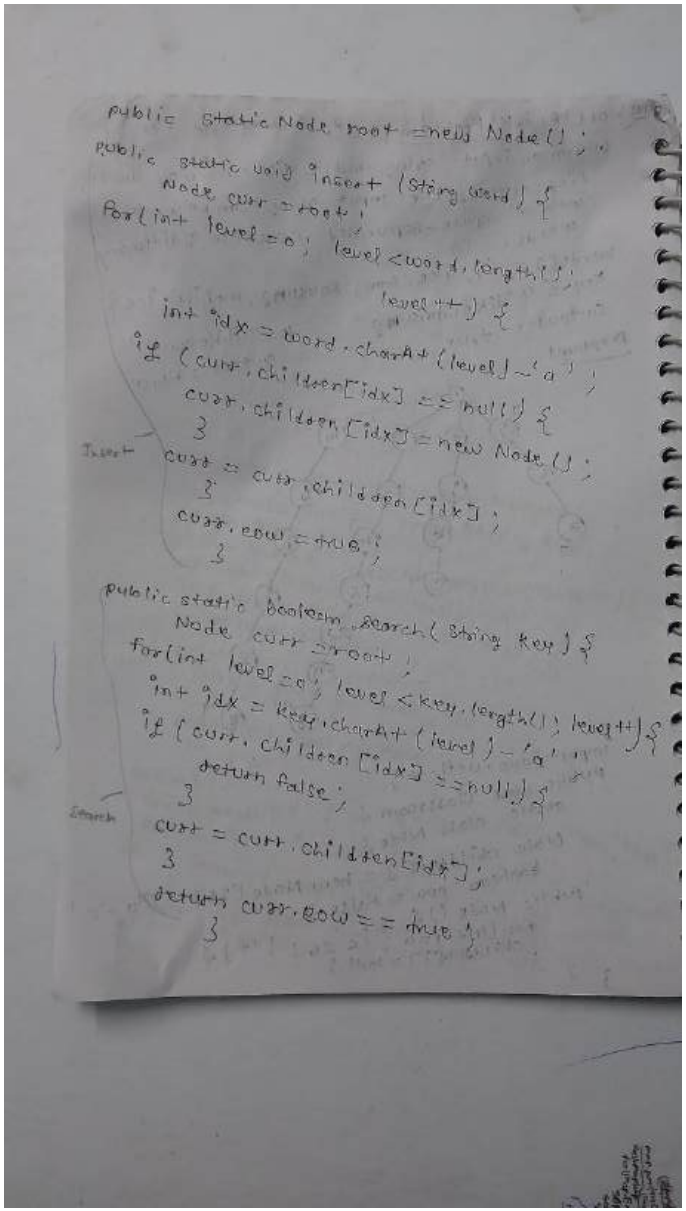
Key = "ilikesamsung" Key = "ilikesung"

Output = true Output = false

Program:



```
import java.util.*;
public class Classroom {
    static class Node {
        Node children[] = new Node[26]; //[a-z]
        boolean eow = false;
        public Node() {
            for(int i=0; i<26; i++) {
                children[i] = null;
            }
        }
    }
```



```
public static Node root = new Node();

public static void insert(String word) {
    Node curr = root;
    for(int level = 0; level < word.length(); level++) {
        int idx = word.charAt(level) - 'a';
        if(curr.children[idx] == null) {
            curr.children[idx] = new Node();
        }
        curr = curr.children[idx];
    }
    curr.eow = true;
}

public static boolean search(String key) {
    Node curr = root;
    for(int level = 0; level < key.length(); level++) {
        int idx = key.charAt(level) - 'a';
        if(curr.children[idx] == null) {
            return false;
        }
        curr = curr.children[idx];
    }
    return curr.eow == true;
}
```

Insert

Search

```
public static boolean wordBreak(String key) {
    if (key.length() == 0) {
        return true;
    }
    for (int i=1; i<=key.length(); i++) {
        if (search(key.substring(0,i)) && wordBreak
                (key.substring(i))) {
            return true;
        }
    }
    return false;
}
public static void main(String args[]) {
    String arr[] = {"i", "like", "sam", "samsung",
                    "mobile", "ice"};
    for (int i=0; i<arr.length; i++) {
        insert(arr[i]);
    }
    String key = "ilikesamsung";
    System.out.println(wordBreak(key));
}
```

Output: true

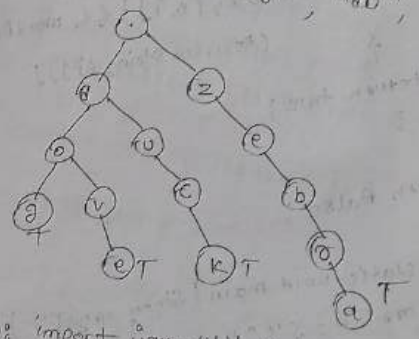
Qus: Prefix Problem

Find shortest unique prefix for every word in a given list. Assume no word is prefix of another.

arr[] = {"zebra", "dog", "duck", "dove"}

output: {"z", "dog", "du", "dov"}

Sol:



Program:

```
import java.util.*;
Public class Classroom {
static class Node {
    Node[] children = new Node[26];
    boolean eow = false;
    int freq;
    public Node() {
    for(int i=0; i<children.length; i++)
    {
    children[i] = null;
    }
    freq = 1;
}
```

```
public static Node root = new Node();
public static void insert(String word) {
  Node curr = root;
for(int i=0; i<word.length(); i++) {
int idx = word.charAt(i) - 'a';
if(curr.children[idx] == null) {
  curr.children[idx] = new Node();
} else {
    curr.children[idx].freq++;
}
curr = curr.children[idx];
}
curr.eow = true;
}
```

```
public static void findPrefix(Node
        root, String ans) {
    if(root == null) {
        return;
    }
    if(root.freq == 1) {
    sop(ans);
    return;
    }
    for(int i=0; i<root.children.length; i++)
    {
    if(root.children[i] != null) {
    findPrefix(root.children[i], ans+(char)(i+'a'));
}
}
```

```
psvm ( ) {
String arr[] = { "zebra", "dog", "duck",
                 "dove" } ;
for(int i=0 ; i<arr.length ; i++) {
    insert (arr[i]) ;
    }
    root.freq = -1 ;
    findPrefix (root, "") ;
    }
}
```

output: dog
dov
dv
z

Qus: S
Create
for
Return
String
Othe
words[]

Prefix

Prefix

Solⁿ

Qus: startsWith Problem

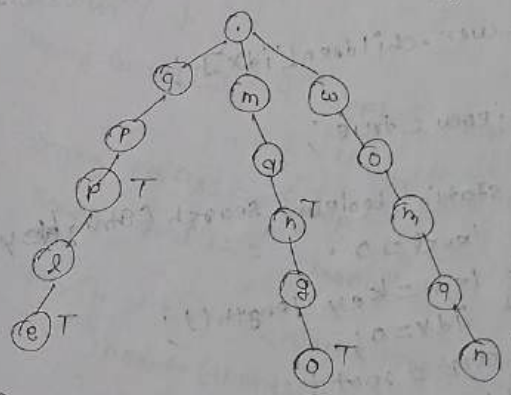
Create a function, boolean startsWith(String Prefix) for a tree,
Returns true if there is a previously inserted String word that has the prefix, and false otherwise true.

words[] = { "apple", "app", "mango", "man", "woman" }

Prefix = "app" output: true

Prefix = "moon" output: false

Sol=




```
import java.util.*;
public class Classroom {
    static class Node{
    Node[] children = new Node[26];
    boolean eow;
    public Node(){
        for(int i=0; i<26; i++){
        children[i] = null;
    }
}
```

```
public static Node root = new Node();
public static void insert(String word){
    int level = 0;
    int len = word.length();
    int idx = 0;
    Node curr = root;
    for(; level < len; level++){
        idx = word.charAt(level) - 'a';
        if(curr.children[idx] == null){
            curr.children[idx] = new Node();
        }
        curr = curr.children[idx];
    }
    curr.eow = true;
}
public static boolean search(String key){
    int level = 0;
    int len = key.length();
    int idx = 0;
    Node curr = root;
    for(; level < len; level++){
        idx = key.charAt(level) - 'a';
        if(curr.children[idx] == null){
            return false;
        }
        curr = curr.children[idx];
    }
    return curr.eow == true;
}
```

```
public static boolean startsWith(String prefix){
    Node curr = root;
    for(int i=0; i<prefix.length(); i++){
        int idx = prefix.charAt(i) - 'a';
        if(curr.children[idx] == null){
            return false;
        }
        curr = curr.children[idx];
    }
    return true;
}
psvm(){
    String words[] = {"apple", "app", "mango",
                      "man", "woman"};
    String prefix1 = "app";
    String prefix2 = "moon";
    for(int i=0; i<words.length; i++){
        insert(words[i]);
    }
    sop(startsWith(prefix1));
    sop(startsWith(prefix2));
}
```

Output: true
false

Q.15 Count Unique Substrings

Given a string of length n of lowercase alphabet characters, we need to count total number of distinct substrings of this string

Str = "abab"

output : 10

Solution :

all substrings

a	ab	aba	abab	ababa
b	ba	bab	baba	
a	ab	abb		
b	bq			
a				

Program :

```
import java.util.*;
public class Classroom {
    static class Node {
        Node[] children = new Node[26];
        boolean eow;
        public Node() {
            for (int i=0; i<26; i++) {
                children[i] = null;
            }
        }
    }
    public static Node root = new Node();
```

```
public static int countNodes(Node root) {
    if (root == null) {
        return 0;
    }
    int count = 0;
    for (int i=0; i<26; i++) {
        if (root.children[i] != null) {
            count += countNodes(root.children[i]);
        }
    }
    return count+1;
}
psum() {
    String str = "ababa";
    for (int i=0; i<str.length(); i++) {
        String suffix = str.substring(i);
        insert(suffix);
    }
    sop(countNodes(root));
}
```

output : 10

Topic : GRAPHS

GRAPH : Graphic "is a collection or network of nodes.

Bidirectional structure

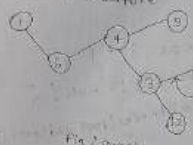
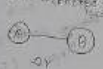


Fig : Graph

Edges → 2 Type → (1) Uni-Directional
(2) Un-Directed

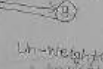
(1) Uni-Directional

(2) Un-Directed



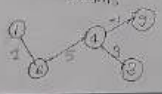
OR

Bi-Directional

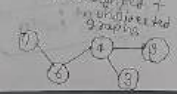


Types

Weighted + UnDirected graph



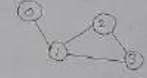
Un-Weighted + UnDirected graph



String a Graph Represent structure

1. Adjacency List
2. Adjacency Matrix
3. Edge List
4. 2D Matrix (Implicit Graph)

1. Adjacency List — TC = O(V+E) → O(E)



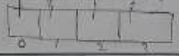
Vertex = 4
Edges = 4

Undirected / Directed Graphs

Information

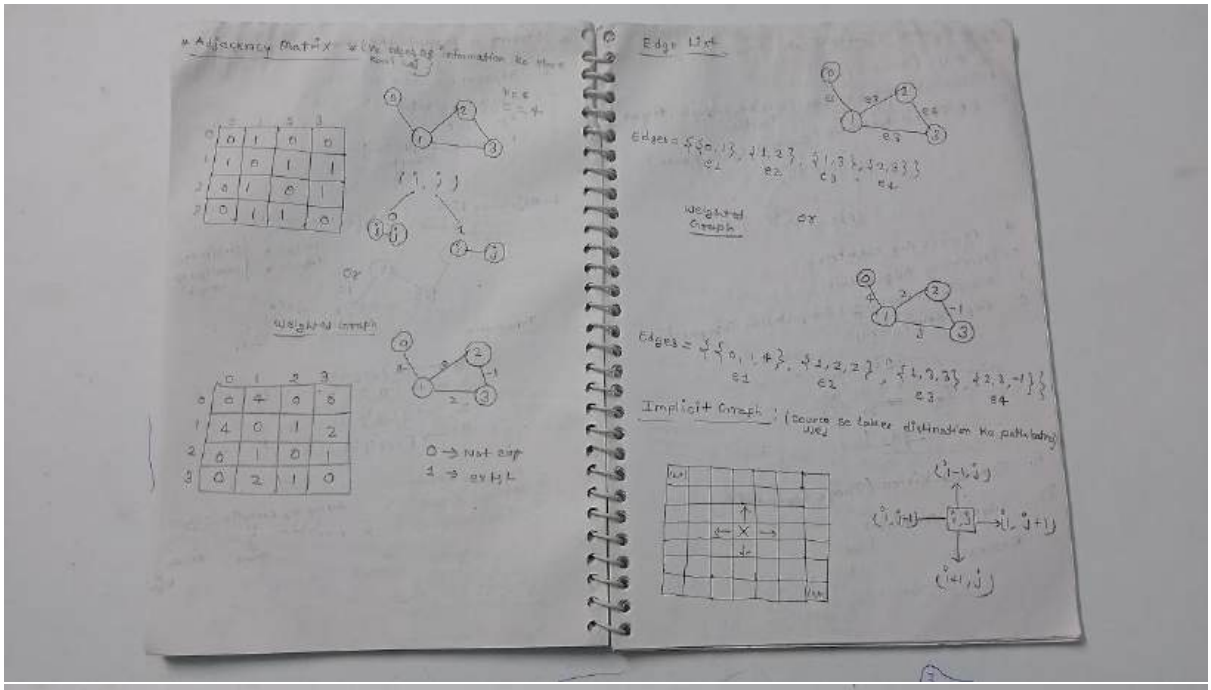
0 → [1] list[0]
1 → [0, 2, 3] list[1]
2 → [1, 3] list[2]
3 → [1, 2] list[3]

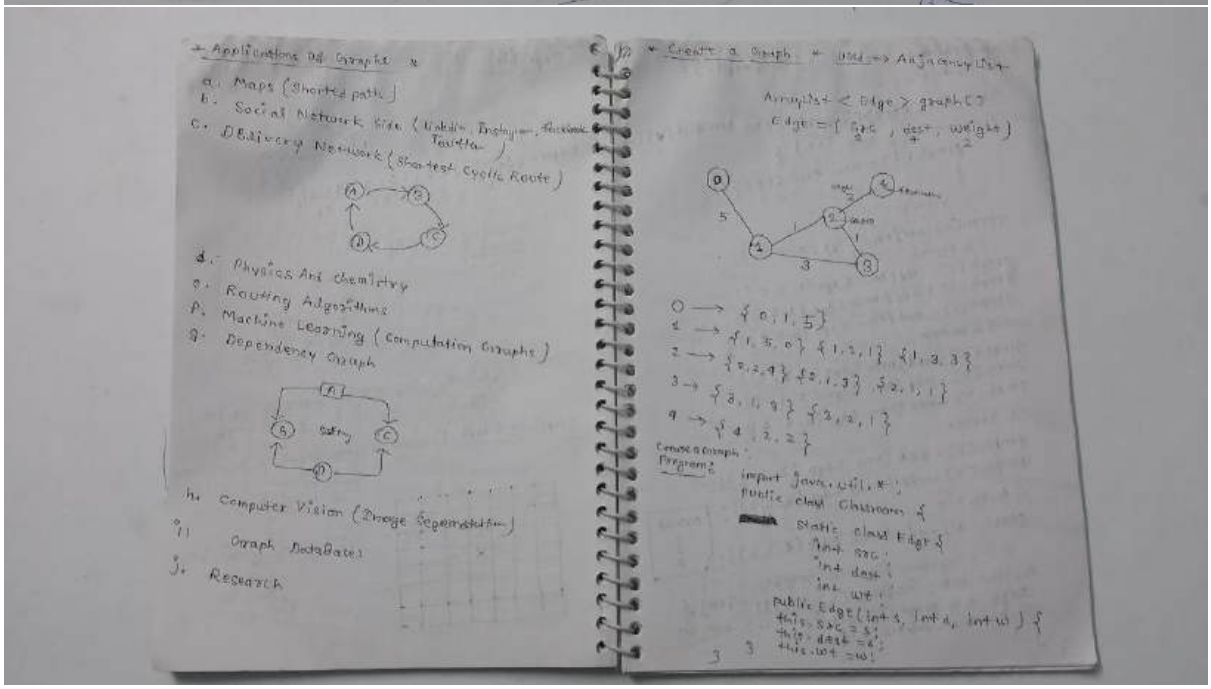
ArrayList<ArrayList>
OR
Array<ArrayList>
OR
Hashmap<int, list>
Vertex (key) → (value)



Node

Array of ArrayList<Edge>


* Adjacency Matrix *
Edge List
Edges = {{0,1}, {1,2}, {1,3}, {2,3}}
e1 e2 e3 e4
Weighted Graph
or
Edges = {{0,1,4}, {1,2,2}, {1,3,3}, {2,3,-1}}
e1 e2 e3 e4
Weighted Graph
0 → Not exp
1 → exist
Implicit Graph : (Source se leker destination ka pata lagna)


* Applications of Graphs *
a. Maps (Shortest path)
b. Social Network Site (Linkedin, Instagram, Facebook, Twitter)
c. Delivery Network (Shortest Cyclic Route)
d. Physics And Chemistry
e. Routing Algorithms
f. Machine Learning (Computation Graphs)
g. Dependency Graph
Safety
h. Computer Vision (Image Segmentation)
i. Graph Database
j. Research
* Create a Graph * Used → Adjacency List
ArrayList < Edge > graph[]
Edge = { src, dest, weight }
0 → {0, 1, 5}
1 → {1, 3, 0} {1, 2, 1} {1, 3, 3}
2 → {2, 1, 4} {2, 1, 3} {2, 3, 1}
3 → {3, 1, 3} {3, 2, 1}
4 → {4, 3, 2}
Create a Graph Program :
import java.util.*;
public class Classroom {
static class Edge {
int src;
int dest;
int wt;
public Edge(int s, int d, int w) {
this.src = s;
this.dest = d;
this.wt = w;
}

```
public static void main (String args[]) {
    int V = 5;
ArrayList<Edge>[] graph = new ArrayList[V];
for (int i=0; i<V; i++) {
    graph[i] = new ArrayList<>();
}
// 0-vertex
graph[0].add(new Edge(0, 1, 5));
// 1 vertex
graph[1].add(new Edge(1, 0, 5));
graph[1].add(new Edge(1, 2, 1));
graph[1].add(new Edge(1, 3, 3))
// 2 vertex
graph[2].add(new Edge(2, 1, 1));
graph[2].add(new Edge(2, 3, 1));
graph[2].add(new Edge(2, 4, 4));
// 3 vertex
graph[3].add(new Edge(3, 1, 3));
graph[3].add(new Edge(3, 2, 1));
// 4 vertex
graph[4].add(new Edge(4, 2, 2));
// 2's neighbors find-out neighbour
for(int i=0; i<graph[2].size(); i++) {
Edge e = graph[2].get(i); //src, dest, wt
System.out.println(e.dest);
} } }
```

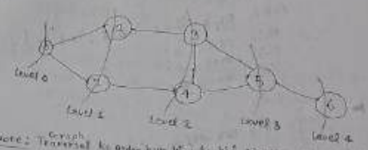
output
1
3
4

Graph Traversals → 2 Type: BFS, DFS

1. Breadth First Search (BFS)

Time complexity: O(V+E)

Ex

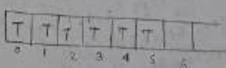


Note: Graph Traversal to order hum K(ii) ko bhi starting point le skate hai.

- Go to immediate neighbors first.
- BFS are Queue Implementation used. (FIFO)
- Graph are cycle → visited (visited starting point are false or empty); array visited used
- Visited each node is visited one. F X, T ✓

queue

q.remove() ; if(!visit[curr]) visit[curr] = print[curr]



Print
0 1 2 3 4 5

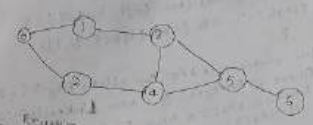
// create BFS Program:

```
import java.util.*;
public class Classroom {
    static class Edge {
        int src;
        int dest;
        int wt;
    public Edge(int s, int d, int w) {
        this.src = s;
        this.dest = d;
        this.wt = w;
    }
    }
static void createGraph(ArrayList<Edge> graph[])
    {
for (int i=0; i<graph.length; i++) {
    graph[i] = new ArrayList<>();
    }
graph[0].add(new Edge(0, 1, 1));
graph[0].add(new Edge(0, 2, 1));
graph[1].add(new Edge(1, 0, 1));
graph[1].add(new Edge(1, 3, 1));
graph[2].add(new Edge(2, 0, 1));
graph[2].add(new Edge(2, 4, 1));
graph[3].add(new Edge(3, 1, 1));
graph[3].add(new Edge(3, 4, 1));
graph[3].add(new Edge(3, 5, 1));
graph[4].add(new Edge(4, 2, 1));
graph[4].add(new Edge(4, 3, 1));
graph[4].add(new Edge(4, 5, 1));
graph[5].add(new Edge(5, 3, 1));
graph[5].add(new Edge(5, 4, 1));
graph[5].add(new Edge(5, 6, 1));
    }
public static void bfs(ArrayList<Edge>[] graph) {
  Queue<Integer> q = new LinkedList<>();
  boolean vis[] = new boolean[graph.length];
    q.add(0);          // source=0
  while (!q.isEmpty()) {
        int curr = q.remove();
      if (!vis[curr]) { // visit curr
      System.out.print(curr + " ");
        vis[curr] = true;
    for(int i=0; i<graph[curr].size(); i++)
     {
  Edge e = graph[curr].get(i);
        q.add(e.dest);
   } } } }
 psvm() {
 int V=7;
 ArrayList<Edge> graph[] = new ArrayList[V];
     createGraph(graph);
      bfs(graph);
  }
 }
```

Output
0 1 2 3 4 5 6

DFS (Depth First Search) — O(V+E)

Note:
- Keep going to the 1st neighbour
- Using Recursion - DFS



Recursion

```
cur -> 0
for(int i=0 to n)
  if(!vis[cur])
    dfs(neighbour)
}
```

Circuit DFS Program:

```
1. import java.util.*;
2. public class Diagram {
3.   static class Edge {
4.     int src;
5.     int dest;
6.     int wt;
7.     public Edge(int s, int d, int w) {
8.       this.src = s;
9.       this.dest = d;
10.      this.wt = w;
11.    }
12.  }
```

```
13. public static void createGraph(ArrayList<Edge> graph[]) {
14.   for(int i=0; i<graph.length; i++) {
15.     graph[i] = new ArrayList<>();
16.   }
17.   graph[0].add(new Edge(0,1,1));
18.   graph[0].add(new Edge(0,2,1));
19.   graph[1].add(new Edge(1,0,1));
20.   graph[1].add(new Edge(1,3,1));
21.   graph[2].add(new Edge(2,0,1));
22.   graph[2].add(new Edge(2,4,1));
23.   graph[3].add(new Edge(3,1,1));
24.   graph[3].add(new Edge(3,4,1));
25.   graph[3].add(new Edge(3,5,1));
26.   graph[4].add(new Edge(4,2,1));
27.   graph[4].add(new Edge(4,3,1));
28.   graph[4].add(new Edge(4,5,1));
29.   graph[5].add(new Edge(5,3,1));
30.   graph[5].add(new Edge(5,4,1));
31.   graph[5].add(new Edge(5,6,1));
32.   graph[6].add(new Edge(6,5,1));
33. }
34. public static void dfs(ArrayList<Edge> graph[],
        int curr, boolean vis[]) {
35.   System.out.print(curr + " ");
36.   vis[curr] = true;
```

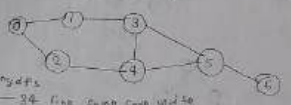
```
37. for(int i=0; i<graph[curr].size(); i++) {
38.   Edge e = graph[curr].get(i);
39.   if(!vis[e.dest]) {
40.     dfs(graph, e.dest, vis);
41.   }
42. }
43. public static void main(String args[]) {
44.   int v = 7;
45.   ArrayList<Edge> graph[] = new ArrayList[v];
46.   createGraph(graph);
47.   dfs(graph, 0, new boolean[v]);
48. }
```

output: 0 1 3 4 2 5 6

Q.1 HasPath ?

For given src & dest, tell if a path exists from src to dest

src = 0, dest = 5



Soln: using dfs

↳ same code as before

```
public static boolean hasPath(ArrayList<Edge>[] graph,
        int src, int dest, boolean vis[]) {
  if(src == dest) {
    return true;
  }
  vis[src] = true;
  for(int i=0; i<graph[src].size(); i++) {
    Edge e = graph[src].get(i);
    //e.dest = neighbour
    if(!vis[e.dest] && hasPath(graph, e.dest,
        dest, vis)) {
      return true;        output = true.
    }
  }
  return false;
}
```

↳ same code as before

```
sop(hasPath(graph, 0, 5, new boolean[v]));
}
```

Connected Components :

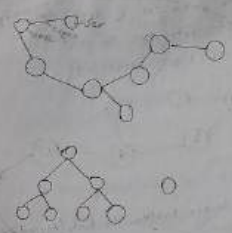


Fig: Single connected components graph

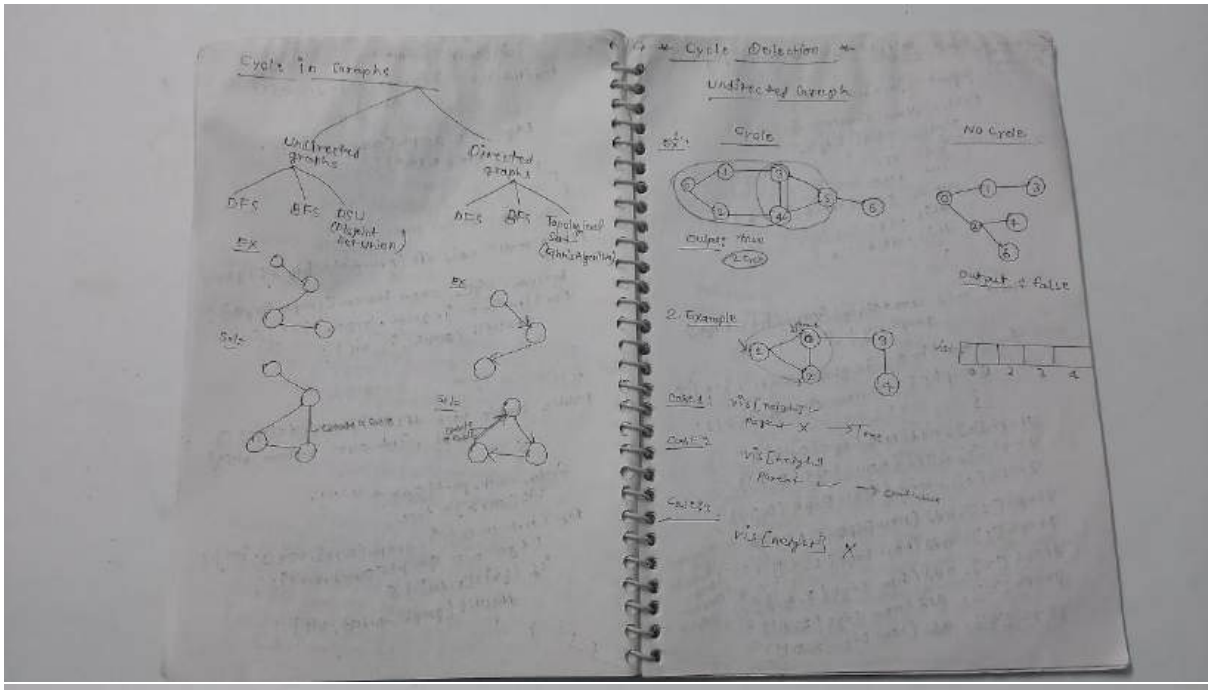
DFS

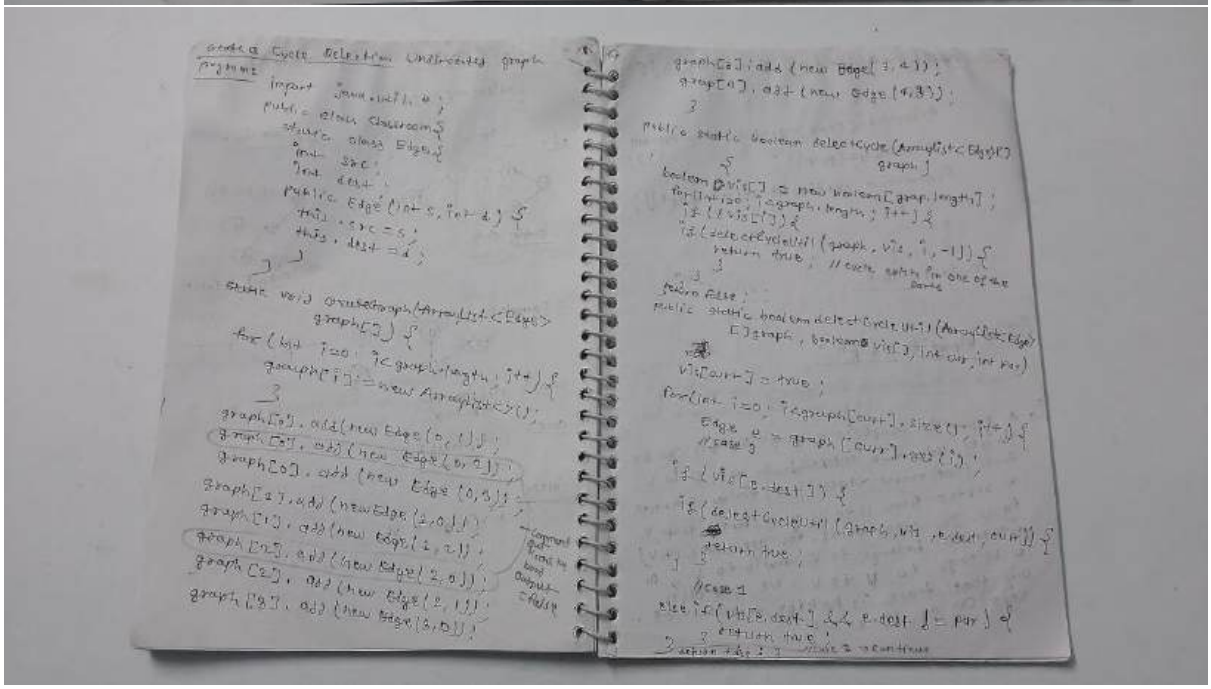
```
visited[] → Boolean
for(int i=0; i<graph.length; i++){
    if(!visited[i])
        dfs(i);
}
dfs visit{
    DFS code
}
```

DFS helper

Create a Connected Components Program :

```
import java.util.*;
public class Classroom {
    static class Edge {
        int src;
        int dest;
        int wt;
        public Edge(int s, int d, int w) {
            this.src = s;
            this.dest = d;
            this.wt = w;
        }
    }
    static void createGraph(ArrayList<Edge> graph[]) {
        for(int i=0; i<graph.length; i++) {
            graph[i] = new ArrayList<>();
        }
        graph[0].add(new Edge(0,1,1));
        graph[0].add(new Edge(0,2,1));
        graph[1].add(new Edge(1,0,1));
        graph[1].add(new Edge(1,3,1));
        graph[2].add(new Edge(2,0,1));
        graph[2].add(new Edge(2,4,1));
        graph[3].add(new Edge(3,1,1));
        graph[3].add(new Edge(3,5,1));
        graph[4].add(new Edge(4,2,1));
        graph[4].add(new Edge(4,3,1));
        graph[4].add(new Edge(4,5,1));
        graph[5].add(new Edge(5,3,1));
        graph[5].add(new Edge(5,4,1));
        graph[6].add(new Edge(6,5,1));
    }
    //create connected graph
    public static void bfs(ArrayList<Edge>[] graph) {
        boolean vis[] = new boolean[graph.length];
        for(int i=0; i<graph.length; i++) {
            if(!vis[i]) {
                bfsUtil(graph, vis);
            }
        }
    }
    //bfs code write          //O(V+E)
    public static void bfsUtil(ArrayList<Edge>[] graph, boolean vis[]) {
        Queue<Integer> q = new LinkedList<>();
        q.add(0); // source =0
        while(!q.isEmpty()) {
            int curr = q.remove();
            if(!vis[curr]) {
                System.out.print(curr+" ");
                vis[curr] = true;
                for(int i=0; i<graph[curr].size(); i++) {
                    Edge e = graph[curr].get(i);
                    q.add(e.dest);
                }
            }
        }
    }
    //dfs code - connected graph
    public static void dfs(ArrayList<Edge>[] graph) {
        boolean vis[] = new boolean[graph.length];
        for(int i=0; i<graph.length; i++) {
            if(!vis[i]) {
                dfsUtil(graph, i, vis);
            }
        }
    }
    //dfs code write          //O(V+E)
    public static void dfsUtil(ArrayList<Edge>[] graph, int curr, boolean vis[]) {
        System.out.print(curr+" ");
        vis[curr] = true;
        for(int i=0; i<graph[curr].size(); i++) {
            Edge e = graph[curr].get(i);
            if(!vis[e.dest]) {
                dfsUtil(graph, e.dest, vis);
            }
        }
    }
```


Cycle in Graphs
Undirected Graphs
Directed Graphs
DFS
BFS
DSU (Disjoint Set Union)
DFS
BFS
Topological sort (Kahn's Algorithm)
Ex
Ex
Ex
* Cycle Detection *
Undirected Graph
Ex:
Cycle
No Cycle
Output: true
Output: false
2. Example
Case 1:
Case 2:
Case 3:


Graph Cycle Detection Undirected graph
Program
public class Edge
public void createGraph(ArrayList<Edge>[] graph)
graph[0].add(new Edge(0,1));
graph[0].add(new Edge(0,2));
graph[0].add(new Edge(0,3));
graph[2].add(new Edge(2,0));
graph[1].add(new Edge(1,2));
graph[2].add(new Edge(2,1));
graph[3].add(new Edge(3,0));
public static boolean detectCycle(ArrayList<Edge>[] graph)
public static boolean detectCycleUtil(ArrayList<Edge>[] graph, boolean vis[], int curr, int par)
// case 1
// case 2
// case 3

```
public static void main (String args[]) {
    int v = 5;
    ArrayList<Edge> graph[] = new ArrayList[v];
    createGraph(graph);
    System.out.println(detectCircle(graph));
}
```

Output : true

Ques :

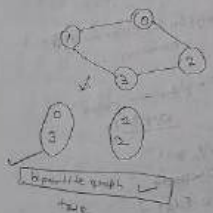
Bipartite Graph :

Bipartite Graph → aisa graph hota hai jisko two part me divide kiya jaa sakta hai.

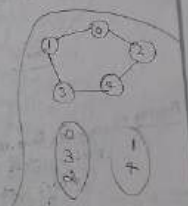
↳ A bipartite graph is a graph whose vertices can be divided into two independent sets, U and V such that every edge (u, v) either connects a vertex from U to V or a vertex from V to U. In other words, for each edge (u, v) either u belongs to U and v to V, or u belongs to V and v to U. We can also say that there is no edge that connects vertices of same set.

set 1 (U) set 2 (V)

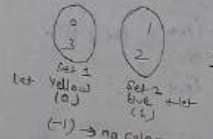
Ques : check if given graph is bipartite or not



bipartite graph ✓ true

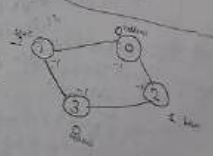


non-bipartite graph false

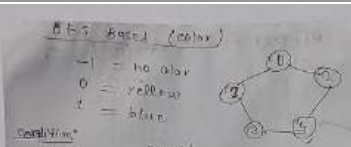


set 1 (yellow) (0) set 2 (blue) (1)

(-1) → no color visited



true



BFS based (color)

-1 = no color
0 = yellow
1 = blue

Conditions

Case 1 → neighbor — color — same → False

Case 2 → neighbor — color — different → Continue

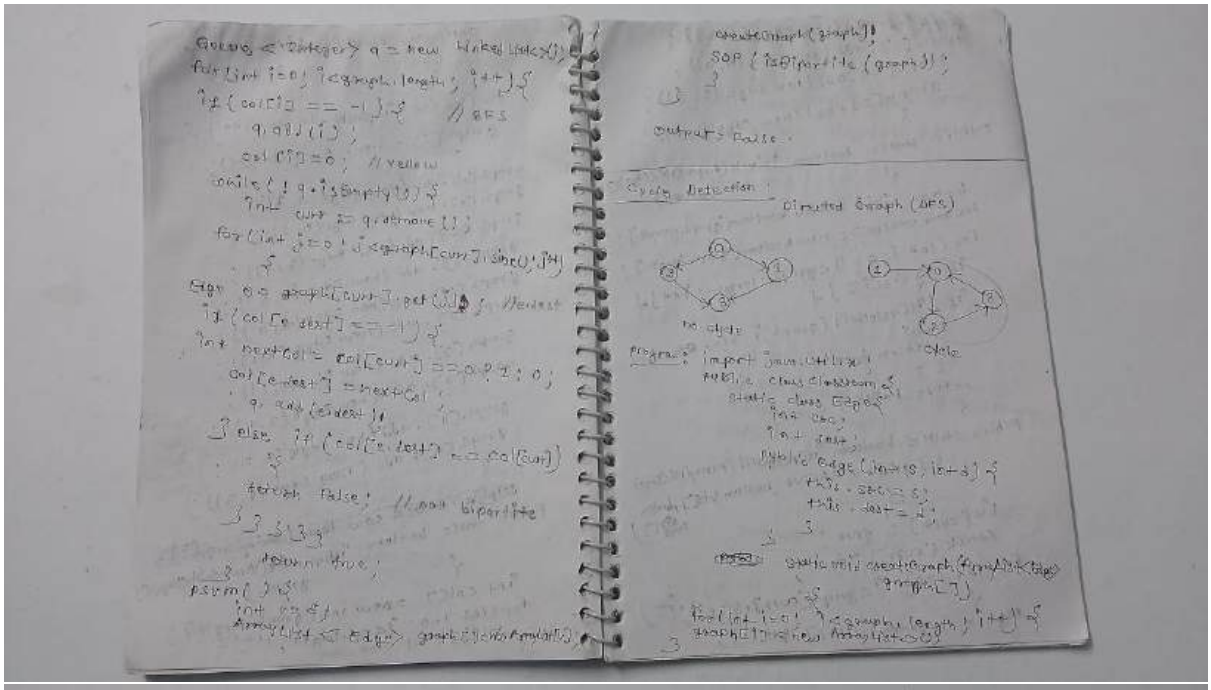
Case 3 → neighbor — color — opposite

Program :

```
import java.util.*;
public class Classroom {
    static class Edge {
        int src;
        int dest;

        public Edge(int s, int d) {
            this.src = s;
            this.dest = d;
        }
    }
```

```
static void createGraph(ArrayList<Edge> graph[]) {
    for(int i=0; i<graph.length; i++) {
        graph[i] = new ArrayList<>();
    }
    graph[0].add(new Edge(0, 2));
    graph[0].add(new Edge(0, 3));
    graph[1].add(new Edge(1, 0));
    graph[1].add(new Edge(1, 3));
    graph[2].add(new Edge(2, 0));
    graph[2].add(new Edge(2, 1));
    graph[3].add(new Edge(3, 1));
    graph[3].add(new Edge(3, 4));
    graph[0].add(new Edge(4, 2));
    graph[4].add(new Edge(4, 3));
}
```

```
public static boolean isBipartite(ArrayList<Edge> graph[]) {
    int col[] = new int[graph.length];
    for(int i=0; i<col.length; i++) {
        col[i] = -1 // no color
    }
```



```
Queue<Integer> q = new LinkedList<>();
for(int i=0; i<graph.length; i++){
if(col[i] == -1){      // BFS
    q.add(i);
    col[i]=0;  // yellow
while(!q.isEmpty()){
    int curr = q.remove();
for(int j=0; j<graph[curr].size(); j++)
}
Edge e = graph[curr].get(i);  //neighbour
if(col[e.dest] == -1){
int nextCol = col[curr] == 0 ? 1 : 0;
    col[e.dest] = nextCol;
    q.add(e.dest);
} else if(col[e.dest] == col[curr])
    return false; //not bipartite
}}}
    return true;
}
psvm(){
int n=4;
ArrayList<Edge> graph[] = new ArrayList[n];
```

```
createGraph(graph);
SOP("Is Bipartite(graph));
}
output: false.
```

Cycle Detection:

Directed Graph (DFS)

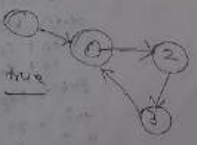
no cycle

cycle

```
Program: import java.util.*;
public class classroom{
    static class Edge{
        int src;
        int dest;
        public Edge(int s, int d){
            this.src = s;
            this.dest = d;
        }
    }
    public static void createGraph(ArrayList<Edge>
                                   graph[]){
for(int i=0; i<graph.length; i++){
graph[i] = new ArrayList<>();
```

```
graph[0].add(new Edge(0,2));
graph[1].add(new Edge(1,0));
graph[2].add(new Edge(2,3));
graph[3].add(new Edge(3,0));
}
public static boolean isCycle(ArrayList<Edge>[]
                              graph){
boolean vis[] = new boolean[graph.length];
boolean stack[] = new boolean[graph.length];
for(int i=0; i<graph.length; i++){
if(!vis[i]){
if(isCyclicUtil(graph,i,vis,stack))
    return true;
}}
    return false;
}
public static boolean isCyclicUtil(ArrayList<Edge>
    []graph,int curr,boolean vis[],boolean
                                   stack[]){
vis[curr] = true;
stack[curr] = true;
for(int i=0; i<graph[curr].size(); i++)
{
Edge e = graph[curr].get(i);
```

```
if(stack[e.dest])      //cycle
    return true;
if(!vis[e.dest] && isCyclicUtil(graph,
        e.dest, vis,stack))
{
    return true;
}
}
stack[curr] = false;
    return false;
}
psvm(){
int v=4;
ArrayList<Edge> graph[] = new ArrayList[v];
createGraph(graph);
SOP("isCycle"+graph));
}
```



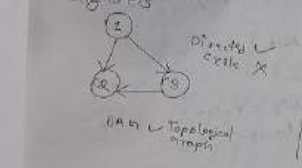
Output: true

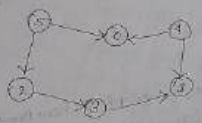
Topological Sorting : (DAG)

Note:
- Directed Acyclic Graph (DAG) is a directed graph with no cycles.
- Topological Sorting is used only for DAG (not for non-DAGs)

Ex

- Using 10 Edges





output
5, 4, 2, 3, 1, 0

Program: import Java.util.*;
public class Testcase{
static class Edge{
int src;
int dest;
public Edge(int s, int d){
this.src = s;
this.dest = d;
}
}

static void createGraph(ArrayList<Edge> graph[])
{
for(int i=0; i<graph.length; i++){
graph[i] = new ArrayList<>();
}
graph[2].add(new Edge(2,3));
graph[3].add(new Edge(3,1));
graph[4].add(new Edge(4,0));
graph[4].add(new Edge(4,1));
graph[5].add(new Edge(5,0));
graph[5].add(new Edge(5,2));
}
public static void topSort(ArrayList<Edge>[] graph){
boolean vis[] = new boolean[graph.length];
Stack<Integer> s = new Stack<>();
for(int i=0; i<graph.length; i++){
if(!vis[i]){
topSortUtil(graph, i, vis, s);
}
}

```
while(!s.isEmpty()){
System.out.print(s.pop()+" ");
}
}
public static void topSortUtil(ArrayList
<Edge>[] graph, int curr, boolean[] vis,
Stack<Integer> s){
vis[curr] = true;
for(int i=0; i<graph[curr].size(); i++){
Edge e = graph[curr].get(i);
if(!vis[e.dest]){
topSortUtil(graph, e.dest, vis, s);
}
}
s.push(curr);
}
public static void main(String args[]){
int V = 6;
ArrayList<Edge> graph[] = new ArrayList[V];
createGraph(graph);
topSort(graph);
}
```

O(V+E)

output: 5 4 2 3 1 0

Q. Reverse Number Print

Solution:

```
import java.util.*;
public class Solution {
public static void main(String args[]) {
int n = 234567;
while(n>0){
int lastDigit = n%10;
System.out.print(lastDigit);
n = n/10;
}
System.out.println();
}
}
```

output: 7 6 5 4 3 2

Q. Reverse the Given Number

Solution:

```
import java.util.*;
public class Solution {
public static void main(String args[]) {
int n = 234567;
int rev = 0;
while(n>0){
int lastDigit = n%10;
rev = (rev*10) + lastDigit;
n = n/10;
}
System.out.println(rev);
}
```

output: 7 6 5 4 3 2

Topic 6: Arrays

Arrays: list of elements of the same type.
: size fixed

Creating an Array

```
datatype arrayName[] = new datatype[size];

// creating an array
int marks[] = new int[50];
int numbers[] = {1, 2, 3};
int moreNumbers[] = {4, 5, 6};
String fruits[] = {"apple", "mango", "orange"};
```

Example:

```
import java.util.*;
public class solution {
    public static void main(String args[]) {
        int marks[] = new int[100];
        Scanner sc = new Scanner(System.in);
        marks[0] = sc.nextInt();
        marks[1] = sc.nextInt();
        marks[2] = sc.nextInt();
        System.out.println("Phy: " + marks[0]);
        System.out.println("Chem: " + marks[1]);
        System.out.println("Math: " + marks[2]);
    }
}
```

Enter: 98
95
96

Output: 98
95
96

6. Linear Search

finding index of element in a given array.
Search Key = 10

2	4	6	8	10	12	14	16

Solution:

```
import java.util.*;
public class solution {
    public static int linearSearch(int number[], int key) {
        for(int i=0; i< number.length; i++) {
            if(number[i] == key) {
                return i;
            }
        }
        return -1;
    }
    public static void main(String args[]) {
        int number[] = {2,4,6,8,10,12,14,16};
        int key = 10;
        int index = linearSearch(number, key);
        if(index == -1) {
            System.out.println("NOT exist");
        } else {
            System.out.println("key is at index: " + index);
        }
    }
}
```

Output: key is at index: 4

String Search

```
import java.util.*;
Public class solution {
// create function
Public static int linearSearch (String fruits[], String key)
{
for (int i=0; i<fruits.length; i++) {
  if (fruits[i] == key) {
    return i;          ------ exit key match
  }
  return -1;           ------ // not exit
}
Public static void main (String args[]) {
String fruits[] = {"Apple", "mango", "banana", "Orange",
                   "lemon"};
int index = linearSearch(fruits, key);
if (index == -1) {
  System.out.println ("Element not exists");
} else {
System.out.println ("Element at index is" + index);
}}}
```

Output: Elent at index is: 3

9. Largest Number Array

find the largest number in a given array.

1	2	6	3	5

output = 6

Solution:

largest = -∞ 1, 2, 6

1	2	6	3	5
0	1	2	3	4

```
import java.util.*;
Public class Solution {
  public static int getlargest (int numbers[]) {
  int largest = Integer.MIN_VALUE;
  for (int i=0; i<numbers.length; i++) {
    if (largest < numbers[i]) {
      largest = numbers[i];
    }
    return largest;
  }
  public static void main (String args[]) {
  int numbers[] = {1, 2, 6, 3, 5};
  System.out.println ("Largest value is: " + getlargest(largest));
  }
}
```

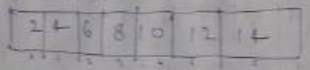
Output: 6

Smallest value find

```
import java.util.*;
public class Solution {
public static int getSmallest(int number[])
{
int smallest = Integer.MAX_VALUE;
for(int i=0; i<number.length; i++) {
if(number[i] < smallest) {
smallest = number[i];
}
}
return smallest;
}
public static void main(String args[]) {
int number[] = {1,2,3,6,5};
System.out.println("Smallest value is: "+getSmallest(number));
}
}
```

output: Smallest value is: 1

Q. Binary Search



output: 4

Search key = 10

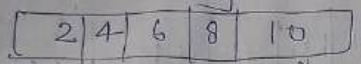
Note: Binary Search key always sorted Order.

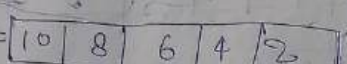
Solution:


```
import java.util.*;
public class Solution {
public static int binarySearch(int number[], int key)
{
int start=0, end=number.length-1;
while(start <= end) {
int mid = (start+end)/2;
//comparison
if(number[mid] == key) { //found
return mid;
}
if(number[mid] < key) { //right
start = mid+1;
} else {
end = mid-1;
}
}
return -1;
}
P.sum() {
int number[] = {2,4,6,8,10,12,14};
int Key=10
SOP("Index for Key is: "+binarySearch(number, Key));
}
```

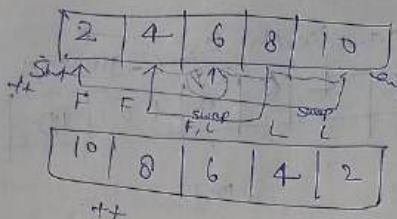
output: 4

Q. Reverse an Array



output = 

Solution:




```
import java.util.*;
public class Solution{
public static void getReverse(int number[]){
    int start = 0, end = number.length - 1;
  while ( start < end ){
     int temp = number[end];
     number[end] = number[start];
     number[start] = temp;
       start++
       end--
     }}
public static void main(String args[]){
     int number[] = {2, 4, 6, 8, 10};
     reverse (number);
     //printfunction
 for(int i=0; i< number.length; i++){
     System.out.print(number[i]+" ");
       }
     System.out.println();
  }}
```

Q. Pairs in an Array

2	4	6	8	10

Solution

(2,4) (2,6) (2,8) (2,10)
(4,6) (4,8) (4,10)
(6,8) (6,10)
(8,10)

```
import java.util.*;
public class solution {
public static void printPairs(int numbers[]) {
for(int i=0; i<numbers.length; i++) {
int curr = numbers[i]; //2, 4, 6, 8, 10
for(int j=i+1; j<number.length; j++) {
System.out.print("("+curr+","+number[j]+")");
}
System.out.println();
}}

public static void main(String args[]){
int numbers[] = {2,4,6,8,10};
printPairs(numbers);
}
}
```

* Pair Sum with total count *

O(n^2)

```
import java.util.*;
Public Class Solution {
public static void printPair(int number[]){
    int tp=0;
for(int i=0; i<number.length; i++) {
    int curr = number[i];
for( int j=i+1; j<number.length; j++){
System.out.print("(" +curr+ ", " +number[j]+ ") ");
    tp++;
    }
System.out.println();
    }
    System.out.println("total pairs" +tp);
    }
public static void main(String args[]) {
int number[] = {2,4,6,8,10};
    printPair(number);
    }}
```

output: (2,4) (2,6) (2,8) (2,10)
(4,6) (4,8) (4,10)
(6,8) (6,10)
(8,10)
total pairs = 10;

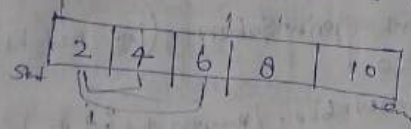
Q. Print Subarrays

↳ a continuous part of array

Ex 2, 4, 6, 8, 10 ✓

2 4 8 3 5 10 ✗

Q.




```
import java.util.*;
public class Solution {
public static int printSubarray(int numbers[]){
for(int i=0; i<numbers.length; i++){
    int start = i;
for(int j=i; j<numbers.length; j++)
    int end = j;
for(int k=start; k<=end; k++){      // print stat
System.out.print(numbers[k]+" ");
}
System.out.println();
}
System.out.println();
}
psvm(String args[]){
int numbers[]={2,4,6,8,10};
printSubarray(numbers);
}
}
```

Output

```
2
2 4
2 4 6
2 4 6 8
2 4 6 8 10
4
4 6
4 6 8
4 6 8 10
6
6 8
6 8 10
8
8 10
10
```

C = Dennis Ritch
Java = James Gosli
C++ = Bjarne Stroo
JS = Brendan E
PHP = Rasmus Ler
PYTHON = Guido van

Total subarray print

```
import java.util.*;
public class solution{
public static int printSubarray(int number[]){
    int tp=0;
for(int i=0; i<number.length; i++){
    int start=i;
for(int j=i; j<number.length; j++){
    int end=j;
for(int k=start; k<=end; k++){
System.out.println(number[k]+" ");
    }
    tp++;
    System.out.println();
    }
    System.out.println()
    }
    System.out.println("total subarray="+tp);
    }
 psvm(){
int number[]={2,4,6,8,10};
printSubarray(number);
    }
}
```

Q. MAX SUBARRAY PRINT

O(n²)

```
import java.util.*;
public class Solution {
public static void maxSubarray(int number[]){
int currSum = 0;
int maxSum = Integer.MIN_VALUE;
int prefix[] = new int [number.length];
prefix[0] = number[0];
// calculate prefix array
for(int i=1; i<prefix.length; i++){
prefix[i] = prefix[i-1] + numbers[i];
}
for(int i=0; i<number.length; i++){
int start = i;
for(int j=i; j<number.length; j++){
int end = j;
currSum = start==0 ? prefix[end] : prefix[end] -
prefix[start-1];
if(maxSum < currSum){
maxSum = currSum;
}
}
}
System.out.println("max sum = "+ maxSum);
}
psum(){
int number[] = {1,-2,6,-1,3};
maxSubarray(number);
}
```

(output: 8)

* Q. Max Subarray Sum (Kadane's algorithm) Time O(n)

-2, -3, 4, -1, -2, 1, 5, -3

Solution:

	0							
(cs) Current sum	0	0	4	3	1	2	7	4
(ms) Max sum	0	0	4	4	4	4	7	7
-∞ index	0	1	2					

max output

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

negative number = 0

```
import java.util.*;
public class solution {
public static void Kadanas(int number[]) {
    int cs=0;
    int ms = Integer.MIN_VALUE;
for(int i=0; i<number.length; i++) {
    cs = cs + number[i];
    if(cs <0) {
        cs=0;
    }
    ms = Math.max(cs, ms);
    }
    System.out.println("max sum subarray is :" +ms);
    }
    Psum() {
    int number[] ={-2, -3, 4, -1, -2, 1, 5, -3};
    Kadanas(number);
    }
}
```

Q. Given an integer number return true if any value appears at least twice in the array, and return false if every element is distinct.

Ex
nums = [1, 2, 3, 1] true ✓
nums = [1, 2, 3, 4] false ✗

1. Method

```
public static boolean twiceArray(int number[]){
for(int i=0; i<number.length; i++){
for(int j=i+1; j<number.length; j++){
if(number[i]==number[j]){
return true;
}}
return false;}
```

2. Method

```
public static boolean twiceArray(int number[]){
HashSet<Integer> set = new HashSet<>();
for(int i=0; i<number.length; i++){
if(set.contains(number[i])){
return true;
}else{
set.add(number[i]);
}}
return false;
}
```

* array second Highest number print

```
int[] = {4, 6, 7, 91, 34, 67}
public static void getSecond(int[] a) {
int SecondHighest = Integer.MIN_VALUE;
int hight = Integer.MIN_VALUE;
for(int i=0; i<a.length; i++) {
if (a[i] > hight) {
SecondHighest = hight;
hight = a[i];
} else if (a[i] > secondHighest && a[i] != hight)
{
secondHighest = a[i]
}
}
sop(+SecondHright);
}
```

67 output

Q. arr = [12, 35, 1, 10, 34, 1]
Output: 34

```
class solution {
public int printSecondLargest (List<Integer> arr) {
    int max = Integer.MIN_VALUE;
    int secondMax = Integer.MIN_VALUE;
    for (int num : arr) {
        if (num > max) {
            second = max;
            max = num;
        } else if (num > secondMax && num != max)
        {
            secondMax = num;
        }
    return secondMax == Integer.MIN_VALUE ? -1 : secondMax;
    }
}
```

Q. check if an Array is Sorted

```
class TUF {
    static boolean isSorted (int arr[], int n) {
    for (int i=1; i<n; i++) {
        if (arr[i] < arr[i-1])
            return false;
    }
    return true;
}
```

n = 5
a = [1, 2, 3, 9, 5]
output = 1

26. Remove Duplicates from Sorted Array

arr[] = {1,1,2,2,2,3,3,...};
output : {1,2,3,4,...}

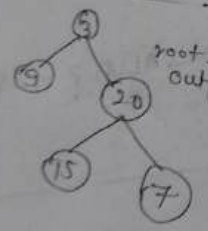
```
public static int removeDuplicates(int[] arr)
{
int i=0;
for(int j=1; j<arr.length; j++){
if(arr[i] != arr[j]){
i++;
arr[i]=arr[j];
}
return i+1;
}
```

O(N) - Time
O(1) - Space

54

T.C.
O. Level Order Traversal

FIFO
queue

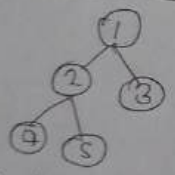


root = [3,9,20,null,null,15,7]
output: [[3],[9,20],[15,7]]

```
class Solution{
public List<List<Integer>> levelOrder(TreeNode root){
List<List<Integer>> ans = new ArrayList<>();
if(root == null) return ans;
Queue<TreeNode> q = new LinkedList<>();
q.add(root);
while(!q.isEmpty()){
int size = q.size();
```

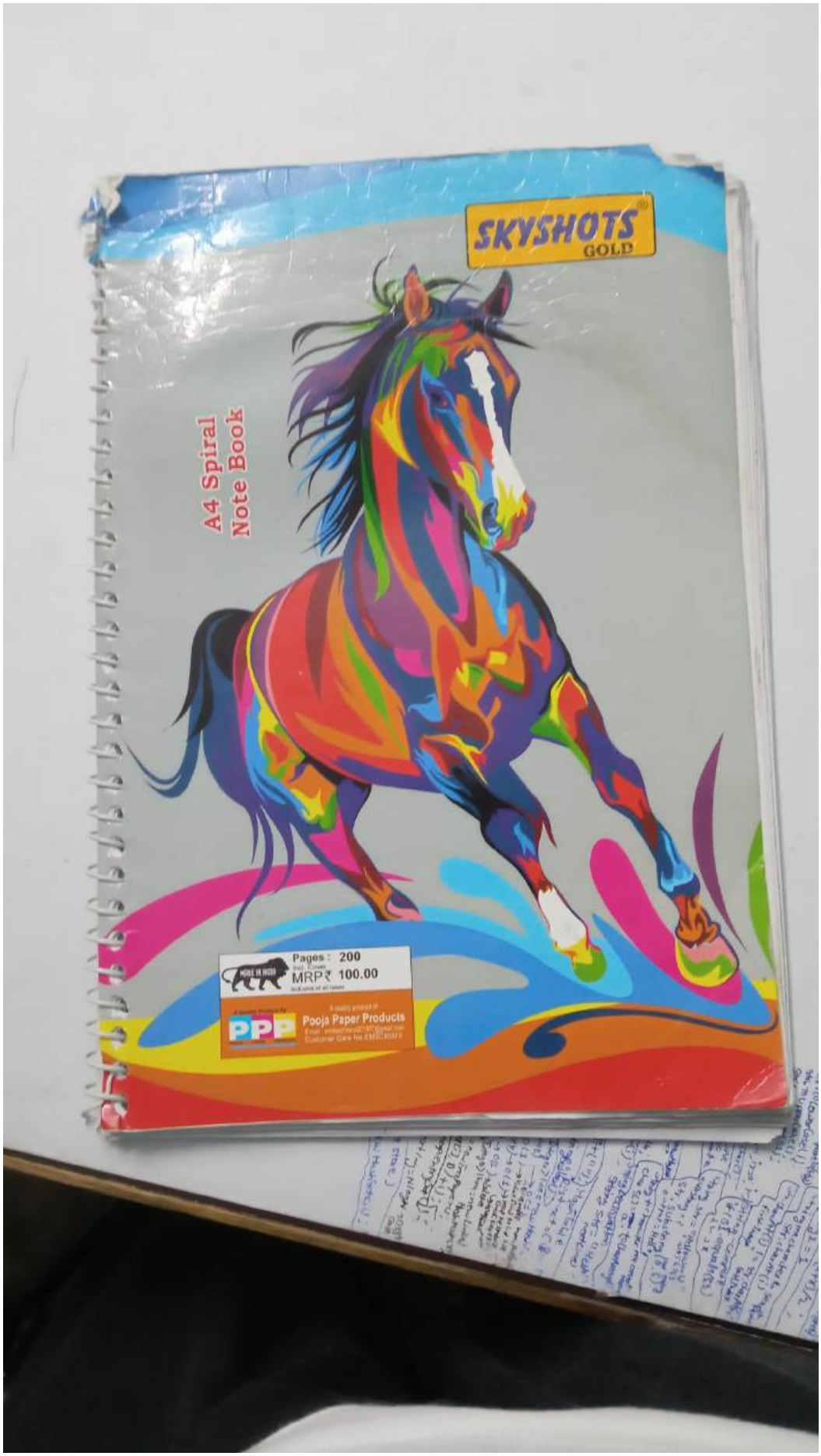
```
List<Integer> level = new ArrayList<>();
for(int i=0; i< size; i++) {
    TreeNode node = q.poll();
    level.add(node.val);
    if(node.left != null) {
        q.add(node.left);
    }
    if(node.right != null) {
        q.add(node.right);
    }
    ans.add(level);
}
return ans;
}
}
```

543 Diameter of Binary tree:



Input: root = [1,2,3,4,5]
output: 3

```
class Solution {
    int maxDiameter = 0;
    public int dfs(TreeNode root){
        if(root == null) return 0;
        int leftdfs = dfs(root.left);
        int rightdfs = dfs(root.right);
        maxdiameter = Math.max(maxdiameter, leftdfs + rightdfs);
        return Math.max(leftdfs, rightdfs)+1;
    }
    public int diameterOfBinaryTree(TreeNode root) {
        dfs(root);
        return maxdiameter;
    }
}
```


SKYSHOTS
GOLD
A4 Spiral
Note Book
Pages : 200
MRP ₹ 100.00
PPP
Pooja Paper Products

Level-1

* CSS *

CSS → Cascading style sheet :- It is a language ... It is used to describe the style of a document.

Basic Syntax

```
Selector
  ↓
  h1 {
        color: red;
  }     ↳ Property   value
```

Including Style color:

- Inline

<h1 style = "color: red"> MNNIT ALLAHABAD</h1>

- <style> tag

```
<style>
   h1 {
        color: red;
   }
</style>
```

output: MNNIT ALLAHABAD (Red Color)

- External Stylesheet:

writing CSS in a separate document & linking it with HTML file.

link karne ke liye:

<link rel = "stylesheet" href = "filename.css">

File banana hai (ClassRoom)

→ File click → open Folder click → ClassRoom

ClassRoom New File → index.html (Example)

New File Create → style.css (Example)

index.html (Program write)

!+ 1 → ! (Shortcut) (Program write)

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<h1>Hello From Apna College</h1>
<h2>Hello From MNNIT</h2>
</body>
</html>
```

Ctrl + S → Save

↓ Run

Hello From Apna College

Hello From MNNIT

style.css (Program write)

```
h1 {
   color: red;
}
```

Ctrl + S click → Save

```
P {
   color: rgb(120, 30, 45);
}
button {
   color: rgb(57, 15, 42);
   background-color: white;
}
body {
   background-color: #DEF...;
}
```

*** Color Property *** : used to set the color of ...

color: red;
color: pink;
color: blue;
color: green;

*** Background Color Property:** used to set the color of background.

background-color: red;
background-color: pink;
background-color: blue;
background-color: green;

*** Color System:**

- RGB

Color: rgb(255, 0, 0);
Color: rgb(0, 255, 0);

R → Red
G → Green
B → Blue

color picker → Check / Create a Palette

*** Color Systems:**

- Hex (Hexadecimal)

Color: #ff0000;
Color: #00ff00;

Note: RGB Color Create

ff 00 00 (R G B)

R: 0-255, 00-ff
G: 0-255, 00-ff
B: 0-255, 00-ff

Yellow create color: R+G

yellow → #ffff00

Universal Selector

```
* { }
```

Element Selector

```
h1 { }
```

id Selector

```
#myId { }
```

- Class Selector

```
.myClass { }
```

Note:

* = asterisk
& = ampersand

Example

```
* {
    color : black;
}
button {
    color : aqua;
}
#headingOne {
    color : yellow;
}
<h1 id="headingOne">Hello from Apna College</h1>

.myClass {
    color : magenta;
}
<h1 class="myclass">This is heading1</h1>
<p class="myclass">This is the paragraph</p>
```

Note:

```
.myclass → [illegible] like
<div class="myclass"></div>
#myid → [illegible] <div id="myid"></div>
#box → [illegible] <div id="box"></div>
```

comment (short cut) → (ctrl + /) click

Q. Create a simple div with an id "box". Add some text content inside the div. Set its background color to blue.

Ans

Program

index.html

```
<link rel="stylesheet" href="style.css">
</head>
<body>
<div id="box">
  [illegible]
</div>
</body>
</html>
```

style.css

```
#box {
    background-color: blue;
}
```

② Create 3 headings with h1, h2 & h3. Given them all a class "heading" & set color of "heading" to red.

Ans index.html

```
<h1 class="heading">Heading1</h1>
<h2 class="heading">Heading2</h2>
<h3 class="heading">Heading3</h3>
```

style.css

```
h1, h2, h3 {
    color: red;
}
or
.heading {
    color: red;
}
```

Note comment → ctrl + /

Q. A button & set its background color to:
- green using css stylesheet
- blue using <style> tag
- pink using inline style

index.html

```
<link rel="stylesheet" href="style.css">
<style>
button {
    background-color: blue;
}
</style>
<body>
<h2>      </h2>
<button style="background-color: pink;">click me!</button>
</body>
</html>
```

style.css

```
button {
    background-color: green;
}
```

* Text Properties :

text-align :

Syntax: text-align: left/right/center

Example:

```
p {
    text-align: center;
}
h1 {
    text-align: center
}
```

Note: [illegible]

passage ko
- left side
- or right side
- center

* Text Properties :

text-decoration :

Syntax

text-decoration : underline / overline / line-through

Example

```
h1 {
    text-align : center;
    text-decoration: blue underline;
} or text-decoration: blue wavy underline;
```

or solid / double / dotted / dashed

* Text Properties :

font-weight

Syntax:

font-weight : normal / bold / bolder / lighter

font-weight : 100 -- 900

lightest 100, darkest 900, bold (400)

Example

```
#h1 {
    font-weight: 100;
}
```

→ id

* Text Properties :

font-family

Syntax:

```
font-family : arial ;
font-family : arial, roboto ;
```

Example

```
#one
{
    font-family : arial;
}
```

change

CSS → 2 Type

...olute ② Relative

...te:

Pixels (PX)

96px = 1 inch = 2.54 cm

font-size: 2px;

Ex

```
p {
  fontsize: 20px;
}
```

* Text Properties:

line-height

line-height: 2px;

line-height: normal;

Ex

```
stone {
  line-height: 2px;
}
```

* Text Properties

text-transform

text-transform: uppercase/lowercase/capitalize/one

Ex

```
#one {
  text-transform: ;
}
```

Practise set-2

Q. Create a heading centred on the page with all its text capitalised by default.

Program: index.html

```
<body>
<h1>m.n.n.i.t alphabet</h1>
</body>
</html>
```

style.css

```
h1 {
  text-align: center;
  text-transform: capitalize;
}
```

Q. Set the font family of all the content in the document to "Times New Roman".

Program: style.css

```
* {
  font-family: Times New Roman;
}
```

Q. Create one div inside another div. Set id & text "outer" for the first one & "inner" for the second one. Set the outer div text size to 25px & inner div text size to 10px.

Sol: index.html

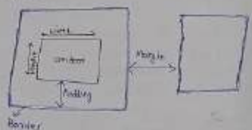
```
<link rel="stylesheet" href="style.css
</head>
<body>
<div id="outer">
  outer
<div id="inner">
  inner
</div>
</div>
</body>
</html>
```

style.css

```
#outer {
  font-size: 25px;
}
#inner {
  font-size: 10px;
}
```

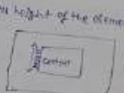
Box Model in CSS:

* Padding
* Margin



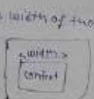
* Height: It set the content area height of the element.

```
div {
  height: 50px;
}
```



* Width: It set the content area width of the element

```
div {
  width: 50px;
}
```



* Border: used to set an element's border

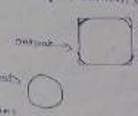
```
border-width: 2px;
border-style: solid/dotted/dashed
border-color: black;
```

* Border "Shorthand"

```
border: 2px solid black;
```

* Border: used to round the corners of an element's outer border edge.

```
border-radius: 10px;
border-radius: 50%
```



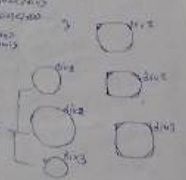
Ex

style.css

```
div {
  height: 100px;
  width: 100px;
  background-color: brown;
  border: 2px solid brown;
  text-align: right;
  border-radius: 15px;
  border-radius: 50%;
}
```

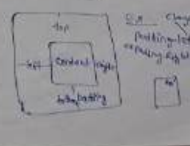
index.html

```
</head>
<body>
<div>div1</div>
<br>
<div>div2</div>
<br>
<div>div3</div>
</body>
</html>
```



* Padding:

* Padding-left
* padding-right
* padding-top
* padding-bottom



Ex

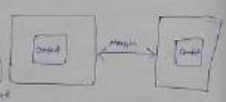
```
padding-left: 10px
padding-right: 10px
```

padding

```
padding: 50px;
padding: 1px 2px 3px 4px;
         top/right/bottom/left→clockwise
```

Margin (Outside)

- margin-right
- margin-left
- margin-top
- margin-bottom



Ex: margin-right: 20px;

- margin: 50px;
- margin: 1px 2px 3px 4px
  top/right/bottom/left→clockwise

Practice Set 3

Q. Create a div with height & width of 100px. Set its background color to green & the border radius of 50%.

Sol:

index.html

```
<html>
<body>
<div></div>
</body>
</html>
```

style.css

```
div {
  height: 100px;
  width: 100px;
  background-color: green;
  border-radius: 50%;
}
```

output: (green circle)

Q. Create the following navbar.

Program write

index.html

```
<link rel="stylesheet" href="style.css">
</head>
<body>
<div id="navbar">
<a id="logo" href="https://www.amazon.in/">amazon.in</a>
<a href="#">Account</a>
<a href="#">my cart</a>
<a href="#">Contact Us</a>
<input placeholder="Search Amazon.in">
<button>Search</button>
</div>
</body>
</html>
```

style.css

```
* {
  padding: 0;
  margin: 0;
  color: white;
}
#navbar {
  height: 50px;
  background-color: #0f1111;
}
button {
  background-color: #febd69;
}
#logo {
  color: #febd69;
  font-size: 25px;
}
a {
  margin-right: 150px;
}
```

display

display: inline/block/inline-block/none

IL: Takes only the space required by the element (no margin/padding)

ck: Takes full space available in width.

- inline-block: Similar to inline but we can set margin & padding
- none: To remove element from document flow.

Ex

index.html

```
<body>
<div>This is div</div>
<button>Click me</button>
</body>
</html>
```

style.css

```
div {
  background-color: white;
}
```

output: This is div / Click me

Change

```
<body>
<div>This is div</div>
<button>Click me</button>
</body>
</html>
```

style.css

```
div {
  background-color: white;
  display: inline;
}
button {
  background-color: brown;
}
```

output: This is div / Click me

Visibility

visibility: hidden;

Note: when visibility is set to none, space for the element is reserved but for display set to none no space is reserved or blocked for the element.

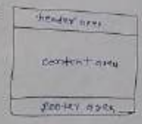
Alpha Channel (color code)

opacity (0 to 1)

- RGBA

```
color: rgba(255,0,0,0.5);
color: rgba(255,0,0,1);
```

Note:



Lab

Create a webpage layout with a header, a footer & content area containing 8 divs.

Set the height & width of divs to 100px.

(Add the previous navbar in the header)

Q. Add borders to all the divs.

Q. Add a different background color to each div with an opacity of 0.5.

Q. Give the content area an appropriate height.

Sol: index.html

```
<html>
<header>
<div id="navbar">
<a href="/" id="logo">amazon.in</a>
<a href="">My Account</a>
<a href="">My Cart</a>
<a href="">Contact us</a>
<input placeholder="Search Amazon.in">
<button>Search</button>
</div>
</header>
<div id="content">
<div class="contentbox" style="background-color: rgba(255,0,0,0.5);">[illegible]</div>
<div class="contentbox" style="background-color: rgba(0,0,255,0.5);">[illegible]</div>
<div class="contentbox" style="background-color: rgba(0,0,0,0.5);">[illegible]</div>
</div>
<footer>
made with &hearts; by Amazon <br>
&copy; 1996-2023, Amazon.com, Inc. or its affiliates
</footer>
</body>
</html>
```

style.css

```
* {
  margin: 0;
  padding: 0;
}
#navbar {
  background-color: #0f1111;
  color: white;
  height: 60px;
}
a {
  text-decoration: none;
  color: white;
  margin-left: 20px;
}
#logo {
  margin-left: 20px;
  font-size: 20px;
  color: #f08804;
}
input {
  margin-left: 180px;
}
button {
  background-color: #febd69;
  color: white;
}
.contentbox {
  height: 100px;
  width: 100px;
  border: 1px solid black;
  display: inline-block;
}
#content { height: 200px; }
footer {
  background-color: #0f1111;
  color: white;
  text-align: center;
}
```

In css

Relative

%

em

rem

* Percentage (%):

It is often used to define a size as relative to an element's parent object.

width: 33%

margin-left: 50%

* Em:

Unit	Relative to
Em	Font size of the parent, in the case of typographical properties like font-size, and font size of the element itself, in the case of other properties like width.

* Rem (Root Em)

Unit	Relative to
rem	font size of the root element

* Others:

vh: relative to 1% viewport height

vw: relative to 1% viewport width

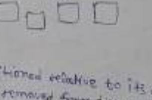
* Position:

The position CSS property sets how an element is positioned in a document.

position: static / relative / absolute / fixed

* static: default position (The top, right, bottom, left, and z-index properties have no effect)

* relative: element is relative to itself. (The top, right, bottom, left, and z-index will work)



left side → output

5x

```
.box {
  position: relative;
  top: 20px;
}
```

* absolute: positioned relative to its closest positioned ancestor (removed from the flow)

* fixed = positioned relative to browser (removed from flow)

* sticky = positioned based on user's scroll position.

* z-index: it decides the stack level of elements.

Overlapping elements with a larger z-index cover those with a smaller one.

z-index: auto (0)

z-index: 1/2/...

z-index: -1/-2/...

Ex

Level: Unsplash – Pixel Library

ground Image: used to set an image as background.

```
background-image: url("image.jpg");
```

Background Size:

```
background-size: cover/contain/auto
```

Practice set 5

Q. Create the following layout using the given html).
* Give the div a height, width & some background image.
* Use the appropriate position property for the div element to place it at the right end of the page. (The div should not move, even on scroll)

```
<p> lorem*5 </p>
<div> Love Nature </div>
<p> lorem*5 </p>
```

position property / relative / absolute / fixed

Ans: index.html

```
<title>
</head>
<body>
<p> lorem*2 </p>
<div> Love Nature </div>
<p> lorem*10 </p>
</body>
</html>
```

Style.css

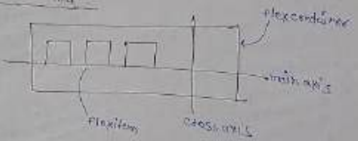
```
div{
  height: 400px;
  width: 400px;
  background-image: url("image.png");
  background-size: cover;
  position: absolute;
  right: 0px;
  position: fixed;
  right: 0px;
  bottom: 0px;
}
```

Level 4

* Flexbox:

Flexible Box Layout: It is a one-dimensional layout method for arranging items in rows or columns.

* The Flex model:

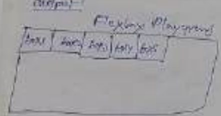


* Flexbox Direction: It sets how flex items are placed in the flex container, along which axis and direction.
* flexbox-direction: row; (default) ⟹ L → R
* flexbox-direction: row-reverse; ⟹ R → L
* flexbox-direction: column; ⟹ top → bottom
* flexbox-direction: column-reverse; ⟹ bottom → top

index.html

```
<title>
</head>
<body>
<h1> Flexbox Playground </h1>
<div id="container">
<div id="box1"> box1 </div>
<div id="box2"> box2 </div>
<div id="box3"> box3 </div>
<div id="box4"> box4 </div>
<div id="box5"> box5 </div>
</div>
</body>
</html>
```

Output:



Style.css

```
body{
  text-align: center;
}
div{
  height: 100px;
  width: 100px;
  display: inline-block;
  border: 1px solid black;
}
#container{
  width: 800px;
  height: 300px;
  display: flex;
}
#box1{
  background-color: yellow;
}
#box2{
  background-color: pink;
}
#box3{
  background-color: red;
}
#box4{
  background-color: orange;
}
#box5{
  background-color: brown;
}
```

Flex Properties:

for Flex Container:

* justify-content: alignment along the main axis.
  flex-start / flex-end / center / space-evenly /
* flex-wrap: nowrap / wrap / wrap-reverse
* align-items: alignment along the cross axis.
* align-content: alignment of space between & around the content

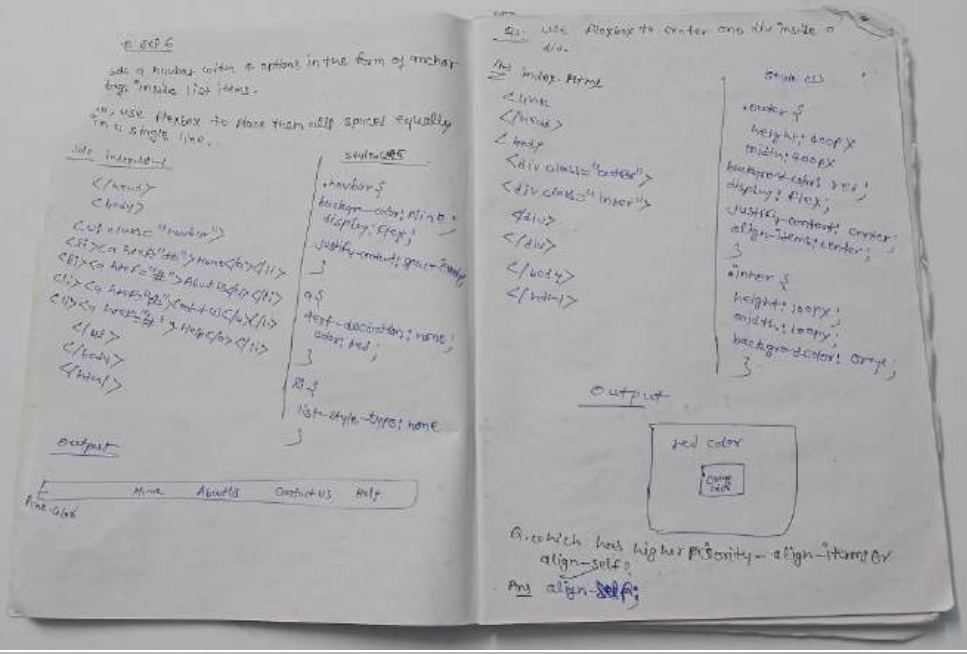
```
Ex #container{
  ...
  flex-direction: row;
  justify-content: center;
  flex-wrap: wrap-reverse;
  align-items: center;
}
```

* Flex Properties:

for Flex Item:

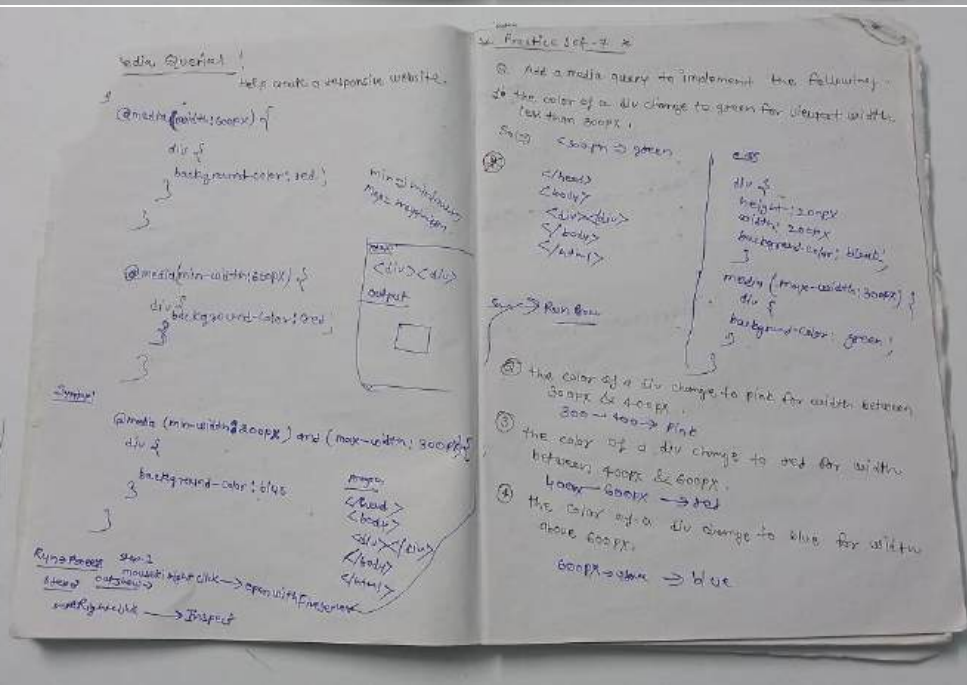
* align-self: alignment of individual along the cross axis.
* flex-grow: how much a flex item will grow relative to the rest of the flex items if space is available.
* flex-shrink: how much a flex item will shrink relative to the rest of the items if space is available.

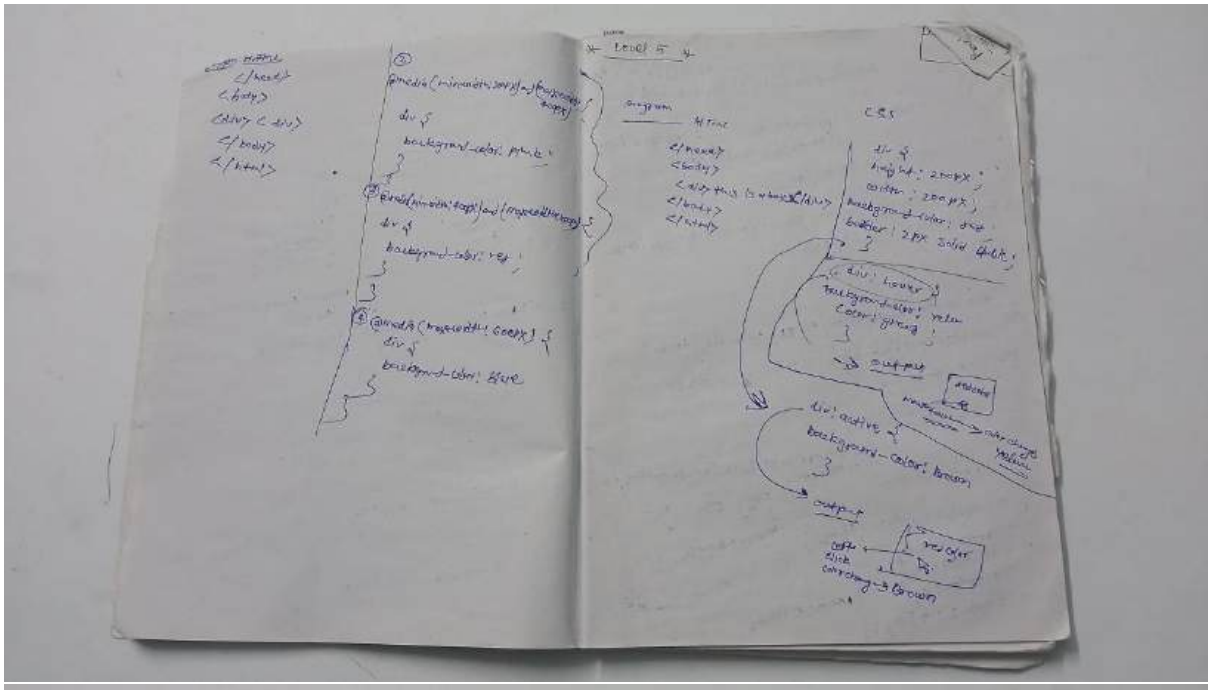
Q. use flexbox to center one div inside a div.

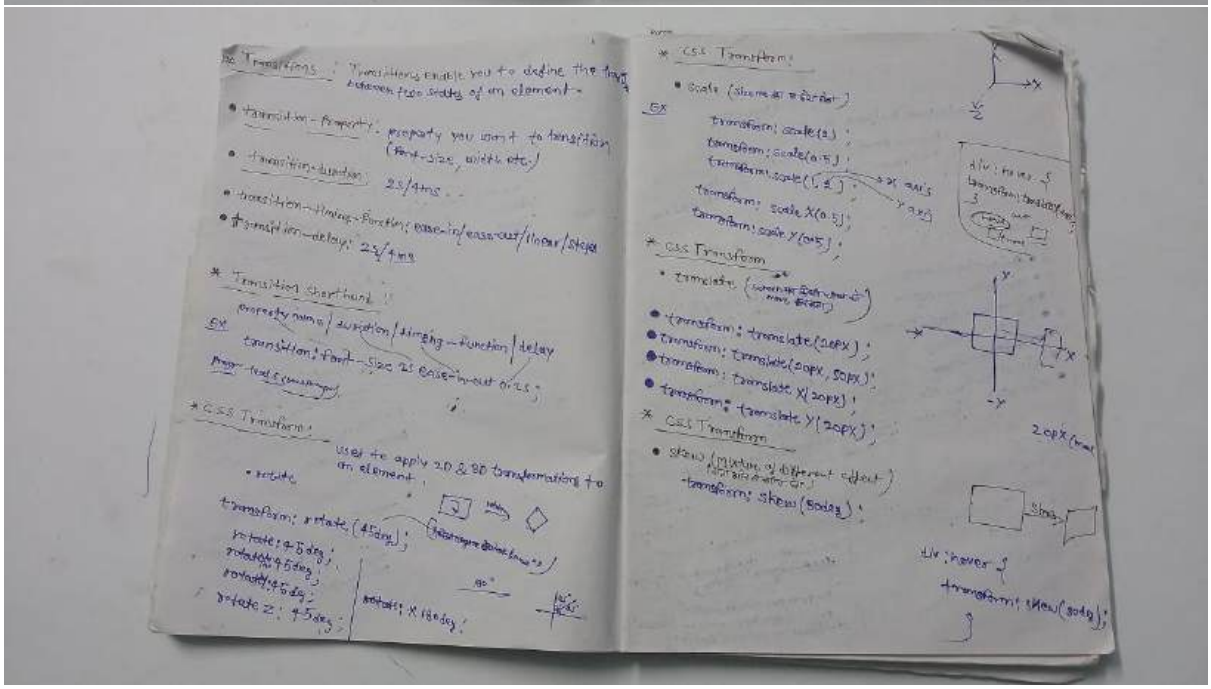


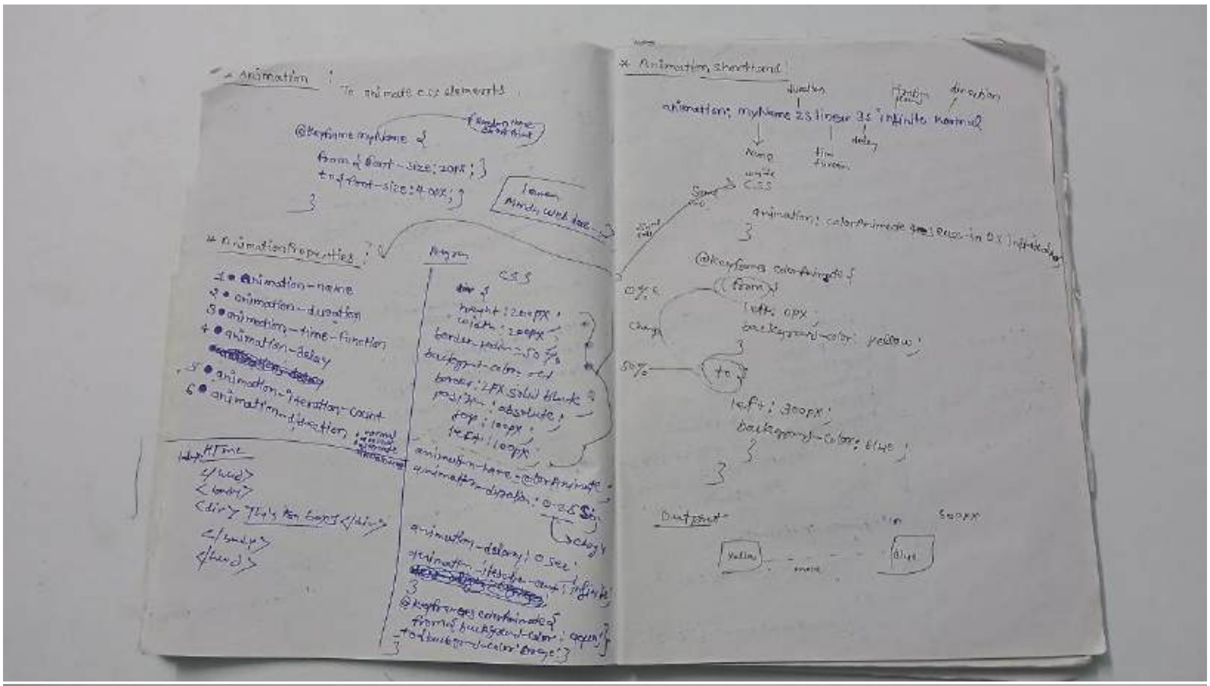
Q. which has higher priority - align-items or align-self?

Ans align-self;

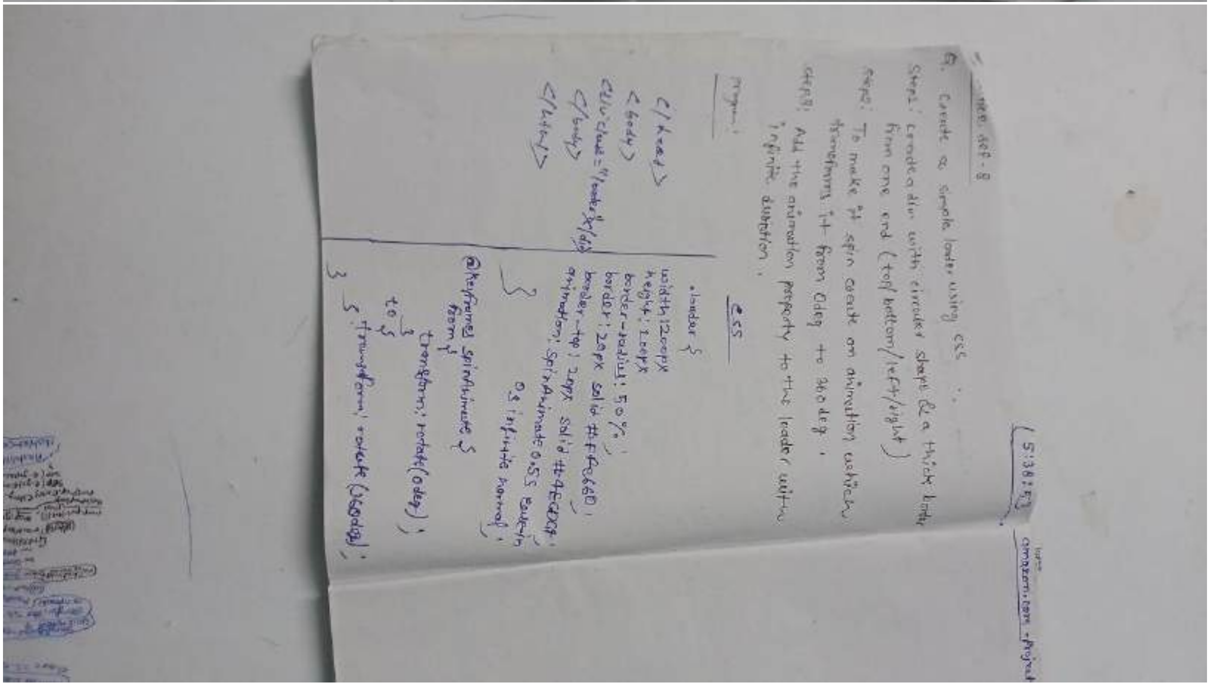




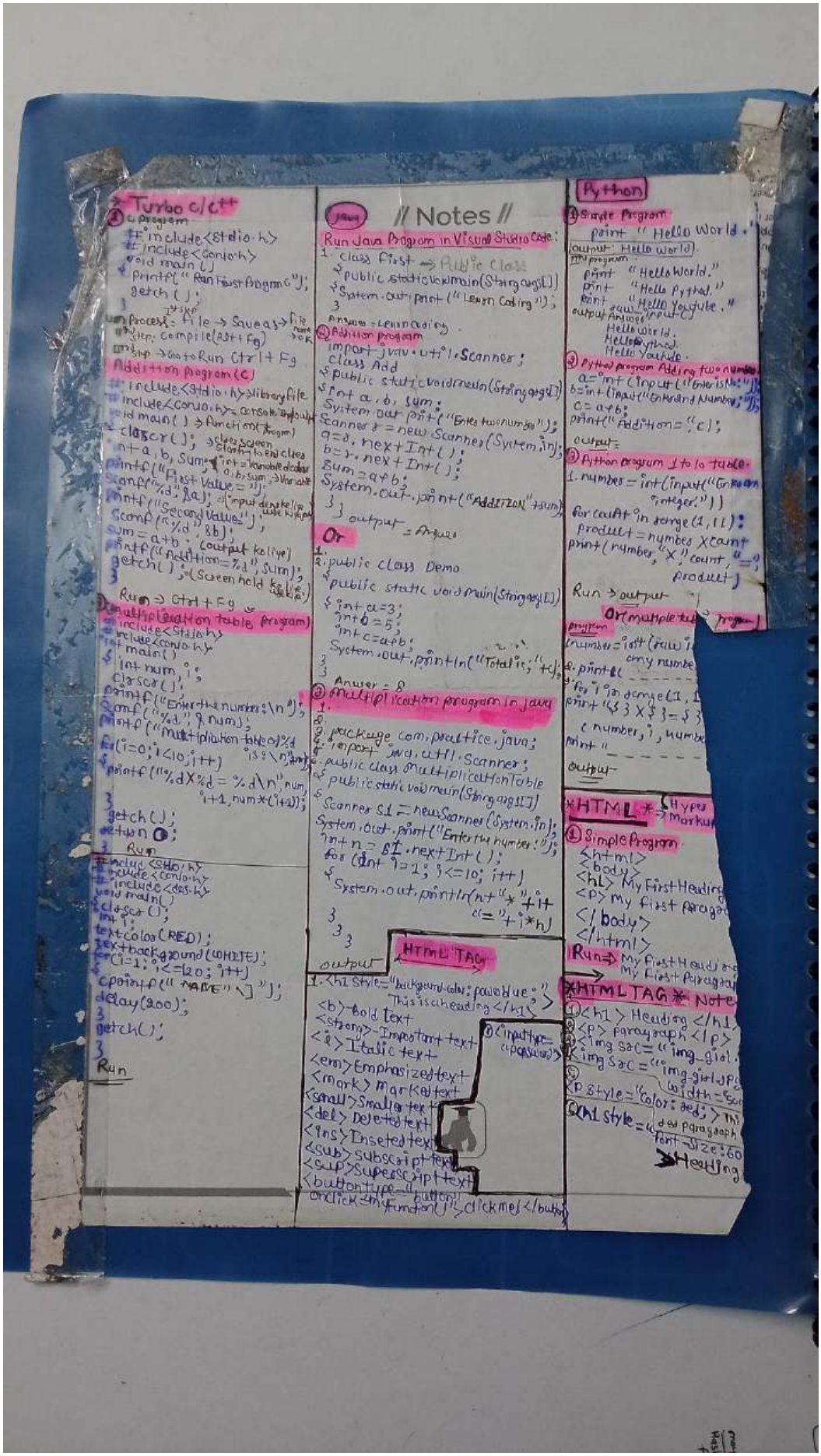




* Animation : To animate css elements.
@keyframe myName {
from { font-size: 20px; }
to { font-size: 400px; }
}
* Animation Properties :
1 • animation-name
2 • animation-duration
3 • animation-time-function
4 • animation-delay
5 • animation-iteration-count
6 • animation-direction
normal
reverse
alternate
HTML
</head>
<body>
<div> This is the boys </div>
</body>
</html>
Program
CSS
div {
height: 200px;
width: 200px;
border-radius: 50%;
background-color: red;
border: 2px solid black;
position: absolute;
top: 100px;
left: 100px;
animation-name: colorAnimate;
animation-duration: 0.25s;
animation-delay: 0 sec;
animation-iteration-count: infinite;
}
@keyframes colorAnimate {
from { background-color: green; }
to { background-color: orange; }
}
* Animation shorthand
duration
iteration
direction
animation: myName 2s linear 3s infinite normal;
Name
time function
delay
write CSS
animation: colorAnimate ease-in 0s infinite;
}
@keyframes colorAnimate {
0%
from {
left: 0px;
background-color: yellow;
}
50%
to {
left: 300px;
background-color: blue;
}
}
Output
500px
Yellow
move
Blue



5:38:53
amazon.com - Project
Q. Create a simple loader using CSS.
Step1: Create a div with circular shape & a thick border from one end (top/bottom/left/right)
Step2: To make it spin create an animation which transforms it from 0deg to 360deg.
Step3: Add the animation property to the loader with infinite duration.
Program:
</head>
<body>
<div class="loader"></div>
</body>
</html>
CSS
.loader {
width: 200px;
height: 200px;
border-radius: 50%;
border: 20px solid #FF6600;
border-top: 20px solid #4EC0C4;
animation: SpinAnimate 0.5s ease-in 0s infinite normal;
}
@keyframes spinAnimate {
from {
transform: rotate(0deg);
}
to {
transform: rotate(360deg);
}
}


Turbo c/c++
Notes
Python
HTML
HTML TAG
HTML TAG Note

* Note - point : *

& — and percentage

() → parathesis

Vs code → Run

1 → gcc fileName.c

2 → a.out

Interest

* Area of Circle = 3.14 * r * r (float)
* Volume of cube = 2 * b * h;
* Simple interest ⇒ float i, P, r, n;
  i = (P * r * n) / 100;
* Area of Triangle = 0.5 * base * height
* Area of Rectangle = breadth * length
* Area of Square = side * side
* Area of circumference = 2 * 3.14 * r;

Note:

#include <stdio.h>

header file

#Include = It is used to include a header file.

#include <stdio.h>

library

header file

tells the program to include

→ Preprocessor directive

std io

Standard input/output

int main ( )

int → return type

main → function name

( ) → Parenthesis

arguments

→ function Signature

void → empty return type

{ → Scope }

printf("%d", a);

int → %d

char → %c

float → %f

double → %lf

scanf("%d", &a);

→ address of

→ used to user se input like be like

Note: Use blackbox.io ai tools

② Look ai

③ Passing lego ideogram.com

vs → https://open-vsx.org/

↳ code में add download karne ke liye.

Turbo → c/c++

① C program

```
# include <stdio.h>
# include <conio.h>
void main ( )
{
Printf (" Ran First Program C ");
getch ( );
}
```

Run Process

Step 1: file → Save as → file name → OK
Step 2: Compile (Alt + F9)
Step 3: Go to Run (Ctrl + F9)

② Addition Program (C)

```
# include <stdio.h> → library file
# include <conio.h> → console Input/output
void main( ) → function program
{
clrscr( ); → clear Screen starting to end clear.
int a, b, sum; → int → Variable declar
                a, b, sum ; → variable
Printf (" First Value ");
Scanf (" %d ", &a);
Printf (" Second Value ");
Scanf (" %d ", &b);
Sum = a+b; → output Keliye
Printf (" Addition = %d ", Sum);
getch ( ); → Screen hold Ke liye use kiya jata hai.
}
```

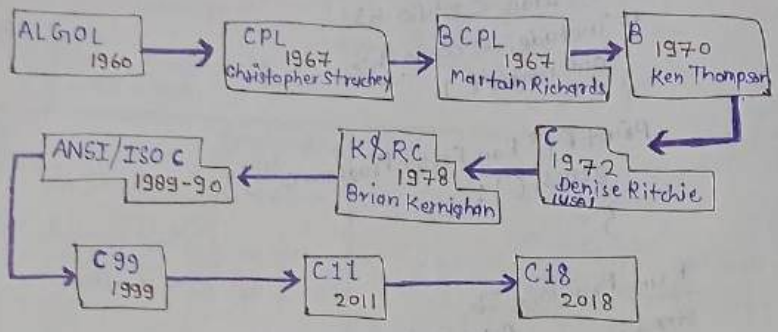
Library function

user se input lene ke liye

input dene Ke liye use kiya jata hai.

Run → Ctrl + F9

* History of C Programming *



Year	Name	Full Form	Used for
1957	FORTRAN	Formula Translation	Mathematical and Scientific application / softwares.
1960	COBOL	Common Business Oriented Language	Business applications Or Softwares
1967	BCPL	Basic Combined Programming Language	General Purpose but with some issues.
1970	B		
1972	C	Basic combined Programming Language	Issues continues- General purpose Programming language so get famous.

Note! * C is a general-purpose of High-Level language.
written in C language

example! Unix, Linux, Windows, Mac, Android etc.

→ C Programming *

Introduction

* History of C programming *

1. ALGOL :- The first most popular high-level language was ALGOL.
   ALGOL was introduced in 1960. It is also called the base father of programming language.
   ALGOL introduced the concept of structured programming.

* Structured Programming :- It mainly used in concept of C programming.
   (a) Sequence.
   (b) Decision
   (c) Repetition.

2. CPL (Common Programming Language) :- CPL was introduced in 1967. it is developed by christopher strachey.
3. BCPL (Basic Combined Programming Language) :- BCPL was introduced in 1967. it is developed by Martain Richards.
   BCPL was used System Software.
4. B :- B programming was introduced in 1967. it is developed by Ken Thompson.

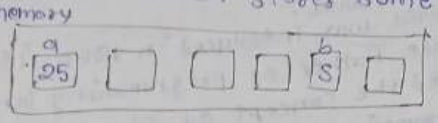
Note:

* Language :- The way of communication between two persons.
* Programming Language :- The way of communication b/w man and machine.

Chapter - 1 * Variables & Identifier

Identifier:

Variables: Variable is the name of a memory location which stores some data.



* Variables Rules :-

(a) Variables are case sensitive, for example < int a = 30;
int A = 40;

(b) 1st character is alphabet or '_' → underscores < int _age = 20;
int age = 22;

(c) no comma / blank space. → example → int final_price = 100;

(d) No other Symbol other than '_' → example < int age = 22;
change
(Update) age = 24;

Program:

```
#include<stdio.h>
 int main()
{
int number = 25;
char star = '*';
int age = 22;
age = 24;
float pi = 3.14;
int a = 30;
int A = 40;
int _age = 22;
int final_price = 100;
return 0;
```

* Variables Data Types *

Data Type	Size in bytes
char or signed char	1
unsigned char	1
int or signed int	2
unsigned int	2
short int or unsigned short int	2
signed short int	2
long int or signed long int	4
unsigned long int	4
float	4
double	8
long double	10

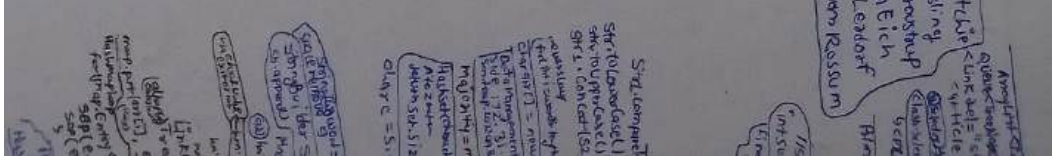
Program example:

```
#include <stdio.h>
int main()
{
int age = 22;
float pi = 3.14;
char hashtag = '#';
return 0;
}
```

* Constants → Values that don't change (fixed)

Types

- Integer constants: 1, 2, 3, 0, -1, -2,
- Real Constants: 1.0, 2.0, 3.14, -2.4
- character Constants: 'a', 'b', 'A', '#', '&'



* Keywords :- Reserved Words that have special meaning to the compiler

↓

32 Keywords in C.

Keywords

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
continue	for	signed	void
do	if	static	while
default	goto	sizeof	volatile
const	float	short	unsigned.

* Comments :- Lines that are not part of program, comments are two types ;

Single Line //

Multiple Line /* */

Program: #include <stdio.h>
// this code displays how data types work in C.
/* This is a multi-line comment
*/
int main() {
printf("Hello world");
return 0;
}

* Output :-

printf("Hello World");

new line
printf("Kuch bhi \n");

Program:

#include <stdio.h>
int main()
{
printf("Hello C \n");
printf("Hello C \n");
return 0;
}

output :- Hello C
Hello C

* Output CASES :-

1. integers
printf("age is %d", age);
2. real numbers
Printf("Value of Pi is %f", pi);
3. characters
Printf("Star looks like this %c", star);

format specificy.

for example

#include <stdio.h>
int main()
{
int age = 22;
printf("age is %d \n", age);
return 0;
}

output :- age is 22.

Input :-

```
scanf("%d", &age);
```

Program :

```
#include <stdio.h>
int main()
{
int a, b;
printf("enter a");
scanf("%d", &a);
printf("enter b");
scanf("%d", &b);
printf("Sum is : %d", a+b);
return 0;
}
```

* Compilation :-

A computer program that translates C code into machine code.

*: Practice Qs 1 *

1. Write a program to calculate area of a square (side is given)

```
#include <stdio.h>
// area of square
int main() {
int side;
printf("enter side");
scanf("%d", &side);
printf("area is : %d", side * side);
return 0;
}
```

Integer Value

```
#include <stdio.h>
// area of square
int main() {
float side;
printf("enter side");
scanf("%f", &side);
printf("area is : %f", side*side);
return 0;
}
```

Float Value

* Practice Qs 2 *

1. Write a program to calculate area of a circle. (radius is given) [πr^2]

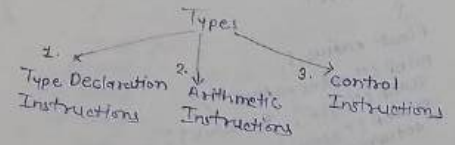
```
#include <stdio.h>
// area of square
int main()
{
float radius;
printf("enter radius");
scanf("%f", &radius);
printf("area is : %f", 3.14 * radius * radius);
return 0;
}
```

enter radius 3.

output : area is : 28.260000

* Chapter 2 * → Instructions

Instructions :- These are statements in a program.
There types of instructions.



Type Declaration Instructions :- Declare variable before using it.

VALID

```
int a = 22;
int b = a;
int c = b+1;
int d = 1, e;

int a, b, c;
a = b = c = 1;
```

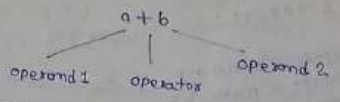
Program:

```
#include <stdio.h>
int main()
{
int a = 22;
int b = a;
int c = b * 6;
int d = 1, e;
return 0;
}
```

Program:

```
#include<stdio.h>
int main()
{
int oldAge = 22;
int years = 2;
int newAge = oldAge+years;
int x, y, z;
x = y = z = 4;
return 0;
}
```

Arithmetic Instructions :-



NOTE → Single variable on the LHS.

(LHS Var = RHS a+b-c*d) (Single variable)

Arithmetic valid instructions (+, -, *, /, % → Modular)

* a = b+c
* a = b*c
* a = b/c

Program:

```
#include <stdio.h>
#include <math.h>
int main()
{
int b, c;
b = c = 1;
int a = b+c;
int powers = pow(b,c);
printf("%d", powers);
return 0;
}
```

output: 1.

* **Arithmetic Instructions** :-

Type conversion:

int → 2 (size)
float → 4 (size)

int op int → int
int op float → float
float op float → float

* Practice Qs. 3

Solve:
int a = 1.999999;

Program:

```
# include <stdio.h>
# include <math.h>
int main()
{
int a = 1.999999;
printf("%d \n", a);
return 0;
}
```

output = 1

* Arithmetic Instructions

Operator Precedence

① *, /, %
↓
② +, -
↓
③ =

For Example ①

x = 4 + 9 * 10

Sol: x = 4 + 90

x = 94 Ans

Program:

```
include <stdio.h>
include <math.h>
int main() {
int a = 4 + 9 * 10;
printf("%d \n", a);
return 0;
}
```

output → 94.

Example 2: x = 4 * 3/6 * 2 (Associativity)

Rules → Left to Right.

Sol: x = ((4*3)/6) * 2
= (12/6) * 2
= 2 * 2
= 4 Ans

Program:

```
int main() {
int a = 4 * 3 / 6 * 2;
printf("%d \n", a);
return 0;
}
```

output → 4

Practice Qs. 4 *

1. 5 * 2 - 2 * 3

Sol: 10 - 6 (Left to Right)
= 4

Program:

```
#include <stdio.h>
#include <math.h>
int main() {
int a = 5 * 2 - 2 * 3;
printf("%d \n", a);
return 0;
}
```

output → 4

2. 5 * 2/2 * 3

Sol: (L→R)
= (5 * 2/2) * 3
= (10/2) * 3
= 5 * 3
= 15.

Program:

```
#include <stdio.h>
#include <math.h>
int main() {
int a = 5 * 2 / 2 * 3;
printf("%d \n", a);
return 0;
}
```

output : 15.

3. 5 * (2\2) * 3

Sol: 5 * (1) * 3
= 5 * 3
= 15

Program:

```
#include <stdio.h>
#include <math.h>
int main()
{
int a = 5 * (2/2) * 3;
printf("%d \n", a);
return 0;
}
```

output : - 15.

4. 5 + 2/2 * 3

Sol: 5 + ((2/2)) * 3
= 5 + (1 * 3)
= 5 + 3
= 8

Program:

```
#include <stdio.h>
#include <math.h>
int main()
{
int a = 5 + 2/2 * 3;
printf("%d \n", a);
return 0;
}
```

Instructions → Control Instructions: - used to determine flow of program.

a. Sequence Control
b. Decision Control.
c. Loop Control
d. Case control.

* Operators *

a. Arithmetic Operators → +, -, \, *, %
b. Relational Operators →
c. Logical Operators
d. Bitwise Operators
e. Assignment operators
f. Ternary Operators

Note: → True → 1
False → 0

* Relational operators: (= =, >, >=, <, <=, !=) (!= → not equal)

Program:

```
#include <stdio.h>
#include <math.h>
int main()
{
printf("%d \n", 4 == 4);
return 0;
}
```

output = 1 (Statement True)

→ change 4 == 3); output = 0 false

* Logical operators: -

&& → AND
|| → OR
! → NOT

&& → Table

1	2	output
T	T	T
T	F	F
F	T	F
F	F	F

&& → Program -

```
#include <stdio.h>
#include <math.h>
int main() {
printf("%d \n", 4>3 && 5>2);
return 0;
}
```

output → 1 (T)

|| → Program:

```
#include <stdio.h>
#include <math.h>
int main()
{
printf("%d \n", 3>4 || 5>3);
return 0;
```

output : 1

|| → Table

1	2	output
T	T	T
T	F	T
F	T	T
F	F	F

! → Table

1	output
T	F
F	T

→ Program: printf("%d \n", !(5>3) && (4>3)); output : 1

* Operator Precedence:

Priority	operator
1.	!
2	*, /, %
3.	+, -
4.	<, <=, >, >=
5.	= =, !=
6.	&&
7.	||
8.	=

* operators

Assignment operators:

= → (a = b)
+= → (a = a+b) or (a += b)
-=
*=
/=

Program:

```
#include <stdio.h>
#include <math.h>
int main()
{
int a = 1;
int b = 4;
a += b;
printf("%d \n", a);
return 0;
}
```

output = 5

Practice Qs: 5

ⓐ Write a program to check if a number is divisible by 2 or not.

Sol:

```
#include <stdio.h>
#include <math.h>
int main() {
int x;
printf("enter a number: ");
scanf("%d", &x);
printf("%d", x%2 == 0);
return 0;
}
```

enter : 10
output : 1

ⓑ Write a program to check if a number is [redacted] odd or even.

Sol:

```
#include <stdio.h>
#include <math.h>
int main() {
(// even -> 1
// odd -> 0)
int x;
printf("enter a number: ");
scanf("%d", &x);
printf("%d", x%2 == 0);
return 0;
}
```

enter : 6
output : 1

enter : 5
output : 0

Practice Qs 7

Are the following valid or not

a. int a = 8^8;

Program

```
int main ( )
{
printf ("%d", 8^8);
return 0;
}
```

Valid → output = 0 Ans

b. int x; int y = x;

```
int main()
{
int x; int y = x;
return 0;
}
```

Valid = 0

c. int x, y = x;

Ans = not valid

d. char star = '*';

Ans = not valid

Practice Qs 8

Print 1 (true) or (false) for following statements

a. If it's Sunday & it's Snowing → true

Program

```
#include <stdio.h>
int main ( )
{
int isSunday = 1;
int isSnowing = 1;
printf ("%d \n", isSunday && isSnowing);
return 0;
}
```

output = 1

b. If it's monday or it's raining → true
// logical or

Sol

```
#include <stdio.h>
int main ( )
{
int isMonday = 0;
int isRaining = 1;
printf (" %d \n", isMonday || isRaining);
return 0;
}
```

output = 1 → Run → gcc fileName / ./a.out

c. If a number is greater than 9 & less than 100 → true (2 digit number)

Program (x>9) && (x<100)

```
#include <stdio.h>
#include <math.h>
int main()
{
int x;
printf (" enter number : ");
scanf (" %d", &x);
printf ("%d \n", x>9 && x<100);
return 0;
}
```

Output: enter number : 78
Ans = 1 (true)

enter number : 101
Ans = 0 (false)

* Chapter - 3 - Conditional Statements *

Conditional Statements
Types
if-else → Switch

Statement:

```
if (condition) {
   // do something if TRUE
}
else {
   // do something if False
}
```

Program:

```
#include<stdio.h>
int main() {
int age;
printf (" enter age : ");
scanf (" %d", &age);
if (age > 18) {             ← condition
printf (" adult \n");
printf (" they can vote \n");
printf (" they can drive \n");
}
else {
printf (" not adult \n");
}
return 0; }
```

output: enter age : 22
adult
they can vote
they can drive

else if

```
if (condition 1) {
   // do something if TRUE
}
else if (condition 2) {
   // do something if 1st is FALSE & 2nd is TRUE
}
```

Note condition:

if
else if
else if
else ↓

→ Example

Program:

```
#include <stdio.h>
int main()
{
int age;
printf (" enter age : ");
scanf (" %d", &age);
if (age >= 18)
{
printf (" adult \n");
}
else if (age > 13 && age < 18)
{
printf (" teenager \n");
}
```

Run
Output - Terminal : gcc file name
./a.out

enter age : 5
child

enter age : 16
teenager

enter age : 22
adult.

Conditional Operators

Ternary → Condition ? doSomething if TRUE : do Something if FALSE;

Program:

```
#include<stdio.h>
int main ()
{
int age;
printf (" enter age : ");
scanf (" %d", &age);
age >= 18 ? printf (" adult \n") : printf (" not adult \n");
return 0;
}
```

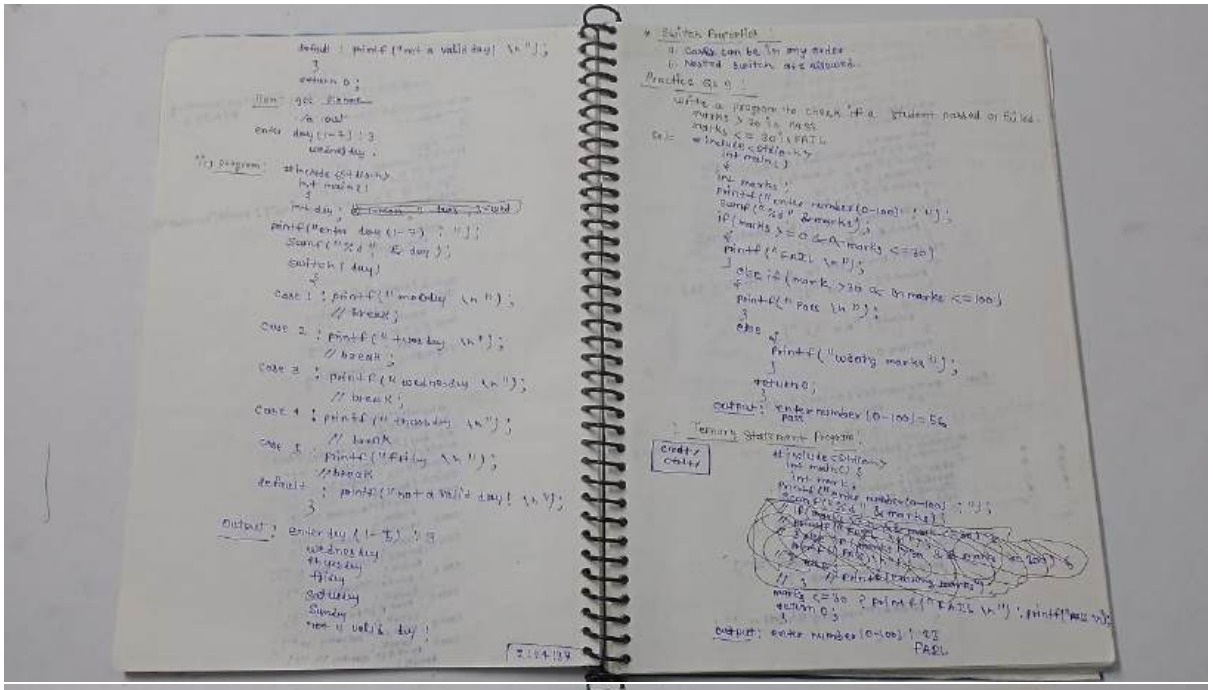
Output: enter age : 15
not adult

Switch

```
switch (number) {
case C1 : // do something
     break;
case C2 : // do something
     break;
default : // do something
}
```

Program:

```
#include<stdio.h>
int main() {
int day; // 1-monday, 2-tuesday, 7-sunday
printf (" enter day (1-7) : ");
scanf (" %d", &day);
switch (day) {
case 1 : printf (" monday \n");
     break;
case 2 : printf (" tuesday \n");
     break;
case 3 : printf (" wednesday \n");
     break;
case 4 : printf (" thursday \n");
     break;
case 5 : printf (" friday \n");
     break;
case 6 : printf (" saturday \n");
     break;
case 7 : printf (" sunday \n");
     break;
```



default : printf("not a valid day! \n");
}
return 0;
}
Run: gcc filename.c
./a.out
enter day (1-7) : 3
wednesday.

1) program:
#include <stdio.h>
int main()
{
int day; (1-mon 2-tue 3-wed)
printf("enter day (1-7) : ");
scanf("%d", &day);
switch (day)
{
case 1 : printf("monday \n");
// break;
case 2 : printf("tuesday \n");
// break;
case 3 : printf("wednesday \n");
// break;
case 4 : printf("thursday \n");
// break;
case 5 : printf("friday \n");
//break;
default : printf("not a valid day! \n");
}
Output: enter day (1-7) : 3
wednesday
thursday
friday
Saturday
Sunday
not a valid day!

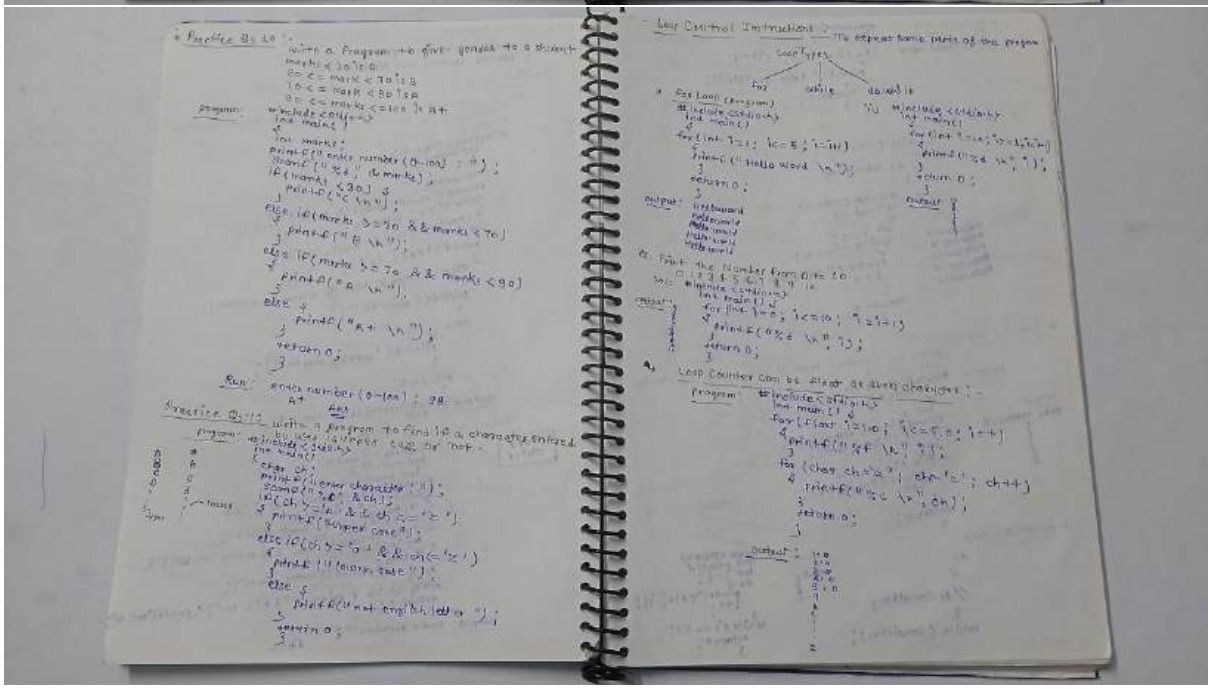
* Switch Parallel :
1. Cases can be in any order
2. Nested switch are allowed.

Practice Qs 9 :
Write a program to check if a student passed or failed.
marks > 30 is pass
marks <= 30 is FAIL
Ans: #include<stdio.h>
int main()
{
int marks;
printf("enter number (0-100) : ");
scanf("%d", &marks);
if(marks >= 0 && marks <= 30)
{
printf("FAIL \n");
}
else if(mark >30 && marks <= 100)
{
Printf("n Pass \n");
}
else
{
Printf("wrong marks");
}
return 0;
}
output: enter number (0-100) = 56
Pass

Binary Statement Program

Credit / Caller

#include <stdio.h>
int main() {
int mark;
Printf("enter number (0-100) : ");
scanf("%d", &marks);
if (marks >= 0 && marks <= 30)
Printf("FAIL \n");
else if (marks >30 && marks <= 100)
Printf("Pass \n");
else
Printf("wrong marks");
return 0;
}
output: enter number (0-100) : 98
PASS



* Practice Qs 10 :
write a Program to give grades to a student
marks < 30 is D
30 <= mark < 70 is B
70 <= mark < 80 is A
80 <= marks <= 100 is A+
Program: #include <stdio.h>
int main()
{
int marks;
printf("enter number (0-100) : ");
scanf("%d", &marks);
if(marks <30)
{
printf("D \n");
}
else if(marks >=30 && marks <70)
{
printf("B \n");
}
else if(marks >=70 && marks <90)
{
printf("A \n");
}
else
{
printf("A+ \n");
}
return 0;
}
Run: enter number (0-100) : 98
A+

Practice Q: 11 : write a program to find if a character entered is by user is upper case or not.
Program: #include <stdio.h>
int main()
{
char ch;
printf("enter character : ");
scanf("%c", &ch);
if(ch >='a' && ch <='z')
{
printf("lower case");
}
else if(ch >='A' && ch <='Z')
{
printf("Upper case");
}
else
{
printf("not english letter");
}
return 0;
}

Loop Control Instructions : To repeat some parts of the program

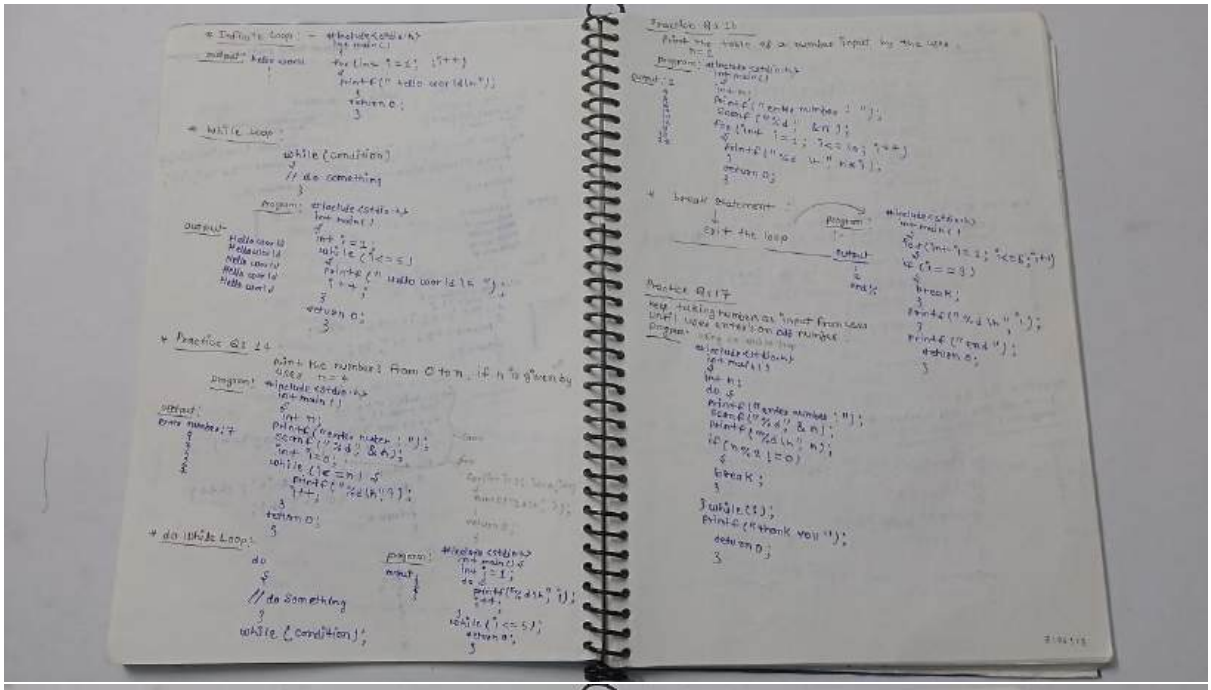
Loop Types: for, while, do-while

* For Loop (program)
#include <stdio.h>
int main()
{
for(int i=1; i<=5; i=i+1)
{
printf("Hello World \n");
}
return 0;
}
output: Hello World (5 times)

#include <stdio.h>
int main()
{
for(int i=1; i<=5; i=i+1)
{
printf("%d ", i);
}
return 0;
}
output: 1 2 3 4 5

Q: Print the Number from 0 to 10
0 1 2 3 4 5 6 7 8 9 10
Ans: #include <stdio.h>
int main() {
for(int i=0; i<=10; i=i+1)
{
printf("%d \n", i);
}
return 0;
}
output: 0 1 2 3 4 5 6 7 8 9 10

Loop Counter can be start as every character :-
Program: #include<stdio.h>
int main()
{
for(float i=1.0; i<=5.0; i++)
{
printf("%f \n", i);
}
for (char ch='a'; ch<='z'; ch++)
{
printf("%c \n", ch);
}
return 0;
}
output: 1.0
2.0
3.0
4.0
5.0
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z



* Infinite Loop :-

```
#include<stdio.h>
int main()
{
for(;;)
{
printf("hello world\n");
}
return 0;
}
```

output:- hello world

* while loop :

```
while (condition)
{
// do something
}
```

Program:

```
#include<stdio.h>
int main()
{
int i=1;
while(i<=5)
{
printf("hello world is %d\n",i);
i++;
}
return 0;
}
```

output:-
Hello world is 1
Hello world is 2
Hello world is 3
Hello world is 4
Hello world is 5

* Practice Q: 14 :-

Print the numbers from 0 to n, if n is given by user. n=4

Program:

```
#include<stdio.h>
int main()
{
int n;
printf("enter number : ");
scanf("%d",&n);
int i=0;
while(i<=n)
{
printf("%d\n",i);
i++;
}
return 0;
}
```

output:
enter number : 7
0 1 2 3 4 5 6 7

* do while Loop:

```
do
{
// do something
}
while (condition);
```

Program:

```
#include<stdio.h>
int main()
{
int i=1;
do
{
printf("%d\n",i);
i++;
}
while(i<=5);
return 0;
}
```

output:
1 2 3 4 5

Practice Q: 15

Print the table of a number input by the user.
n=2

Program:

```
#include<stdio.h>
int main()
{
int n;
printf("enter number : ");
scanf("%d",&n);
for(int i=1;i<=10;i++)
{
printf("%d x %d = %d\n",n,i,n*i);
}
return 0;
}
```

output:
enter number : 2
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20

* break statement :-

exit the loop

Program:

```
#include<stdio.h>
int main()
{
for(int i=1;i<=5;i++)
{
if(i==3)
{
break;
}
printf("%d\n",i);
}
printf("end");
return 0;
}
```

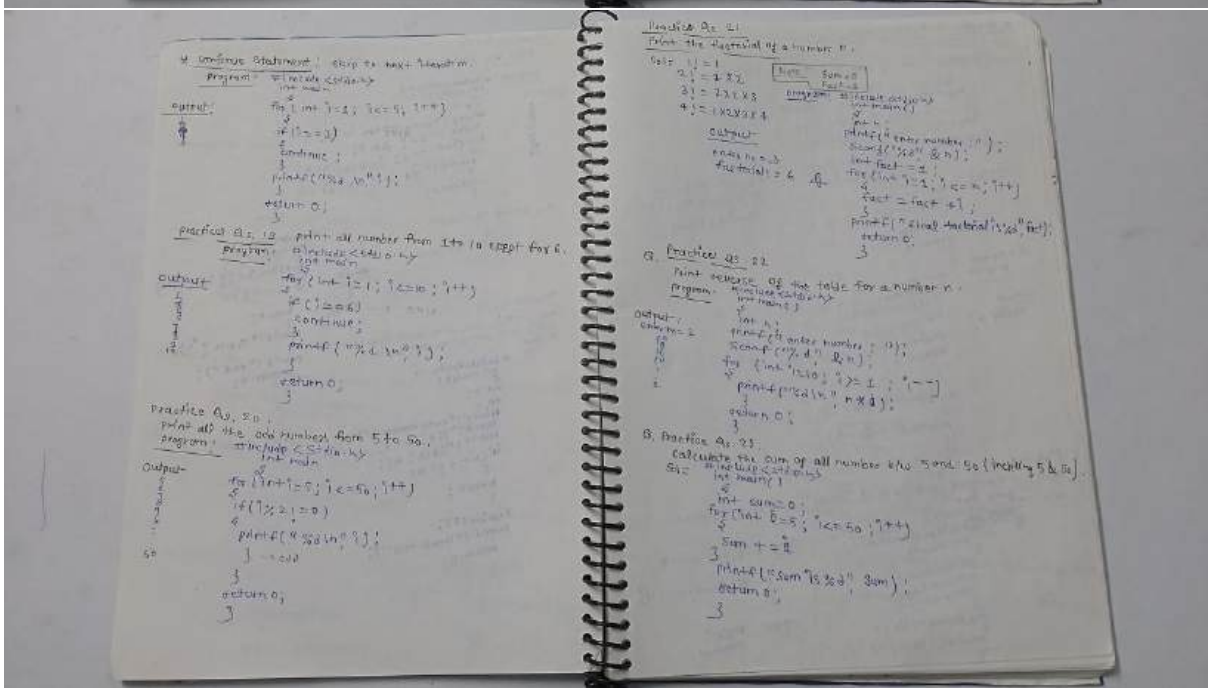
output:
1
2
end

Practice Q: 17

Keep taking numbers as input from user until user enters an odd number.

Program:

```
#include<stdio.h>
int main()
{
int n;
do
{
printf("enter number : ");
scanf("%d",&n);
printf("%d\n",n);
if(n%2!=0)
{
break;
}
}while(1);
printf("thank you");
return 0;
}
```



* continue Statement : skip to next iteration.

Program:

```
#include<stdio.h>
int main()
{
for(int i=1;i<=5;i++)
{
if(i==3)
{
continue;
}
printf("%d\n",i);
}
return 0;
}
```

output:
1
2
4
5

Practice Q: 18 : print all number from 1 to 10 except for 6.

Program:

```
#include<stdio.h>
int main
{
for(int i=1;i<=10;i++)
{
if(i==6)
{
continue;
}
printf("%d\n",i);
}
return 0;
}
```

output:
1
2
3
4
5
7
8
9
10

Practice Q: 19 : print all the odd numbers from 5 to 50.

Program:

```
#include<stdio.h>
int main
{
for(int i=5;i<=50;i++)
{
if(i%2!=0)
{
printf("%d\n",i);
}
}
return 0;
}
```

Output:
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
43
45
47
49

Practice Q: 21

Print the factorial of a number n.

Sol: 1! = 1
2! = 1 x 2
3! = 1 x 2 x 3
4! = 1 x 2 x 3 x 4

Note: Sum = 0, Fact = 1

Program:

```
#include<stdio.h>
int main()
{
int n;
printf("enter number : ");
scanf("%d",&n);
int fact=1;
for(int i=1;i<=n;i++)
{
fact=fact*i;
}
printf("final factorial is %d",fact);
return 0;
}
```

output:
enter no. : 3
factorial : 6

Q. Practice Qs: 22

Print reverse of the table for a number n.

Program:

```
#include<stdio.h>
int main()
{
int n;
printf("enter number : ");
scanf("%d",&n);
for(int i=10;i>=1;i--)
{
printf("%d x %d\n",n,i);
}
return 0;
}
```

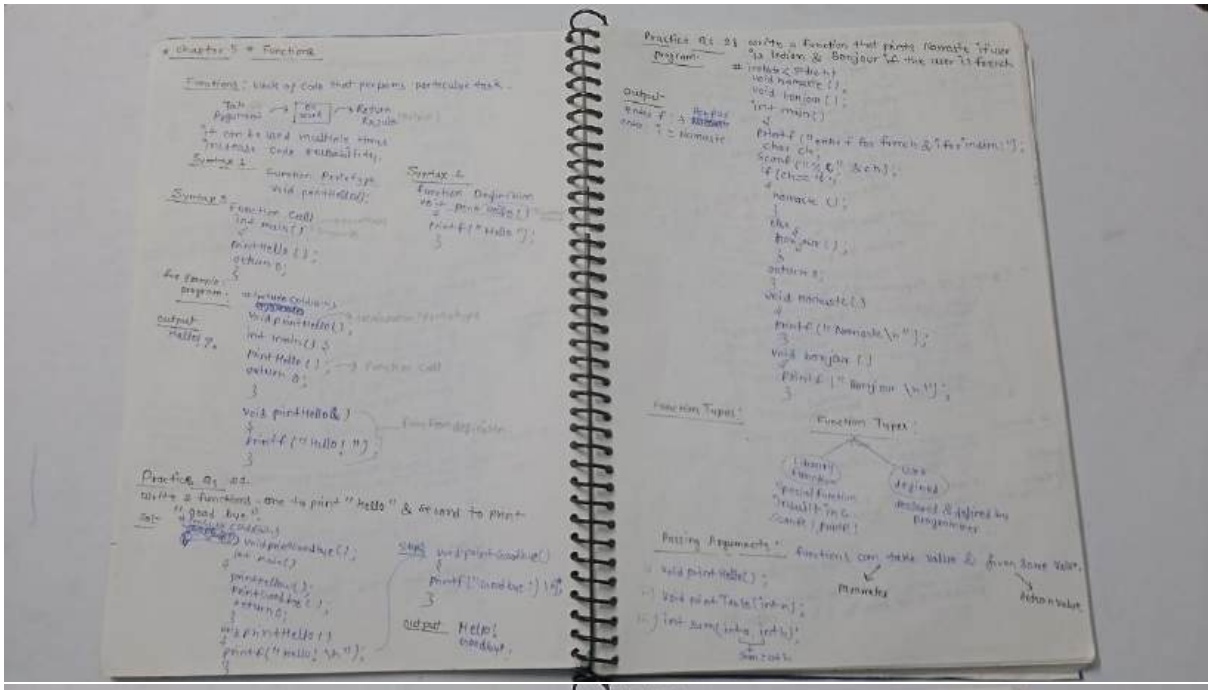
output:
enter no. : 2
2 x 10
2 x 9
2 x 8
2 x 7
2 x 6
2 x 5
2 x 4
2 x 3
2 x 2
2 x 1

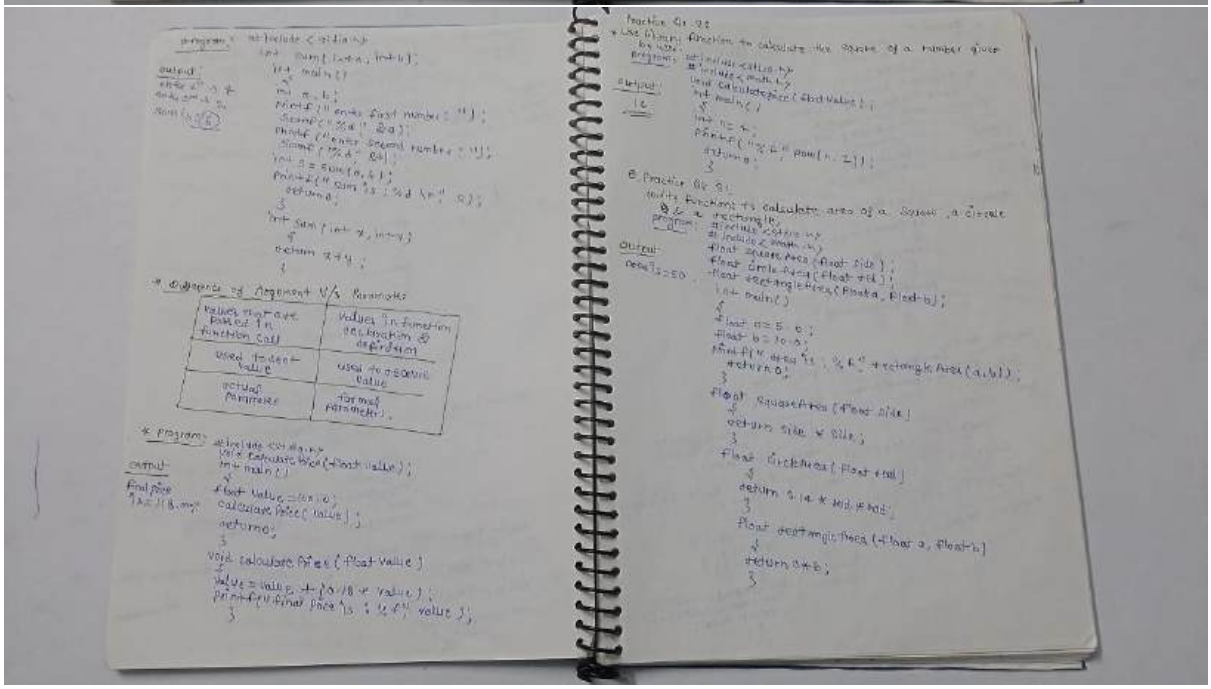
B. Practice Qs: 23

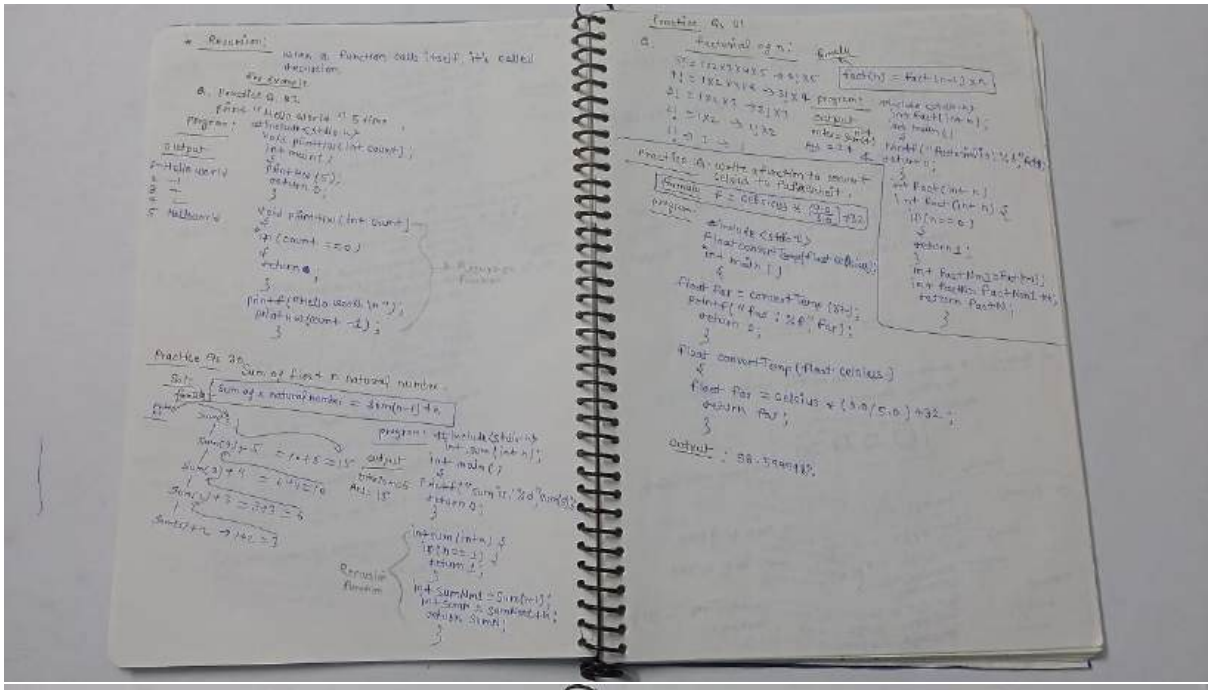
Calculate the sum of all number b/w 5 and 50 (including 5 & 50).

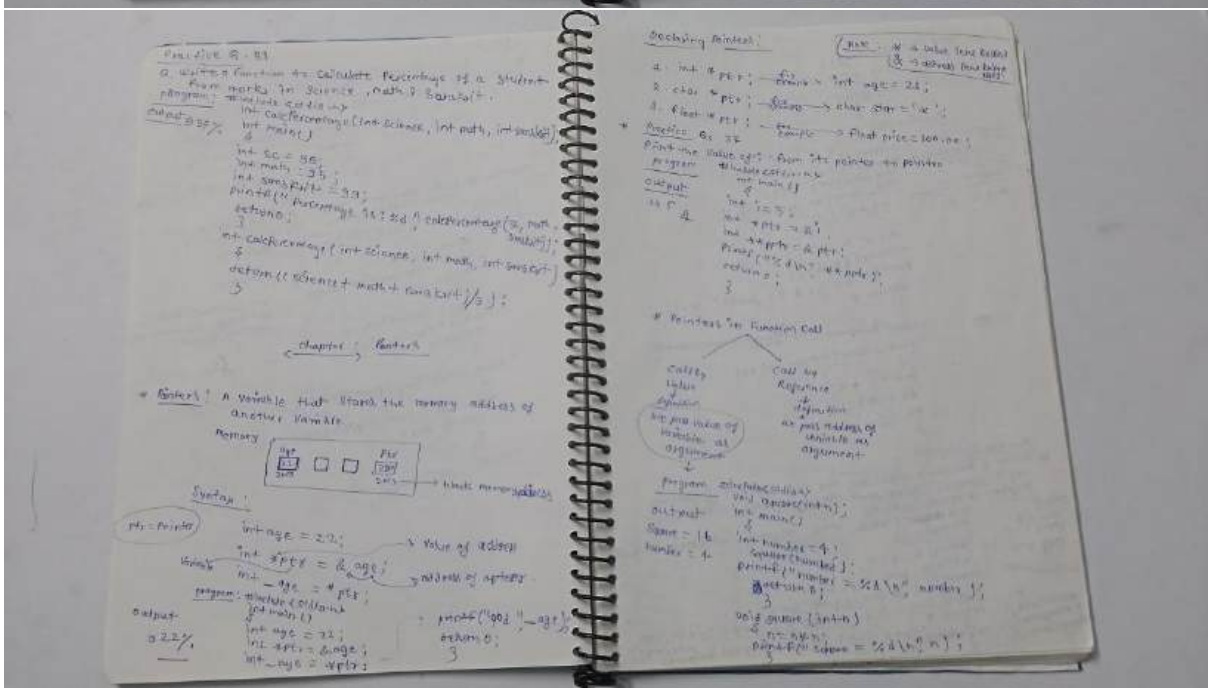
Sol:

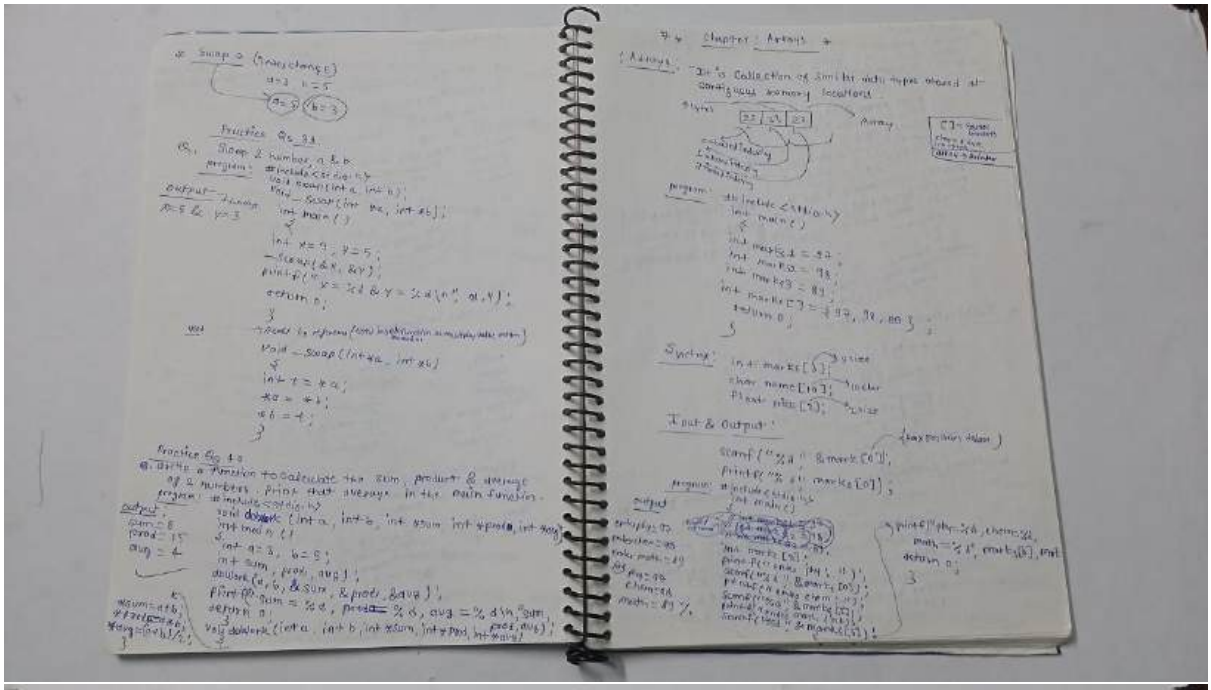
```
#include<stdio.h>
int main()
{
int sum=0;
for(int i=5;i<=50;i++)
{
sum+=i;
}
printf("sum is %d",sum);
return 0;
}
```

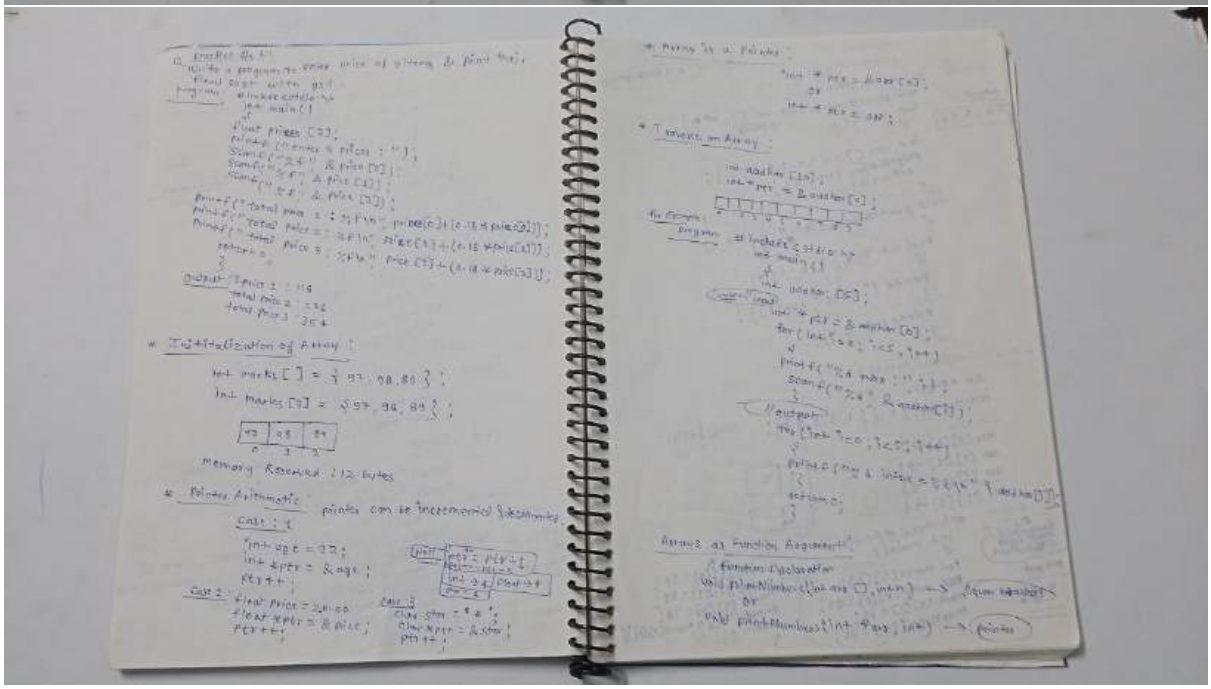












```c
//function call
printNumbers(arr, n);
```

Note — (P → small letter)

Program:

```c
#include <stdio.h>
void printNumbers(int arr[], int n);
int main()
{
int arr[] = {1, 2, 3, 4, 5, 6};
printNumbers(arr, 6);
return 0;
}
void printNumbers(int *arr, int n)
{
for(int i=0; i<n; i++)
{
printf("%d\t", arr[i]);
}
printf("\n");
}
```

output

1 2 3 4 5 6

Multidimensional Arrays:

2D Arrays

```c
int arr[][] = {{1, 2, 3, 4}}; //Declare
//Access
arr[0][0] -> 1
arr[0][1] -> 2
arr[1][0] -> 3
arr[1][1] -> 4
```

00	01
1	2
10	11

program: (find the child marks in ... mark)

```c
#include <stdio.h>
int main()
{
int marks[2][3];
marks[0][0] = 90;
marks[0][1] = 89;
marks[0][2] = 78;
marks[1][0] = 90;
marks[1][1] = 83;
marks[1][2] = 78;
printf("%d", marks[0][0]);
return 0;
}
```

output

90%

Practice Qs 42

Q. Write a function to count the number of odd numbers in an array.

1	2	3	4	5

Note: c → small letter

program:

```c
#include <stdio.h>
int countOdd(int arr[], int n);
int main()
{
int arr[] = {1, 2, 3, 4, 5, 6};
printf("%d", countOdd(arr, 6));
return 0;
}
int countOdd(int arr[], int n)
{
int count = 0;
for(int i=0; i<n; i++)
{
if(arr[i] % 2 != 0)
{
count++;
}
}
return count;
}
```

output
=> 3%
odd number

practice Qs 43

int arr[] = {1, 2, 3, 4, 5}
For the given array, what will the following give?

a. *(arr+2) b. *(arr+5)

Sol:

1	2	3	4	5

program:

```c
#include <stdio.h>
int countOdd(int arr[], int n);
int main(){
int arr[] = {1, 2, 3, 4, 5};
printf("%d\n", *(arr+2));
printf("%d\n", *(arr+5));
return 0;
}
```

output
a = 3
b = 0

→ Same code

Practice Qs 44

Q. Write a function to reverse an array.

Sol:

1	2	3	4

⇔

4	3	2	1

or

1	2	3	4	5

→ Example

Reverse 5 4 3 2 1

program:

```c
#include <stdio.h>
int countOdd(int arr[], int n);
void reverse(int arr[], int n);
void printArr(int arr[], int n);
int main()
{
int arr[] = {1, 2, 3, 4, 5};
reverse(arr, 5);
printArr(arr, 5);
return 0;
}
void printArr(int arr[], int n)
{
for(int i=0; i<n; i++)
{
printf("%d\t", arr[i]);
}
printf("\n");
}
void reverse(int arr[], int n)
{
for(int i=0; i<n/2; i++)
{
int firstVal = arr[i];
int secondVal = arr[n-i-1];
arr[i] = secondVal;
arr[n-i-1] = firstVal;
}
}
```

output
5 4 3 2 1

Practice Qs 45

Write a program to store the first n fibonacci numbers.

Sol: fibonacci number ⇒ 0, 1, 1, 2, 3, 5, 8, 13, 21 ...

Note: nth = (n-1) + (n-2)

0+1=1
1+1=2
1+2=3
2+3=5

program:

```c
#include <stdio.h>
int main()
{
int n;
printf("enter n (n>2): ");
scanf("%d", &n);
int fib[n];
fib[0] = 0;
fib[1] = 1;
for(int i=2; i<n; i++)
{
fib[i] = fib[i-1] + fib[i-2];
printf("%d \t", fib[i]);
}
printf("\n");
return 0;
}
```

output → 10
1 2 3 5 8 13 21 34

Practice Qs 46

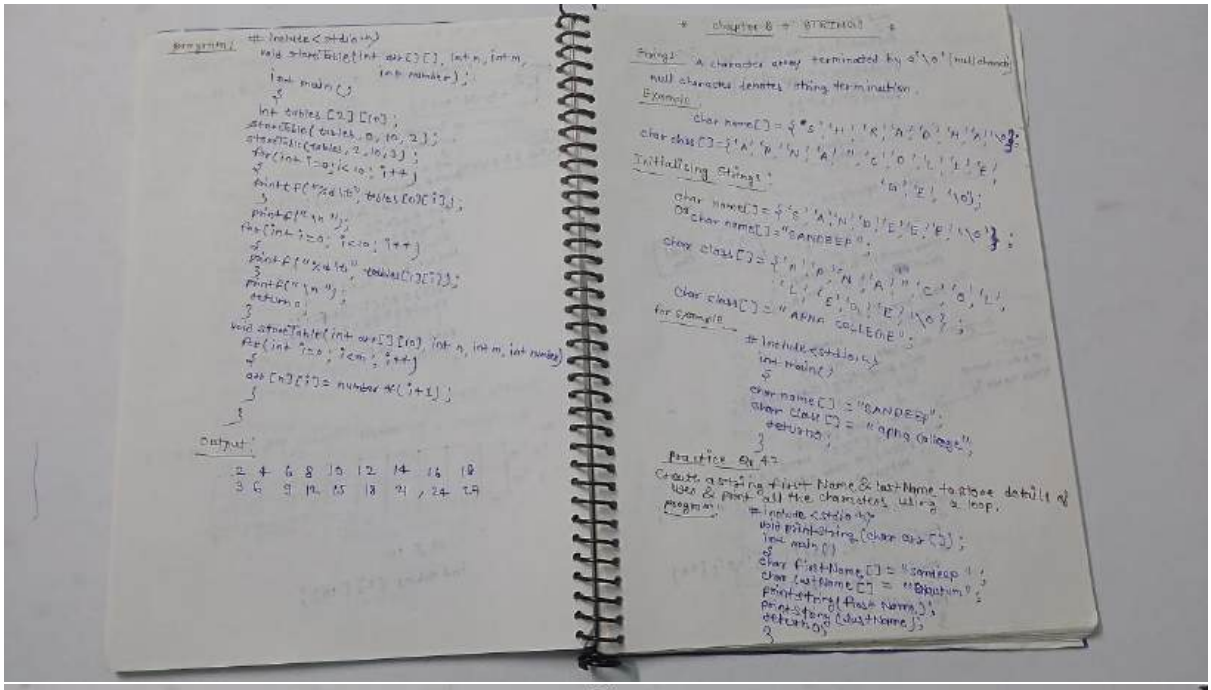
Q. Create a 2D array, storing the tables of 2 & 3.

Sol:

	0	1	2	3	4	5	6	7	8	9
"2" →										
"3" →										

2 X 10

int tables [2] [10];



```
Program:
#include <stdio.h>
void storeTable(int arr[][10], int n, int m, int number);
int main()
{
int table[2][10];
storeTable(table, 0, 10, 2);
storeTable(table, 1, 10, 3);
for(int i=0;i<10;i++)
{
printf("%d\t", table[0][i]);
}
printf("\n");
for(int i=0; i<10; i++)
{
printf("%d\t", table[1][i]);
}
printf("\n");
return 0;
}
void storeTable(int arr[][10], int n, int m, int number)
{
for(int i=0; i<m; i++)
{
arr[n][i] = number * (i+1);
}
}
```

Output:

```
2   4   6   8   10  12  14  16  18
3   6   9   12  15  18  21  24  27
```

* Chapter 8 → STRINGS *

String: A character array terminated by a '\0' (null character). The null character denotes string termination.

Example:

```
char name[] = {'S', 'A', 'N', 'D', 'E', 'E', 'P', '\0'};
char class[] = {'A', 'P', 'N', 'A', ' ', 'C', 'O', 'L', 'L', 'E', 'G', 'E', '\0'};
```

Initializing strings:

```
char name[] = {'S', 'A', 'N', 'D', 'E', 'E', 'P', '\0'};
Or char name[] = "SANDEEP";
char class[] = {'A', 'P', 'N', 'A', ' ', 'C', 'O', 'L', 'L', 'E', 'G', 'E', '\0'};
Or char class[] = "APNA COLLEGE";
```

for example

```
#include <stdio.h>
int main()
{
char name[] = "SANDEEP";
char class[] = "apna college";
return 0;
}
```

Practice Qs 47

Create a string of first Name & last Name to store details of user & print all the characters using a loop.

Program:

```
#include <stdio.h>
void printString(char str[]);
int main()
{
char firstName[] = "sandeep";
char lastName[] = "Bhatia";
printString(firstName);
printString(lastName);
return 0;
}
```

```
void printString(char str[])
{
for(int i=0; str[i]!='\0'; i++)
{
printf("%c", str[i]);
}
printf("\n");
}
```

Output:

```
Sandeep
Bhatia
```

* String Format Specifiers *

"%s" → string

'\0' → null character

```
char name[] = "Sandeep";
printf("%s", name);
Or
char name[40];
scanf("%s", name);
```

Program:

```
#include <stdio.h>
void printString(char str[]);
int main()
{
char name[50];
scanf("%s", name);
printf("Hey, name is %s", name);
return 0;
}
void printString(char str[])
{
for(int i=0; str[i]!='\0'; i++)
{
printf("%c", str[i]);
}
printf("\n");
}
```

Output:

```
Enter name
Your name is
```

* Practice Qs 48 *

Make a program that inputs user's name & prints its length.

Sol:

```
SANDEEP\0
1 2 3 4 5 6 7 8
length = 7
```

```
char name[100];
fgets(name, 100, stdin);
```

Note: 65 → Capital letters, 97 → Small letters

Program:

```
#include <stdio.h>
void printString(char str[]);
int countLength(char str[]);
int main()
{
char name[100];
fgets(name, 100, stdin);
printf("Length is: %d", countLength(name));
return 0;
}
int countLength(char str[])
{
int count = 0;
for(int i=0; str[i]!='\0'; i++)
{
count++;
}
return count-1;
}
void printString(char str[])
{
for(int i=0; str[i]!='\0'; i++)
{
printf("%c", str[i]);
}
printf("\n");
}
```

Output: write SANDEEP

length = 7

Practice Qs 51

* Salting

Q. Find the salted form of a password entered by user if the salt is "123" & added at the end.

Sol: Salting →

Password = "test"
Salt = "123"
New Password = "test123" → let

Program:

```
#include<stdio.h>
#include<string.h>
void printString(char arr[]);
int countLength(char arr[]);
void salting(char password[]);
int main()
{
char password[100];
scanf("%s", password);
salting(password);
}
void salting(char password[])
{
char salt[] = "123";
char newPass[200];
strcpy(newPass, password);
strcat(newPass, salt);
puts(newPass);
}
int countLength(char arr[])
{
int count = 0;
for(int i=0; arr[i]!='\0'; i++)
{
count++;
}
return count-1;
}
```

P → capital letter

Practice Qs 53

Write a function to count the occurrence of vowels in a string.

Sol: str = Helloworld   vowels = a, e, i, o, u
3 vowels

Program:

output: 3 vowels

```
#include<stdio.h>
#include<string.h>
int countVowels(char str[]);
int main()
{
char str[] = "Sandeep";
printf("vowels are : %d", countVowels(str));
}
int countVowels(char str[])
{
int count = 0;
for(int i=0; str[i] != '\0'; i++)
if(str[i]=='a' || str[i]=='e' || str[i]=='i' ||
        str[i]=='o' || str[i]=='u')
{
count++;
}
}
return count;
}
```

|| → or
S → small letter
C → capital

Practice Qs 54.

Q. check if a given character is present in a string or not;

Sol:- str = "Helloworld"
char ch = "w"   Y or N
char ch = "L"   Yes
ch = "x"   No.   No

program:

```
#include<stdio.h>
#include<string.h>
void checkChar(char str[], char ch);
int main()
{
char str[] = "ApnaCollege";
char ch = 'x';
checkChar(str, ch);
}
void checkChar(char str[], char ch)
{
for(int i=0; str[i] != '\0'; i++)
if(str[i] == ch)
{
printf("character is present !");
return;
}
}
printf("character is Not present :(");
}
```

Output → Apna College → present.

* Chapter → 9 → Structures

Structures : a collection of values of different data type.

Example : For a student store the following:
name (String)
roll no (Integer)
cgpa (Float)
} → structure

Note: S = small letter

Note → Structure is a user define data type.

Syntax :

```
struct student {
char name[100];
int roll;
float cgpa;
};
```

declare function for example
struct student s1;
s1.cgpa = 7.5;
s1.name = [illegible]
s1.roll number
Note: s1 → statement [illegible]

for Example

Program : Student marks

```
#include<stdio.h>
#include<string.h>
struct student
{
int roll;
float cgpa;
char name[100];
};
int main()
{
struct student s1;
s1.roll = 1664;
s1.cgpa = 9.2;
strcpy(s1.name, "Sandeep");
printf("student name = %s\n", s1.name);
printf("student roll no = %d\n", s1.roll);
printf("student cgpa = %f\n", s1.cgpa);
return 0;
}
```

Practice Qs 57 | Note: S → Small letter

* Write a program to store the data of 3 students.

Program:
```
#include <stdio.h>
#include <string.h>
struct student
{
int roll;
float cgpa;
char name[100];
};
int main()
{
struct student S1;
S1.roll = 1664;
S1.cgpa = 9.2;
strcpy(S1.name, "sandeep");
printf("student name = %s\n", S1.name);
printf("student roll no = %d\n", S1.roll);
printf("student cgpa = %f\n", S1.cgpa);
struct student S2;
S2.roll = 1662;
S2.cgpa = 9.4;
strcpy(S2.name, "Shardha");
printf("student name = %s\n", S2.name);
printf("student roll no = %d\n", S2.roll);
printf("student cgpa = %f\n", S2.cgpa);
struct student S3;
S3.roll = 1661;
S3.cgpa = 7.4;
strcpy(S3.name, "Aman");
printf("student name = %s\n", S3.name);
printf("student roll no = %d\n", S3.roll);
printf("student cgpa = %f\n", S3.cgpa);
return 0;
}
```
(user defined structure)

8 : 41 : 18

* Array of structures :-
```
struct student ECE[100];
struct student COE[100];
struct student IT[100];
```
* Access :-
```
IT[0].roll = 200;
IT[0].cgpa = 7.6;
```
For example: program
```
#include <stdio.h>
#include <string.h>
struct student
{
int roll;
float cgpa;
char name[100];
};
int main()
{
struct student ece[100];
ece[0].roll = 1664;
ece[0].cgpa = 9.2;
strcpy(ece[0].name, "sandeep");
printf("name = %s\n", ece[0].name);
return 0;
}
```
Initializing Structures :
```
struct student S1 = {"sandeep", 1664, 7.9};
struct student S2 = {"rajat", 1552, 8.3};
struct student S3 = {0};
```
Program:
```
#include <stdio.h>
#include <string.h>
struct student
{
int roll;
float cgpa;
char name[100];
};
int main()
{
struct student S1 = {1664, 9.2, "Sandeep"};
printf("student roll = %d\n", S1.roll);
return 0;
}
```

* Pointers to Structures :
```
struct student S1;
struct student *ptr;
ptr = &S1;
```
Program:
```
#include <stdio.h>
#include <string.h>
struct student
{
int roll;
float cgpa;
char name[100];
};
int main()
{
struct student S1 = {1664, 9.2, "sandeep"};
printf("student roll = %d\n", S1.roll);
struct student *ptr = &S1;
printf("student roll = %d\n", (*ptr).roll);
return 0;
}
```
output: Student roll = 1664
Student roll = 1664

* Chapter 10 : File Input/Output *

File I/O :-

RAM (Volatile) ⇄ Hard Disk (non-volatile)

File → container in a storage device to store data.
- RAM → RAM is volatile memory. Contents are lost when program terminates. Files are used to persist the data.

Types of Files :- 2 Type
- Text Files: .txt, .c
- Binary Files: (examples) .exe, .mp3, .jpg

File Pointer: File is a (hidden) structure that needs to be created for opening a file. A file ptr that points to this structure & is used to access the file.

FILE *fptr (pointer)

* Opening a File :
```
FILE *fptr;
fptr = fopen("filename", mode);
```
→ file function

* Closing a File:
```
fclose(fptr);
```
* File Opening Modes :
- "r" → open to read
- "rb" → open to read in binary

"w" → Open to write

"wb" → Open to write in binary

"a" → Open to append

Program:

check if a file exists before reading from it.

Program:

```
#include <stdio.h>
int main()
{
FILE *fptr;
fptr = fopen("NewTest.txt", "w");
if(fptr == NULL)
{
printf("File doesn't exist\n");
}
else
{
fclose(fptr);
}
return 0;
}
```

Output — Creating New File → NewTest.txt

* **Reading from a file:**

```
char ch;   or   int ch;
fscanf(fptr, "%c", &ch);   fscanf(fptr, "%d", &ch);
```

File create before: Hello.c → Test.txt → Apple

Program write

```
#include <stdio.h>
int main()
{
FILE *fptr;
fptr = fopen("Test.txt", "r");
char ch;
fscanf(fptr, "%c", &ch);
printf("character = %c\n", ch);
fclose(fptr);
return 0;
}
```

output → character = A

or

Ctrl+Shift copy paste

write

output → character = A

P

P

L

* **Writing to a file:**

```
char ch = 'A';
fprintf(fptr, "%c", ch);
```

file → Hello.c → Test.txt → write Apple change (Mango) → Output

Program:

```
#include <stdio.h>
int main()
{
FILE *fptr;
fptr = fopen("Test.txt", "w");
fprintf(fptr, "%c", 'M');
fprintf(fptr, "%c", 'A');
fprintf(fptr, "%c", 'N');
fprintf(fptr, "%c", 'G');
fprintf(fptr, "%c", 'O');
fclose(fptr);
return 0;
}
```

Output → MANGO

* **Read & Write a char:**

fgetc(fptr) → char Read (used)

fputc('A', fptr) → char file to write (used)

Practice Qs 61

Make a program to read 5 integers from a file.

Program — file create: Hello.c → Test.txt → 1 2 3 4 5

```
#include <stdio.h>
int main()
{
FILE *fptr;
fptr = fopen("test.txt", "r");
int n;
fscanf(fptr, "%d", &n);
printf("number = %d\n", n);
fscanf(fptr, "%d", &n);
printf("number = %d\n", n);
fscanf(fptr, "%d", &n);
printf("number = %d\n", n);
fscanf(fptr, "%d", &n);
printf("number = %d\n", n);
fscanf(fptr, "%d", &n);
printf("number = %d\n", n);
fclose(fptr);
return 0;
}
```

Practice Qs 62

* Make a program to input student information from a user & enter it to a file.

Program:

```
#include <stdio.h>
int main()
{
FILE *fptr;
fptr = fopen("Student.txt", "w");
char name[100];
int age;
float cgpa;
printf("enter name : ");
scanf("%s", name);
printf("enter age : ");
scanf("%d", &age);
printf("enter cgpa : ");
scanf("%f", &cgpa);
fprintf(fptr, "student name : %s\n", name);
fprintf(fptr, "student age : %d\n", age);
fprintf(fptr, "student cgpa : %f", cgpa);
fclose(fptr);
return 0;
}
```

(file open in the code) (file name.txt) (file close in the code)

output

enter name = Sandeep

enter age = 21

enter cgpa = 7.2

Run ↓ file create → output

Student.txt

1. Student name : Sandeep
2. Student age : 21
3. Student cgpa : 7.2

Practice Q. 63

1. Write a program to write all the odd numbers from 1 to (n) in a file.

Program:

```
#include<stdio.h>
int main()
{
FILE *fptr;
fptr = fopen("odd.txt", "w");
int n;
printf("enter n: ");
scanf("%d", &n);
for(int i=1; i<=n; i++)
{
if(i%2 != 0)
{
fprintf(fptr, "%d\n", i);
}
}
```

Output: enter n: 10

Practice Q. 64:

Q. 2 numbers - a & b, are written in a file, write a program to replace them with their sum.

Sol:

Program: Step-1 → File create Name Sum.txt

```
Program: write
#include<stdio.h>
int main()
{
FILE *fptr;
fptr = fopen("sum.txt", "w");
int a;
fscanf(fptr, "%d", &a);
int b;
fscanf(fptr, "%d", &b);
fclose(fptr);
fptr = fopen("sum.txt", "w");
fprintf(fptr, "%d", a+b);
fclose(fptr);
return 0;
}
```

Run → Sum.txt

output = 5

Chapter - 11

* Dynamic Memory Allocation *

* Dynamic Memory Allocation: It is a way to allocate memory to a data structure during the runtime.

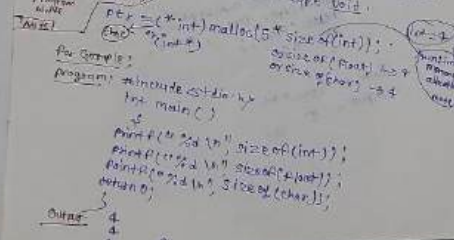
We need some function to allocate & free memory dynamically.

Functions for DMA:

a. malloc() → memory allocation
b. calloc() → continuous
c. free()
d. realloc() → re-allocation

* malloc() (memory allocation):

takes number of bytes to be allocated & returns a pointer of type void.



For Example:

Program:

```
#include<stdio.h>
int main()
{
printf("%d\n", sizeof(int));
printf("%d\n", sizeof(float));
printf("%d\n", sizeof(char));
return 0;
}
```

Output: 4 4 1

Program:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
int *ptr;
ptr = (int*)malloc(5*sizeof(int));
ptr[0] = 1;
ptr[1] = 3;
ptr[2] = 5;
ptr[3] = 7;
ptr[4] = 9;
for(int i=0; i<5; i++)
{
printf("%d\n", ptr[i]);
}
return 0;
}
```

output:

```
1
3
5
7
9
```

Practice Q.s. 65:

Q. Write a Program to allocate memory to store 5 prices (float).

Sol:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
float *ptr;
ptr = (float*)malloc(5*sizeof(float));
ptr[0] = 1;
ptr[1] = 3;
ptr[2] = 5;
ptr[3] = 7;
ptr[4] = 9;
for(int i=0; i<5; i++)
{
printf("%f\n", ptr[i]);
}
return 0;
}
```

output:

```
1.000000
3.000000
5.000000
7.000000
9.000000
```

* calloc() : continuous allocation :

initializes with 0.

ptr = (int*) calloc(5, sizeof(int));

Program:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
float *ptr;
ptr = (float *) calloc(5, sizeof(float));
for (int i=0; i<5; i++)
{
printf("%f\n", ptr[i]);
}
```

Output:

0.000000
0.000000
0.000000
0.000000
0.000000

Practice Qs.64

Q. WAP to allocate memory of size n, where n is entered by the user.

Program:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int *ptr;
int n;
printf("enter n: ");
scanf("%d", &n);
ptr = (int *) calloc(n, sizeof(int));
for(int i=0; i<n; i++)
{
printf("%d\n", ptr[i]);
}
return 0;
}
```

Output:
enter n: 4
0
0
0
0

Practice Qs 68

Q. Allocate memory for 5 numbers. Then dynamically increase it to 8 numbers.

Program:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int *ptr;
ptr = (int *) calloc(5, sizeof(int));
printf("enter numbers(5) : ");
for (int i=0; i<5; i++)
{
scanf("%d", &ptr[i]);
}
ptr = realloc(ptr, 8);
printf("enter numbers(8) : ");
for (int i=0; i<8; i++)
{
scanf("%d\n", &ptr[i]);
}
for (int i=0; i<8; i++)
{
printf("number %d is %d", i, ptr[i]);
}
return 0;
}
```

Output:
enter no(5): 5 4 3 2 1
enter no(8): 8 7 6 5 4 3 2 1
number 0 is 8
" 1 is 7
" 2 is 6
" 3 is 5
" 4 is 4
" 5 is 3
" 6 is 2
" 7 is 1

Practice Qs 61.

Q. Create an array of size 5 (using calloc) & enter its values from the user.

Program:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int *ptr;
ptr = (int *) calloc(5, sizeof(int));
printf("enter numbers(5) : ");
for (int i=0; i<5; i++)
{
scanf("%d", &ptr[i]);
}
for (int i=0; i<5; i++)
{
printf("number %d is %d\n", i, ptr[i]);
}
return 0;
}
```

Output:
enter no(5): 1 3 5 7 9
number 0 is 1
" 1 is 3
" 2 is 5
" 3 is 7
" 4 is 9

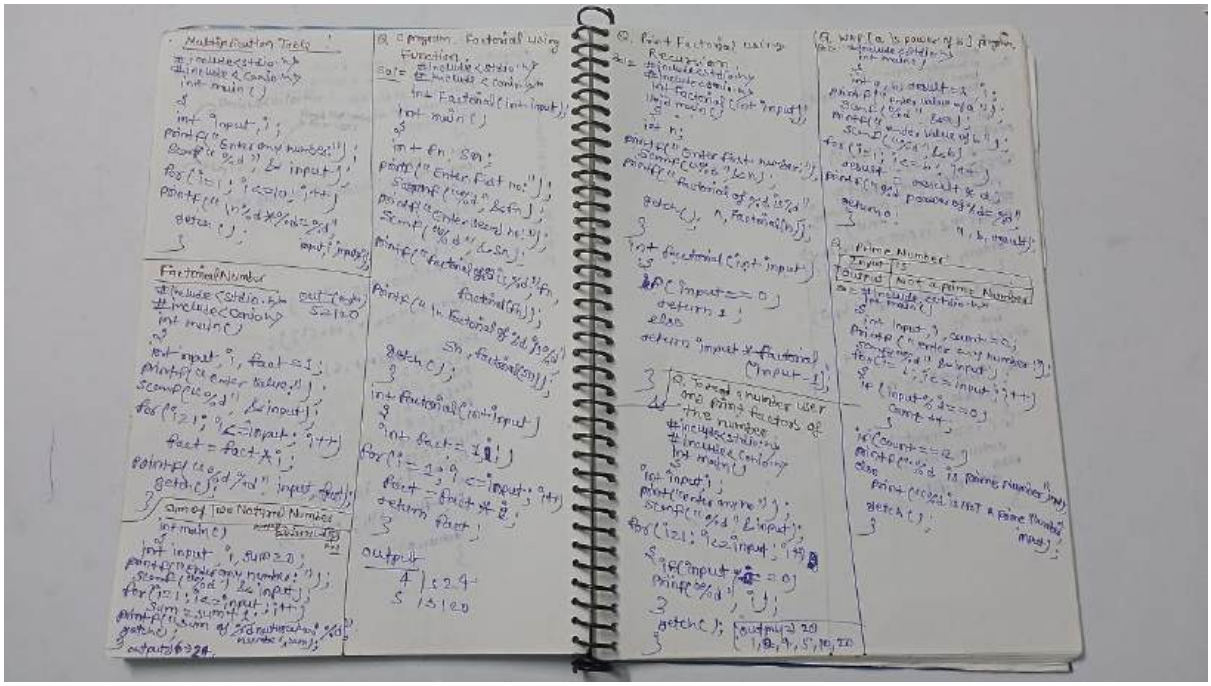
* Practice Qs 70 *

Q. Allocate memory to store first 5 odd numbers then reallocate it to store first 6 even number.

Program:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int *ptr;
ptr = (int *) calloc(5, sizeof(int));
ptr[0] = 1;
ptr[1] = 3;
ptr[2] = 5;
ptr[3] = 7;
ptr[4] = 9;
for(int i=0; i<5; i++)
{
printf("%d\n", ptr[i]);
}
ptr = realloc(ptr, 6);
ptr[0] = 2;
ptr[1] = 4;
ptr[2] = 6;
ptr[3] = 8;
ptr[4] = 10;
ptr[5] = 12;
for(int i=0; i<6; i++)
{
printf("%d\n", ptr[i]);
}
return 0;
}
```

Output:
1
3
5
7
9
2
4
6
8
10
12



Q. Prime Number check to using function.

Sol:-
```
#Include<stdio.h>
bool isPrimeNumber(int input);
void main()
{
int input, i, count=0;
printf("Enter any number:");
scanf("%d", &input);
printf("%d is a Prime Number", input);
printf("%d is Not a Prime Number", input);
return 0;
}
bool isPrimeNumber(int input)
{
int i;
for(i=2; i<input; i++)
{
if(input%i==0)
break;
}
if(i==input)
return true;
else
return false;
}
```