```
name = "Shashank Yadav"
print(name)
```

```
Shashank Yadav
```

```
def gcd(a, b):
    if b==0 :
        return a
    else :
        return gcd(b, a%b)

print(gcd(6, 9))
```

```
3
```

```
def fib(n):
    if n == 0 or n== 1:
        return 1;
    else:
        return fib(n - 1) + fib(n - 2)

print(fib(6))
```

```
13
```

```
def power(a, b) :
    return a**b

print(power(2, 3))
```

```
8
```

```
x = "shashank"

def fun():
    global x
    x = "shashank yadav"

fun()

print(x)
```

```
shashank yadav
```

```
import random
x = int(random.randrange(1, 100))
print(x)
```

```
30
```

```
#multiline strings
A = """Shashank Yadav
```

```
IIT Roorkee
Sophomore @ EPH
From Indore"""
print(A, len(A), sep=" : ")
```

```
Shashank Yadav
IIT Roorkee
Sophomore @ EPH
From Indore : 54
```

```
S = " Hello, World! "
print(S.upper())
print(S.lower())
print(S.strip()) #removes spaces
B = S.split(",") #split but comma and make an array
print(B)
```

```
 HELLO, WORLD!
 hello, world!
Hello, World!
[' Hello', ' World! ']
```

```
S = "I am {} years old, my favourite number is {} "
age, fav = 18, 10
print(S.format(age, fav))
```

```
I am 18 years old, my favourite number is 10
```

```
#lists
names = ["shashank", "mayank", "kartik"]
print(names)
print(names[1:2]) #not include 2
names[0:2] = ["yadav", "dhar"]
print(names)
names.insert(2, "cheetah")
print(names)
names2 = ["rishab", "manav"]
names.extend(names2)
print(names)
names.remove("dhar")
names.reverse()
print(names)

names.clear()
print(names)
```

```
['shashank', 'mayank', 'kartik']
['mayank']
['yadav', 'dhar', 'kartik']
['yadav', 'dhar', 'cheetah', 'kartik']
['yadav', 'dhar', 'cheetah', 'kartik', 'rishab', 'manav']
```

```
['manav', 'rishab', 'kartik', 'cheetah', 'yadav']
[]
```

```
A = [23, 45, 234, 567, 7878]

for i in A:
    print(i)

A.sort(reverse = True)

j = 0
while j < len(A):
    print(A[j])
    j = j + 1

A.sort()

for k in range(len(A)):
    print(A[k])

B = list(A)

print(B)
```

```
23
45
234
567
7878
7878
567
234
45
23
23
45
234
567
7878
[23, 45, 234, 567, 7878]
```

```
# A tuple is a collection which is ordered and unchangeable.
# duplicates are allowed

myTuple = ("banana", "apple", "orange", "watermelon")
print(myTuple)

# adding element to a tuple
myList = list(myTuple)
myList.append("guava")
```

```
myTuple = tuple(myList)

print(myTuple)

('banana', 'apple', 'orange', 'watermelon')
('banana', 'apple', 'orange', 'watermelon', 'guava')

# A set is a collection which is unordered, unchangeable*, and
unindexed.
# Sets are written with curly brackets.

mySet = {"Shashank", "Yadav", "IITR", True, False, 1, 2, 3, 56} # True
and 1 is considered the same value:
print(mySet)

for i in mySet:
    print(i)

mySet.add("Random")

print(mySet)

mySet.pop()
mySet.remove(56)

print(mySet)

mySet.pop()
print(mySet)

{False, True, 2, 3, 'Yadav', 'IITR', 56, 'Shashank'}
False
True
2
3
Yadav
IITR
56
Shashank
{False, True, 2, 3, 'Yadav', 'Random', 'IITR', 56, 'Shashank'}
{True, 2, 3, 'Yadav', 'Random', 'IITR', 'Shashank'}
{2, 3, 'Yadav', 'Random', 'IITR', 'Shashank'}

#Dictionaries are used to store data values in key:value pairs.

dic = {
    "messi" : 10,
    "ronaldo" : 7,
    "shashank" : "abu",
    "kartik" : "lulli",
    "is mayank gay" : True,
```

```
    "names" : ["shashank", "kartik", "manav", "rishab", 1000 ,True]
}

print(dic)

print(dic["shashank"], dic["kartik"], dic["names"], sep=" ")

# all keys ka array
A = dic.keys()
print(A)

B = dic.values()
print(B)

print(dic.items())

if "shashank" in dic :
    print("yes shashank is present")

if "Messi" in dic :
    print("yes Messi is present")
else :
    print("No Messi is not present")

dic.pop("is mayank gay")
print(dic)

for i in dic.values():
    print(i)

for i, j in dic.items():
    print(i, j)
```

```
{'messi': 10, 'ronaldo': 7, 'shashank': 'abu', 'kartik': 'lulli', 'is
mayank gay': True, 'names': ['shashank', 'kartik', 'manav', 'rishab',
1000, True]}
abu lulli ['shashank', 'kartik', 'manav', 'rishab', 1000, True]
dict_keys(['messi', 'ronaldo', 'shashank', 'kartik', 'is mayank gay',
'names'])
dict_values([10, 7, 'abu', 'lulli', True, ['shashank', 'kartik',
'manav', 'rishab', 1000, True]])
dict_items([('messi', 10), ('ronaldo', 7), ('shashank', 'abu'),
('kartik', 'lulli'), ('is mayank gay', True), ('names', ['shashank',
'kartik', 'manav', 'rishab', 1000, True])])
yes shashank is present
No Messi is not present
{'messi': 10, 'ronaldo': 7, 'shashank': 'abu', 'kartik': 'lulli',
'names': ['shashank', 'kartik', 'manav', 'rishab', 1000, True]}
10
7
```

```
abu
lulli
['shashank', 'kartik', 'manav', 'rishab', 1000, True]
messi 10
ronaldo 7
shashank abu
kartik lulli
names ['shashank', 'kartik', 'manav', 'rishab', 1000, True]
```

```
# Nested Dictionary

D = {
    "first" : {
        "first1" : "London",
        "first2" : "Berlin"
    },
    "second" : {
        "second1" : "Tokyo",
        "second2" : "Madrid"
    }
}

print(D)

print(D["first"]["first2"])
print(D["first"]["first1"])
```

```
{'first': {'first1': 'London', 'first2': 'Berlin'}, 'second':
{'second1': 'Tokyo', 'second2': 'Madrid'}}
Berlin
London
```

```
for x in range(2, 7):
    print(x)
```

```
2
3
4
5
6
```

```
for x in range(1, 27, 5):
    print(x)
```

```
1
6
11
16
21
26
```

```python
# lambda function can take any number of arguments but has only one expression
A = lambda x, y, z : x*y + y*z + z*x
print(A(1, 2, 3))
```

```
11
```

```python
# Why Use Lambda Functions? when we have to create a function which multiplies a number with an anoynmous number

def func(x):
    return lambda a : a*x

doubler = func(2)
tripler = func(3)

print(doubler(33))
print(tripler(33))
```

```
66
99
```

```python
# OOPS in Python

class Node():
    val = 5;

head = Node()
print(head.val)
```

```
5
```

```python
class Id:
    def __init__(this, name, enrollment) : # self ko change bhi kr skate hae

        this.name = name
        this.enrollment = enrollment

    def getName(this) :
        print("My name is " + this.name)

    def __str__(this) :
        return f"{this.name}({this.enrollment})"

me = Id("Shashank Yadav", 22123041)

print(me.name, me.enrollment, sep=" : ")
print(me)
me.getName()
```

```
Shashank Yadav : 22123041
Shashank Yadav(22123041)
My name is Shashank Yadav
```

```python
# Python Inheritance

# Inheritance allows us to define a class that inherits all the
methods and properties from another class.
# Parent class is the class being inherited from, also called base
class.
# Child class is the class that inherits from another class, also
called derived class.

# Parent class
class valorant :
    def __init__(self, playerId, rank) :
        self.playerId = playerId
        self.rank = rank

    def getPlayerData(self):
        print(self.playerId, self.rank)

#child class

class freefire(valorant) :
    pass

player1 = freefire("Trivendra", "plastic")

player1.getPlayerData()

# When you add the __init__() function, the child class will no longer
inherit the parent's __init__() function.

class pubg(valorant) :
    def __init__(self, playerId, rank, place):
        super().__init__(playerId, rank)
        self.place = place

    def getPlayerData(self) :
        print(self.playerId, self.rank, self.place)

player2 = pubg("Shashank", "Ascendent", "Indore")
player2.getPlayerData()
```

```
Trivendra plastic
Shashank Ascendent Indore
```

```python
#Class Polymorphism

#Polymorphism is often used in Class methods, where we can have
```

```
multiple classes with the same method name.

class Car:
  def __init__(self, brand, model):
    self.brand = brand
    self.model = model

  def move(self):
    print("Drive!")

class Boat:
  def __init__(self, brand, model):
    self.brand = brand
    self.model = model

  def move(self):
    print("Sail!")

class Plane:
  def __init__(self, brand, model):
    self.brand = brand
    self.model = model

  def move(self):
    print("Fly!")

car1 = Car("Ford", "Mustang")       #Create a Car class
boat1 = Boat("Ibiza", "Touring 20") #Create a Boat class
plane1 = Plane("Boeing", "747")     #Create a Plane class

for x in (car1, boat1, plane1):
  x.move()

Drive!
Sail!
Fly!

# modules

import platform as plt

X = plt.system()
print(X)

print(plt.processor())

print(plt.architecture())

# node information (network)

print(plt.node())
```

```
Windows
Intel64 Family 6 Model 140 Stepping 1, GenuineIntel
('64bit', 'WindowsPE')
Oadduct
```

```
import datetime as dt

print(dt.datetime.now())
```

```
2024-02-06 00:29:44.589663
```

```
import math as mt

print(mt.sqrt(7))
x = mt.pow(2, 3)
print(min(1, x))
```

```
2.6457513110645907
1
```

```
X = int(input())
print(X**2)
```

```
 50
```

```
2500
```

```
def solve():
    N = int(input())
    A = list(map(int, input().split()))
    A.sort()
    print(max(A[0]*A[1], A[-1]*A[-2]))

def main():
    t = int(input())
    for _ in range(t):
        solve()

if __name__ == "__main__":
    main()
```

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

```
12
```

```
print("Hello, World!")
```

```
Hello, World!
```