1. Introduction to Basic Concepts

1.a. Introduction to **LINUX Terminal** and **gcc compiler** by writing and executing addition of two numbers c program.

1.b. Write an algorithm and flow chart to calculate addition of two numbers and execute this program using gcc.

1.c. Introduction to Hackerrank interface by writing a C program to print "Hello World !" .

https://www.hackerrank.com/challenges/30-hello-world

2. Data types and Format Specifiers

2.a. Write a C program that reads a floating point number x and print the value of polynomial $3x^2+5x+9$ and store it in a variable named as y.

2.b. Ramshewar goes to market for buying some fruits and vegetables. He is having a currency of Rs 500 with him for marketing. From a shop he purchases 2.0 kg Apple priced Rs. 50.0 per kg, 1.5 kg Mango priced Rs.35.0 per kg, 2.5 kg Potato priced Rs.10.0 per kg, and 1.0 kg Tomato priced Rs.15 per kg. He gives the currency of Rs. 500 to the shopkeeper. Find out the amount shopkeeper will return to Ramshewar and also tell the total item purchased. Finally Print the Bill in the following format using format specifiers:

S.No.	Name Of the item	Quantity	Price per kg	Total Amount
1	Apple	2.0 kg	Rs. 50.00	Rs. 100.00

2.c. Write a c program to perform sum of two floating point numbers and display the sum in exponent format (Ex: 13.3478+32.1234=0.4572E02)

http://www.geeksforgeeks.org/c/FormatSpecifiers/hf7771.html

3. Operators and Predefined Functions

3.a. Write a C program that takes a floating point number x as input and computes the value of √x using the Babylonian method with the highest accuracy obtainable using the data type double. Do not use any mathematical library function.

The Babylonian method of finding the square-root of a number x works using the following inductive definition. Let y_n be the n^{th} approximation of $\sqrt{x}$, then

$$y_n = \begin{cases} x & \text{if } n = 0, \\ \dfrac{y_{n-1} + \dfrac{x}{y_{n-1}}}{2} & \text{if } n > 0. \end{cases}$$

Hint :

3.b. Write a C Program by taking two sample input values 6 and 3 to perform logical operations on it and output displayed for each operation using Logical AND, OR and NOT Operators.

Logical Operators		
Operator	Description	Example
&&	AND	x=6 y=3 x<10 && y>1 Return True
||	OR	x=6 y=3 x==5 || y==5 Return False
!	NOT	x=6 y=3 !(x==y) Return True

http://studytipsandtricks.blogspot.in/2012/06/write-c-program-by-using-logical-and.html

3.c. Write a c program for the following as two ships are sailing in the sea on the two sides of a lighthouse. The angle of elevation of the top of the light house is observed from the ships are 30^0 and 45^0 respectively. If the lighthouse is 100 m high, print the distance between the two ships.

Hint: Use Predefined Functions in math.h Library

www.careerbless.com/aptitude/qa/height_distance1.php

4. Conditional Statements

4.a. Write a C Program to input Employee Id and basic pay of an employee and then print Employee Id, HRA and Special allowance. Consider HRA is 20% of basic or Rs. 7000/ whichever is less and Special allowance is 10% of basic pay whenever basic exceeds Rs 10000, Otherwise it is 5% of basic pay.

4.b. Develop a program for current billing System based on the given requirements using if else ladder.

Units Consumed	Rate of Charge
0-200	Rs. 0.50 per unit
201- 400	Rs. 0.65 per unit in excess of 200 units
401- 600	Rs. 0.80 per unit in excess of 400 units
601 and above	Rs. 1.00 per unit in excess of 600 units

4.c. The government of India passed a GO regarding tax payment and you have to develop a C program based on some conditions.
If the income is less than 1,50,000 then no tax.
If taxable income is in the range 1,50,001-3,00,000 then charge 10% of tax.
If taxable income is in the range 3,00,001-5,00,000 then charge 20% of tax.
If taxable income is in the range 5,00,001 above then charge 30% of tax.
Calculate the amount of tax he/she has to pay.

5. Loops

5. a. Objective: Stair Case

https://www.hackerrank.com/challenges/staircase

Consider a staircase of size n = 4:

```
   #
  ##
 ###
####
```

Observe that its base and height are both equal to n, and the image is drawn using # symbols and spaces. The last line is not preceded by any spaces.

Write a program that prints a staircase of size n.

Input Format

A single integer, n, denoting the size of the staircase.

Output Format

Print a staircase of size n using # symbols and spaces.

Note: The last line must have 0 spaces in it.

Sample Input

6

Sample Output

```
     #
    ##
   ###
  ####
 #####
######
```

5. b. Objective: Sherlock and Divisors
https://www.hackerrank.com/challenges/sherlock-and-divisors

Watson gives an integer N to Sherlock and asks him: What is the number of divisors of N that are divisible by 2?

Input Format
First line contains T, the number of testcases. This is followed by T lines each containing an integer N.

Output Format
For each testcase, print the required answer in one line.

Constraints
$1 <= T <= 100$
$1 <= N <= 10^9$
Sample Input
2
9
8
Sample Output
0
3
Explanation
9 has three divisors 1, 3 and 9 none of which is divisible by 2.
8 has four divisors 1, 2, 4 and 8, out of which three are divisible by 2.

6. Functions

6.a. Functions are a bunch of statements glued together. A function is provided with zero or more arguments, and it executes the statements on it. Based on the return type, it either returns nothing (void) or something.

https://www.hackerrank.com/challenges/c-tutorial-functions?h_r=internal-search

A sample syntax for a function is

```
return_type function_name(arg_type_1 arg_1, arg_type_2 arg_2, ...) {
    ...
    ...
    ...
    [if return_type is non void]
        return something of type `return_type`;
}
```

For example, a function to read four variables and return the sum of them can be written as

```
int sum_of_four(int a, int b, int c, int d) {
```

```
    int sum = 0;
    sum += a;
    sum += b;
    sum += c;
    sum += d;
    return sum;
}
```

You have to write a function *int max_of_four(int a, int b, int c, int d)* which reads four arguments and returns the greatest of them.

Input Format

Input will contain four integers - a,b,c,d one in each line.

Output Format

Print the greatest of the four integers.
PS: I/O will be automatically handled.

Sample Input

```
3
4
6
5
```

Sample Output

```
6
```

6.b. Objective: Pointers

https://www.hackerrank.com/challenges/c-tutorial-pointer

A pointer in C is a way to share a memory address among different contexts (primarily functions). They are primarily used whenever a function needs to modify the content of a variable, of which it doesn't have ownership.

In order to access the memory address of a variable, val, we need to prepend it with & sign. E.g., &val returns the memory address of val.

This memory address is assigned to a pointer and can be shared among various functions. E.g.int * p = &val will assign the memory address of val to pointer p. To access the content of the memory to which the pointer points, prepend it with a *. For example, *p will return the value reflected by val and any modification to it will be reflected at the source (val).

```
void increment(int *v) {
    (*v)++;
}
```

```
int main() {
    int a;
    scanf("%d", &a);
    increment(&a);
    printf("%d", a);
    return 0;
}
```

You have to complete the function *void update(int *a,int *b)*, which reads two integers as argument, and sets a with the sum of them, and b with the absolute difference of them.

Input Format

Input will contain two integers a, and b, separated by a newline.

Output Format

You have to print the updated value of a and b, on two different lines.

P.S.: Input/ouput will be automatically handled. You only have to complete the *void update(int *a,int *b)* function.

Sample Input

```
4
5
```

Sample Output

```
9
1
```

7. Recursion and Arrays

7.a. Objective: Left Rotation

https://www.hackerrank.com/challenges/array-left-rotation

A *left rotation* operation on an array of size n shifts each of the array's elements 1 unit to the left. For example, if 2 left rotations are performed on array [1,2,3,4,5], then the array would become [3,4,5,1,2].

Given an array of n integers and a number, d, perform d left rotations on the array. Then print the updated array as a single line of space-separated integers.

Input Format

The first line contains two space-separated integers denoting the respective values of n (the number of integers) and d (the number of left rotations you must perform).
The second line contains n space-separated integers describing the respective elements of the array's

initial state.

Constraints

- $1 <= n <= 10^5$
- $1 <= d <= n$
- $1 <= a_i <= 10^6$

Output Format

Print a single line of n space-separated integers denoting the final state of the array after performing d left rotations.

Sample Input

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

Sample Output

```
5 1 2 3 4
```

7. b. Objective: Recursion
https://www.hackerrank.com/challenges/30-recursion

Recursive Method for Calculating Factorial

$$factorial(N) = \begin{cases} 1 & N \leq 1 \\ N \times factorial(N-1) & otherwise \end{cases}$$

Task: Write a *factorial* function that takes a positive integer, N as a parameter and prints the result of N! (N factorial).

Note: If you fail to use recursion or fail to name your recursive function *factorial* or *Factorial*, you will get a score of 0.

Input Format

A single integer, N (the argument to pass to *factorial*).

Constraints

- $2 <= N <= 12$
- Your submission must contain a recursive function named *factorial*.

Output Format

Print a single integer denoting N!.

Sample Input

```
3
```

Sample Output

```
6
```

Explanation

Consider the following steps:

1. factorial(3) = 3 x factorial(2)
2. factorial(2) = 2 x factorial(1)
3. factorial(1) = 1

From steps 2 and 3, we can say factorial (2) = 2 x 1 = 2; then when we apply the value from factorial (2) to step 1, we get factorial (3) = 3 x 2 x 1 = 6. Thus, we print 6 as our answer.

8. SEARCHING & SORTING

8.a. Objective: Ice Cream Parlor

https://www.hackerrank.com/challenges/icecream-parlor

Each time Sunny and Johnny take a trip to the Ice Cream Parlor, they pool together m dollars for ice cream. On any given day, the parlor offers a line of n flavors. Each flavor, i, is numbered sequentially with a unique ID number from 1 to n and has a cost, c_i, associated with it.

Given the value of m and the cost of each flavor t for trips to the Ice Cream Parlor, help Sunny and Johnny choose two flavors such that they spend their entire pool of money (m) during each visit. For each trip to the parlor, print the ID numbers for the two types of ice cream that Sunny and Johnny purchase as two space-separated integers on a new line. You must print the smaller ID first and the larger ID second.

Note: Two ice creams having unique IDs i and j *may* have the same cost (i.e., $c_i == c_j$).

Input Format

The first line contains an integer, t, denoting the number of trips to the ice cream parlor. The 3t subsequent lines describe all of Sunny and Johnny's trips to the parlor; each trip is described as follows:

1. The first line contains m.
2. The second line contains n.
3. The third line contains n space-separated integers denoting the cost of each respective flavor. The i^{th} integer corresponding to the cost, c_i, for the ice cream with ID number i (where 1 <= i <= n).

Constraints

- $1 <= t <= 50$
- $2 <= m <= 10^4$
- $2 <= n <= 10^4$
- $1 <= c_i <= 10^4$
- It is guaranteed that there will always be a unique solution.

Output Format

Print two space-separated integers denoting the respective ID numbers for the flavors they choose to purchase, where the smaller ID is printed first and the larger ID is printed second. Recall that each ice cream flavor has a unique ID number in the inclusive range from 1 to n.

Sample Input

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

Sample Output

```
1 4
1 2
```

Hint: Use Binary Search Concept

8.b. Objective: Big Sorting

https://www.hackerrank.com/challenges/big-sorting?h_r=internal-search

Consider an array of numeric strings, unsorted, where each string is a positive number with anywhere from 1 to 10^6 digits. Sort the array's elements in *non-decreasing* (i.e., ascending) order of their real-world integer values and print each element of the sorted array on a new line.

Input Format

The first line contains an integer, n, denoting the number of strings in unsorted.
Each of the subsequent lines contains a string of integers describing an element of the array.

Constraints

- Each string is guaranteed to represent a positive integer without leading zeros.
- The total number of digits across all strings in is between and (inclusive).

Output Format

Print each element of the sorted array on a new line.

Sample Input 0

```
6
31415926535897932384626433832795
1
3
10
3
5
```

Sample Output 0

```
1
3
3
5
10
31415926535897932384626433832795
```

9. STRINGS - I

9.a. Objective: CamelCase

https://www.hackerrank.com/challenges/camelcase

Alice wrote a sequence of words in CamelCase as a string of letters, s, having the following properties:

- It is a concatenation of one or more *words* consisting of English letters.
- All letters in the first word are *lowercase*.
- For each of the subsequent words, the first letter is *uppercase* and rest of the letters are *lowercase*.

Given s, print the number of words in s on a new line.

Input Format

A single line containing string s.

Constraints

- $1 <= s <= 10^5$

Output Format

Print the number of words in string.

Sample Input

```
saveChangesInTheEditor
```

Sample Output

```
5
```

Explanation

String contains five words:

1. save
2. Changes
3. In
4. The
5. Editor

9.b. Objective: Mars Exploration

https://www.hackerrank.com/challenges/mars-exploration

Sami's spaceship crashed on Mars! She sends n sequential SOS messages to Earth for help.

Letters in some of the `SOS` messages are altered by cosmic radiation during transmission. Given the signal received by Earth as a string, S, determine how many letters of Sami's `SOS` have been changed by radiation.

Input Format

There is one line of input: a single string, S.

Note: As the original message is just `SOS` repeated n times, S's length will be a multiple of 3.

Constraints

- 1 <= S <= 99
- S % 3 = 0
- S will contain only uppercase English letters.

Output Format

Print the number of letters in Sami's message that were altered by cosmic radiation.

Sample Input 0

```
SOSSPSSQSSOR
```

Sample Output 0

```
3
```

Sample Input 1

```
SOSSOT
```

Sample Output 1

```
1
```

10. STRINGS – II

10.a. Objective: Beautiful Binary String

https://www.hackerrank.com/challenges/beautiful-binary-string
Alice has a binary string, B, of length n. She thinks a binary string is beautiful if and only if it doesn't contain the substring “010”.

In one step, Alice can change a 0 to a 1 a (or vice-versa). Count and print the minimum number of steps needed to make Alice see the string as beautiful.

Input Format

The first line contains an integer, n(the length of binary string B).
The second line contains a single binary string, B, of length n.

Constraints

- 1 <= n <= 100
- Each character in B {0,1}.

Output Format

Print the minimum number of steps needed to make the string beautiful.

Sample Input 0

```
7
0101010
```

Sample Output 0

```
2
```

10.b. Objective: Pangrams

https://www.hackerrank.com/challenges/pangrams

Roy wanted to increase his typing speed for programming contests. So, his friend advised him to type the sentence "The quick brown fox jumps over the lazy dog" repeatedly, because it is a pangram. (Pangrams are sentences constructed by using every letter of the alphabet at least once.)

After typing the sentence several times, Roy became bored with it. So he started to look for other pangrams.

Given a sentence s, tell Roy if it is a pangram or not.

Input Format

Input consists of a string s.

Constraints
Length of can be at most 10^3 and it may contain spaces, lower case and upper case letters. Lower-case and upper-case instances of a letter are considered the same.

Output Format

Output a line containing pangram if is a pangram, otherwise output not pangram.

Sample Input

Input #1

We promptly judged antique ivory buckles for the next prize

Input #2

We promptly judged antique ivory buckles for the prize

Sample Output

Output #1

pangram

Output #2

not pangram

Explanation
In the first test case, the answer is pangram because the sentence contains all the letters of the English alphabet.

11. STACKS

11.a. Objective: Maximum Element

https://www.hackerrank.com/challenges/maximum-element?h_r=internal-search

You have an empty sequence, and you will be given N queries. Each query is one of these three types:

```
1 x -Push the element x into the stack.
2   -Delete the element present at the top of the stack.
3   -Print the maximum element in the stack.
```

Input Format

The first line of input contains an integer, N. The next N lines each contain an above mentioned query. *(It is guaranteed that each query is valid.)*

Constraints

Constraints

$1 \leq N \leq 10^5$

$1 \leq x \leq 10^9$

$1 \leq type \leq 3$

Output Format

For each type query, print the maximum element in the stack on a new line.

Sample Input

```
10
1 97
2
1 20
2
1 26
1 20
2
3
1 91
3
```

Sample Output

```
26
91
```

11.b. Objective: Queue using Two Stacks

https://www.hackerrank.com/challenges/queue-using-two-stacks

A queue is an abstract data type that maintains the order in which elements were added to it, allowing the oldest elements to be removed from the front and new elements to be added to the rear. This is called a *First-In-First-Out* (FIFO) data structure because the first element added to the queue (i.e., the one that has been waiting the longest) is always the first one to be removed.

A basic queue has the following operations:

- *Enqueue*: add a new element to the end of the queue.
- *Dequeue*: remove the element from the front of the queue and return it.

In this challenge, you must first implement a queue using *two stacks*. Then process queries, where each query is one of the following types:

1. `1 x`: Enqueue element into the end of the queue.
2. `2`: Dequeue the element at the front of the queue.
3. `3`: Print the element at the front of the queue.

Input Format

The first line contains a single integer, , denoting the number of queries.
Each line of the subsequent lines contains a single query in the form described in the problem statement above. All three queries start with an integer denoting the query , but only query is followed by an additional space-separated value, , denoting the value to be enqueued.

Constraints

- It is guaranteed that a valid answer always exists for each query of type .

Output Format

For each query of type , print the value of the element at the front of the queue on a new line.

Sample Input

```
10
1 42
2
1 14
3
1 28
3
1 60
1 78
2
2
```

Sample Output

```
14
14
```

12. LINKED LIST

12.) a.) Objective: Print the Elements of a Linked List

https://www.hackerrank.com/challenges/print-the-elements-of-a-linked-list

If you're new to *linked lists*, this is a great exercise for learning about them. Given a pointer to the *head* node of a linked list, print its elements in order, one element per line. If the head pointer is null

(indicating the list is empty), don't print anything.

Input Format

The `void Print(Node* head)` method takes the head node of a linked list as a parameter. Each struct *Node* has a *data* field (which stores integer data) and a *next* field (which points to the next element in the list).

Note: Do not read any input from stdin/console. Each test case calls the *Print* method individually and passes it the head of a list.

Output Format

Print the integer data for each element of the linked list to stdout/console (e.g.: using *printf*, *cout*, etc.). There should be one element per line.

Sample Input

This example uses the following two linked lists:

```
NULL
1->2->3->NULL
```

NULL and Node 1 are the two head nodes passed as arguments to `Print(Node* head)`.

Note: In linked list diagrams, `->` describes a pointer to the *next* node in the list.

Sample Output

```
1
2
3
```

Explanation

Test Case 0: *`NULL`*. An empty list is passed to the method, so nothing is printed.
Test Case 1: *`1->2->3->NULL`*. This is a non-empty list so we loop through each element, printing each element's data field on its own line.

12.) b.) Objective: Insert a node at the head of a Linked List

https://www.hackerrank.com/challenges/insert-a-node-at-the-head-of-a-linked-list

You're given the pointer to the head node of a linked list and an integer to add to the list. Create a new node with the given integer, insert this node at the head of the linked list and return the new head node. The head pointer given may be null meaning that the initial list is empty.

Input Format

You have to complete the Node* Insert(Node* head, int data) method which takes two arguments - the head of the linked list and the integer to insert. You should NOT read any input from stdin/console.

Output Format
Insert the new node at the head and return the head of the updated linked list. Do NOT print anything to stdout/console.

Sample Input
NULL , data = 1
1 --> NULL , data = 2

Sample Output
1 --> NULL
2 --> 1 --> NULL

Explanation
1. We have an empty list, on inserting 1, 1 becomes new head.
2. We have a list with 1 as head, on inserting 2, 2 becomes the new head.

Sample Input

1 -> 1 -> 3 -> 3 -> 5 -> 6 -> NULL
NULL

Sample Output

1 -> 3 -> 5 -> 6 -> NULL
NULL

Explanation
1. 1 and 3 are repeated, and are deleted.
2. Empty list remains empty.

12.) c.) Objective: Delete duplicate-value nodes from a sorted linked list

https://www.hackerrank.com/challenges/delete-duplicate-value-nodes-from-a-sorted-linked-list

You're given the pointer to the head node of a sorted linked list, where the data in the nodes is in ascending order. Delete as few nodes as possible so that the list does not contain any value more than once. The given head pointer may be null indicating that the list is empty.

> For now do not be concerned with the memory deallocation. In common abstract data structure scenarios, deleting an element might also require deallocating the memory occupied by it. For an initial intro to the topic of dynamic memory please consult: http://www.cplusplus.com/doc/tutorial/dynamic/

Input Format

You have to complete the `Node* RemoveDuplicates(Node* head)` method which takes one argument - the head of the sorted linked list. You should NOT read any input from stdin/console.

Output Format

Delete as few nodes as possible to ensure that no two nodes have the same data. Adjust the `next` pointers to ensure that the remaining nodes form a single sorted linked list. Then `return` the head of the sorted updated linked list. Do NOT print anything to stdout/console.

Sample Input

```
1 -> 1 -> 3 -> 3 -> 5 -> 6 -> NULL
NULL
```

Sample Output

```
1 -> 3 -> 5 -> 6 -> NULL
NULL
```