Array Elements - Even - Odd

 In your program, you have to declare an array of size 9 and have code to read elements with cin. You need not give any input. As a tester,  I only gave input and expected output for this GDB platform. You just compile and press test button. Your program will take my input and will give output. That oputput will be verified by this GDB.( as I informed GDB the expected output). You will have to get positive pass result.

No need of command line arguments now.

A to Z structure

**A to Z structure:**

Define a structure(s) as show below. Declare a structure pointer variable S. Create all the structures.

**Input** will be given as three numbers which are to be read(cin) into the field variables of **n** , **r, u.**

**Output**should be of five numbers (cout) of the values of field variables of **n , r , u , x , y**

The **logic**of the program is : if the value of  **u** (field variable) is even then **n**(field variable) value should be copied into **x** (field variable)  and **r**(field variable) value should be copied into **y** (field variable) , otherwise,  **n**(field variable) value should be copied into **y** (field variable)  and **r**(field variable) value should be copied into **x** (field variable).

Example1 : input  5 9 6     output : 5 9 6 5 9   (  as **u** (field variable) = 6 , even,   x (field variable)= 5 and **y** (field variable) = 9

Example2 : input 4 8 3   output : 4 8 3 8 4

You need not give any input. As a tester I am giving input and the expected output.

You look at the Structure diagram of your email attachment.

Std Structure

Student structure

Assume the following structure std. fullest

Create 5 such structure elements (array) and fill them with values.

The addr can be cno or hno depending on the value of tag.

Then print all the elements.

Your declaration must be as :  struct std \*S[5];

You look at your email for the diagram of std.

You only have to give input and check .

You have to submit correct code.

Recursive function - single function for min, max, average of an Array

Write detailed instructions of assignment here...

### Recursive Selection sort , Bubble sort

**Special Note : See that there** **should not** be**any for loops in the recursive functions.**

**Don't submit code with for loops in recursive functions.**

**Submit only if you have done both the recursive functions..**

**I am now giving input as 12 numbers. Read first 6 numbers for selection sort and next 6 numbers for bubble sort and sort and output.**

Example Input : 6 9 1 2 8 45        27 5 3 7 4 63

Expected output : 1 2 6 8 9  45     3 4 5  7  27 63

You  read 12 numbers. You need not give input. I only give input and check the output.

### Recursive - Sorts - Selection sort , Bubble sort

**Special Note : See that there** **should not** be**any for loops in the recursive functions.**

**Don't submit code with for loops in recursive functions.**

**Submit only if you have done both the recursive functions..**

**I am now giving input as 12 numbers. Read first 6 numbers for selection sort and next 6 numbers for bubble sort and sort and output.**

Example Input : 6 9 1 2 8 45        27 5 3 7 4 63

Expected output : 1 2 6 8 9  45     3 4 5  7  27 63

You  read 12 numbers. You need not give input. I only give input and check the output.

### Blessed - Recursion

1. Given a number n, generate all distinct ways to write n as the sum of positive integers. For example, with n = 4, the options are 4, 3 + 1, 2 + 2, 2 + 1 + 1, and 1 + 1 + 1 + 1.

2. Write a recursive function that, given two strings, returns whether the first string is a

subsequence of the second. For example, given *hac* and *cathartic*, you should return true,

but given *bat* and *table*, you should return false.

3. Given a nonnegative integer n, write a function that lists all strings formed from exactly n pairs of balanced parentheses. For example, given n = 3, you'd list these five strings:

              ((( )))                 (( )( ))             (( ))( )          ( )(( ))         ( )( )( )

As a hint, given any string of n≥ 1 balanced parenthesis, focus on the first open parentheses and the close parentheses it matches.

4. The Fibonacci strings are a series of recursively­defined strings. F₀ is the string **a**, F₁ is the string **bc**, and Fₙ₊₂ is the concatenation of Fₙ and Fₙ₊₁. For example, F₂ is **abc**, F₃ is **bcabc**, F₄ is **abcbcabc**, etc. Given a number n and an index k, return the kth character of the string Fₙ.

### Sorts - I - 5Qs

Submit the Assignments is a single program with different functions. Minimum three Answers submissions in a single program is expected.

Looks at the QAs-Sorts-I-8-9-2020.pdf for the questions.

### Sorts - III

Submit all either as  functions or main programs.

### Stacks - I - C

Submit all the five as functions or main() programs.

### Stacks - I - S

Submit all the five as functions or main() programs.

### Stacks - \* - Maze

Create the matrix ( 8\*8)  of 0's and 1's as given in the slides. The input 1 1 6 6 will be given by me as test case 1.  The output should be 1 if there is a path or 0 if there is no path.

### T1 - 11- 9 -2020

**Find all elements in an array that are greater than all elements present to their**

**right**

Given an unsorted array of integers, print all elements which are greater than all elements

present to its right.

For example,

**Input:**{ 10, 4, 6, 3, 5 }

**Output:**The elements greater than all elements to its right are 10, 6, 5

### T2 - 11 - 9 -2020

Infix evaluation with two stacks and in a single pass ( that means you have to read the input string only once)

### T3 - 11 - 9 - 2020

Given two integer sequences, one of which is the push sequence of a stack, check whether the other sequence is a corresponding pop sequence or not.

For example, if 1, 2, 3, 4, 5 is a push sequence( assume that integers from 1 to 5 can be  pushed on to stack in sequence not all at a time.) , 4, 5, 3, 2, 1 is a corresponding pop sequence, but the sequence 4, 3, 5, 1, 2 is not.

For case 1 : automatic check.. I give input the pop sequence as 4 5 3 2 1  (you have to read this)

And the output should be 1     ( means Yes)

And for case2 : I give input as 4 3 5 1 2

And the output should be 0    (means No)

### SSS - 12 - 09 - 2020

**SSS**- **Saturday Sunday Specials**

1. Infix expression evaluation using a single stack﻿

2. Iterative Quick sort

3. Minimum number of bracket(symbol) reversals needed to make an expression balanced, if the given input expression is not balanced.

4. Recursive postfix evaluation ( without a stack)

5. Recursive function to convert prefix expression to postfix expression

      Prefix : \*\*+AB-+CDEF        postfix : AB+CD+E-\*F\*

Fixes : affixes your success

**Fixes** : **affixes your success** – one of these (fixes) ,  in  many of them ( OTs and Interviews)

| **Infix Expression** | **Prefix Expression** | **Postfix Expression** |
| --- | --- | --- |
| A + B \* C + D | + + A \* B C D | A B C \* + D + |
| (A + B) \* (C + D) | \* + A B + C D | A B + C D + \* |
| A \* B + C \* D | + \* A B \* C D | A B \* C D \* + |
| A + B + C + D | + + + A B C D | A B + C + D + |

Write functions to convert: ( **C** – any 3,    **S**– any 5 ,    **E** – all )

1. Infix to prefix

2. Infix to postfix

3.  Prefix to infix

4. Prefix to postfix

5. Postfix to infix

6.  Post fix to prefix

All the above are easy only. One way or the other, they connect to the code of infix to postfix conversion and so on.

You think on your own and try to get your own method , **E = MC2**

Sample Example algorithm:  Infix to prefix

* Reverse the given infix expression. (Note: do another reversal only for brackets).

(A + B) \* (C + D)    reverse it to       ) D + C ( \* ) B + A (

again reverse only brackets  ( D + C ) \* ( B + A )

* Do Infix to postfix expression and get the result.

         DC + BA + \*

* Reverse the result to get the final expression. (prefix expression).

             \*+ A B + C D

All other conversions do on your OWN to be WON

### SSS - 13 - 09 - 2020

**SSS**- **Saturday Sunday Specials**

Saturday **Sunday**Specials of 13 - 9 - 2020.

( **C** – any 3,    **S**– any 4 ,    **E** – all 5 )

Look at SSS-13-9-2020.pdf  file for the Questions

### RCC1 - 14- 9 - 2020

Write two functions and call then in main()

read input as specified here...

1. Revererse  Q.

Example input :   A N J P K L M #

output :  M L K P J N A

 2. Check for Q a Palindrome

Example input :  a b c d d c b a #

output : 1

input : a b c d l j b a #

output : 0

### Q1-1-Islands

**Count the Number of Islands**

o for water

1 for land

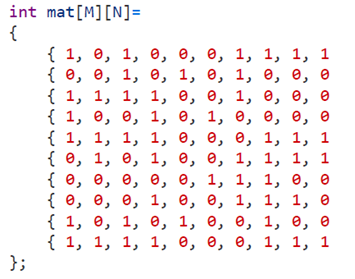
1's in all eight directions put together constitutes an island. Look at PPT for the island matrix to get idea.

you have to get answer 5:

No input is given in the test case, as you have to initialize the matrix in your program as shown below.

After executing your program , only one cout<< that should match with answer 5.

Your program should contain no cin >> statements, and only one cout << statement.



### Q1-2- Prefix Evaluation using Q

**Evaluate prefix expression using a Q.**

No input is given in the test case, as you have to initialize a character array in your program as shown below.

P[]  =  "-+\*9+28\*+4863"  ;

You can use P[] as your input.

After executing your program , only one cout<< that should match with answer.

Your program should contain no cin >> statements, and only one cout << statement.

### Q1-Cs

**1. Interleave the first half of the queue with second half**

 Given a queue of integers of even length, rearrange the elements by interleaving the first half of the queue with the second half of the queue. You can use only one stack can be used as an additional space.

Examples:     Input :  1 2 3 4

                    Output : 1 3 2 4

Input :   11  12  13  14  15  16  17  18  19  20

Output : 11  16  12  17  13  18  14  19  15  20

**2. Permutations of string of length n**

**3. Generate Binary numbers from 1 to n using a Queue**

**4. Sort a Queue ( Either recursion or iteration or Both)**

### Radix-sort with Count sort

Radix Sort :     with Count Sort : to sort the digits of 1's place , 10's place 100's place, you must have to use count sort.

Input :  n = 10

A[] :   21   10  17  34  44  11  654  123  957  89

Output : 10  11  17  21   34  44  89 123  654 957

first you have to cin >> n

Then read 10 numbers.

Next you cout<< the 10 sorted numbers which should match the above.

Positives convert negatives

**Positive converts negative to be positive:**

Given a M x N matrix of integers whose each cell can contain a negative, zero or positive value, code to find the minimum number of passes required to convert all negative values in the matrix to positive.

Only a non-zero positive value at cell (i, j) can convert negative values present at its adjacent cells (i-1, j), (i+1, j), (i, j-1), and (i, j+1) i.e. up, down, left and right.

For example, matrix that is given in the file "Queues-Qs -Positives convert.pdf" needs 3 passes as demonstrated in that figure.

Input : You initialize the matrix in the program itself.

Output :  3

You should have only one cout<<  whose result could be  3 to get matched/passed.

### Queues - Reviews

1.**Implement k Queues in a single array.**

void enqueue ( struct queue &Q , int i, int x )  // enters element x into queue i

int dequeue ( struct queue &Q, int i ) // deletes an element from queue i

Note that every Queue is a circular queue.

2. **Bubble sort using one Queue and 5 variables.**

You are given an array of integers. You have to use only one Queue and any five variables only( like  i , j , k , l , m )

You are not supposed to use any other structures, arrays and variables.

3.  **Generic Queue**

A Generic Queue contains elements of type int, char, and float. Implement enqueue , and dequeue operations.

### T1 - 17 -9 -2020 Stack - Queue - thro Dequeue

**Implement Stack and queue using a Dequeue**

### T2 - 17 - 9 - 2020 - Dequeue of int, char

**Dequeue of int, char**

input : example  N, I , 17 , T , 9 , 20 , C , S , 21 , E

The dequeue DQ should look like now N I T C S E 17 9 20 21

Use a Queue to arrange the elements of DQ as :

output : 17 9 20 21 N I T C S E

you have to type the input on your own.

Out put will be checked with test case : 17 9 20 21 N I T C S E

### T3 - 17 - 9 - 2020 - Knight Reach

**Knight Reach**

Find the minimum number of steps for a Knight in a chess board to reach from a source location to destination location.

You have to read the input four numbers :  (sx , sy ) and (dx, dy )

you need not type the numbers, they will be given as input in the test case. you just use cin>>

Input : 7 0 07  ( this will be given as input in the test case)

output : one cout<<  , your output should match the test case output

### Linked Lists - 1 - lptr opeartions

**Linked List Operations :**

Create a Linked list with given numbers termination of input is -1   ( you have to read with cin>>  )

Print all data values  (you have to use cout<<  )

Print all data values in reverse order  ( you have to use cout << )

Print the count of number of nodes ( how many nodes ) ( one cout<< )

Print minimum of all data values ( one cout <<)

Print maximum of all data values ( one cout <<)

Print middle node data value ( one cout <<  , the input , that I give as test case, will have odd number count, so that you can comfortably find the middle node)

Print number of odd data values  ( one cout << )

Print number of even data values ( one cout << )

Find k, ( one cin>>k , and if found you have to cout <<     1  , not found cout <<    0

### Linked - Lists - 2 - Alias of Genius

**Linked List Operations :**

**First Implement the following functions :**

**Create Linked List**

**Printing Liked List**

**Add-front   k ( one cin>> )**

**Add-before  x  , y ( two cin>>  add x before y )    y could be data value of first node**

**Add-after     x, y  ( two cin >> add x after y)**

**Delete k     ( one cin >>  k  )  k could be first node data value**

**Delete min   ( minimum data value node is to be deleted )**

**Delete max ( maximum data value node is to be deleted )**

**The following sequence ( cin>>   , cout << )  is to be followed to pass test case.**

Create a Linked list with given numbers termination of input is -1   ( you have to read with cin>>  )

Print all data values  (you have to use cout<<  )

Add-front   k ( one cin>> )

Print all data values  (you have to use cout<<  )

Add-before  x  , y ( two cin>>  add x before y )

Print all data values  (you have to use cout<<  )

Add-after     x, y  ( two cin >> add x after y)

Print all data values  (you have to use cout<<  )

Delete k     ( one cin >>  k  )  k could be first node data value

Print all data values  (you have to use cout<<  )

Delete min   ( minimum data value node is to be deleted )

Print all data values  (you have to use cout<<  )

Delete max ( maximum data value node is to be deleted )

Print all data values  (you have to use cout<<  )

SSS -19-9-2020 - Fit - LL - Set

**SSS**- **Saturday Sunday Specials**

**Saturday**SundaySpecials of **19 - 9 - 2020**.

Write functions for the following :

L1 , L2 are pointers to linked list of  sorted (ascending order)  unique numbers.

L3 is a pointer to linked list of numbers.

**1.    union**function for   L3  =  union of L1 , L2  ( L3 will also be a sorted list )

**2.    intersection** function for  L3   = intersection of L1 , L2  ( L3 will also be a sorted list )

**3.    diff** function for  L3    = difference of L1 , L2   ( L3 will also be a sorted list

L1 , L2 are pointers to linked list of  Unsorted      unique numbers.

L3 is a pointer to linked list of numbers.

**4.**  **u-union** function for   L3  =  union of L1 , L2      (  data values of L1 should appear first)

**5.**    **u-intersection** unction for  L3    = intersection of L1 , L2   (  data values of L1 should appear first)

**6,**     **u-diff** function for  L3    = difference of L1 , L2    : L1 - L2

**The following sequence ( cin>>   , cout << )  is to be followed to pass test case.**

First use the **create/insert**function to create linked lists  L1 , L2  with given numbers termination of input is -1

( you have to read with cin>>  till  -1 for L1, and  again you have to read with cin>>  till  -1 for L2 )

**L3 =** union( L1 , L2)     ( union of L1 , L2 has to return L3 )

Print all data values of L3  (you have to use cout<<  )

**L3 =** intersection( L1 , L2)     ( intersection of L1 , L2 has to return L3 )

 Print all data values of L3  (you have to use cout<<  )

**L3 =**  diff( L1 , L2)     ( difference of L1 , L2  : L1 - L2 has to return L3 )

Print all data values of L3  (you have to use cout<<  )

Again use the **create/insert**function to create linked lists  L1 , L2  with given numbers termination of input is -1

( you have to read with cin>>  till  -1 for L1, and  you have to read with cin>>  till  -1 for L2 )

**L3 =**  u-union( L1 , L2)     ( union of L1 , L2 has to return L3   ,   data values of L1 should appear first)

Print all data values of L3  (you have to use cout<<  )

**L3 =**   u-intersection( L1 , L2)     ( intersection of L1 , L2 has to return L3 )

 Print all data values of L3  (you have to use cout<<  )

**L3 =**   u -diff( L1 , L2)     ( difference of L1 , L2  :  L1 - L2  has to return L3 )

Print all data values of L3  (you have to use cout<<  )

* [**Draft version**](https://www.onlinegdb.com/s/as/12199)

SSS -20-9-2020 - Blinks of Links

**SSS**- **Saturday Sunday Specials**

**Saturday Sunday Specials** of **20 - 9 - 2020.**

**Write Functions ( recursive / non-reccursive ) for the following:**

**Assume all LLs are having data values as integers.**

**1.**Check if a given LL is a palindrome.

.**2.** Remove duplicates from a given sorted LL (recursive)

**3.**Remove duplicates from a given unsorted LL.

**4.** Delete last occurrence of a duplicate data value node from a given LL ( duplicate data values can be there)

Example : input : L = { 1 , 3, 2, 4, 5, 7, 8, 3, 4 , 2, 7, 4, 2 }

                 output :  L = { 1 , 3, 2, 4, 5, 7, 8 , 4 , 2 }

**5.**Segregate even data values to left and odd data values right of a given LL

input : L = {  4 , 7 , 8,  2, 5, 4, 9, 1, 6 }

output : L = { 4,  8,  2, 4, 6, 7  5 9 1  }

**6.** insertion sort on a give LL with unique data values

**7.** reverse a given LL ( try for recursion )

**8.** Swap Kth node from the beginning with Kth node from end in a given LL

input :  L = { 4, 9 , 8, 7 , 2 , 6, 1, 5 , 3 }

k = 3 .

output : L = {  4, 9, 1, 7, 2, 6, 8, 5, 3 }

**9.** Check whether L2 is present in L1.

Example : L1 = { 5, 8, 2, 9, 5, 8, 2, 3, 7, 1 , 6}

L2 = { 8, 2, 3 }  output : 1     if L2 = { 8, 2, 7 }   output : 0   if L2 = { 7 , 1}  output :

**10.**  Given two linked lists, merge their nodes together to make one list, taking nodes alternatively between the two lists. If either list runs out, all the nodes should be taken from the other list.

Example :

Input : L1 = { 1 ,  3, 5 }  L2 = { 2, 4, 6, 8, 9}

Output : L3 = { 1, 2, 3, 4, 5, 6, 8, 9 }

**Sequence ( cin>>   , cout << )  to be followed to pass test case:**

Create a Linked list with given numbers termination of input is -1   ( you have to read with cin>>  )

**1.**Check if the LL  is a palindrome or not . for 'Yes' cout <<    1  for  'No'  cout<<   0

Create a Linked list with given numbers termination of input is -1   ( you have to read with cin>>  )

.**2.** Remove duplicates from the sorted LL (recursive)

Print all data values  (you have to use cout<<  )

Create a Linked list with given numbers termination of input is -1   ( you have to read with cin>>  )

**3.**Remove duplicates from a given unsorted LL.

Print all data values  (you have to use cout<<  )

Create a Linked list with given numbers termination of input is -1   ( you have to read with cin>>  )

**4.** Delete last occurrence of a duplicate data value node from a given LL ( duplicate data values can be there)

Print all data values  (you have to use cout<<  )

 Create a Linked list with given numbers termination of input is -1   ( you have to read with cin>>  )

**5.**Segregate even data values to left and odd data values right of a given LL

Print all data values  (you have to use cout<<  )

Create a Linked list with given numbers termination of input is -1   ( you have to read with cin>>  )

**6.** insertion sort on a give LL with unique data values

Print all data values  (you have to use cout<<  )

Create a Linked list with given numbers termination of input is -1   ( you have to read with cin>>  )

**7.** reverse a given LL ( try for recursion )

Print all data values  (you have to use cout<<  )

Create a Linked list with given numbers termination of input is -1   ( you have to read with cin>>  )

Cin >> k

**8.** Swap Kth node from the beginning with Kth node from end in a given LL

 Print all data values  (you have to use cout<<  )

Create two Linked lists with given numbers termination of input is -1  ( you have to read with cin>>  till  -1 for L1, and  you have to read with cin>>  till  -1 for L2 )

**9.** Check whether L2 is present in L1.

 if L2 is present in L1  cout<<  1   if   L2  is not present cout<< 0

Create two Linked lists with given numbers termination of input is -1  ( you have to read with cin>>  till  -1 for L1, and  you have to read with cin>>  till  -1 for L2 )

**10.**  Given two linked lists, merge their nodes together to make one list, taking nodes alternatively between the two lists. If either list runs out, all the nodes should be taken from the other list.

 Print all data values  of L3 (you have to use cout<<  )

* [**Draft version**](https://www.onlinegdb.com/s/as/12252)

### RCC - T1 - 21-9-2020

**Polynomial Addition**

Let polynomial**:  P1 = 12 x7+ 10 x4 + 18 x2**

Let polynomial**: P2 =  8 x6+ 9 x4 + 27 x + 45**

**Write code for**:  **P3 = P1 + P2  = 12 x7+ 8 x6+ 19 x4 + 18 x2 + 27 x + 45**

Test case Input : ( cin >> ) :    12 7 10 4 18 2 -1

                            (cin >>  ) :    8 6 9 4 27 1 45 0 -1

Output: ( cout <<  ) :               12  7   8  6  19  4  18  2  27 1  45 0

### RCC -T2- 21-9-2020

**Create a Generic Linked list**

whose node value can be either an integer or character.

Read the input as given in the sequence.

If the input is a character, add to the beginning of the list and if it is an integer add at the end.

Input sequence would be :  N  I  21  T  9  20  C   S   45  E

test case input is : 0 N 0 I 1 21 0 T 1 9 1 20 0 C 0 S 1 45 0 E 0 #

(you read using one  cin>>  for tag , one  cin >> for value,  till termination character ‘#’ )

The list should be formed with node contents in sequence as:

{ N  I  T C S E  21  9 20 45 }

Print ( cout <<  ) the contents of the list as output

(test case output):  N I T C S E 21 9 20 45

RCC - T3 - 21-9-2020

**Reverse specified portion of a Linked List**

Example( test case) :   ( cin >> )    L   =   1 2 3 4 5 6 7 8  - 1

Start position       ( cin >> )   i = 2

End position         ( cin >> )   j = 5

Output ( test case) :  ( cout  << )       1 5 4 3 2 6 7 8

* [**Draft version**](https://www.onlinegdb.com/s/as/12315)

### CCC - T1 - 22-9-2020

Create a Linked list by making the last node's next pointer points to the fifth node on the list.

Now write a function to find, whether there is a loop in the LL and if it there what is the node value of the start of the loop node(5th node). ( Hint: use slow ptr, fast ptr )

Examplle :   L =  {  4  7  8  6  2  1  9  5 3  } , here the last node's ( data 3 ) next ptr should be linked to 5th node ( data value 2)

Input :  4  7  8  6  2  1  9  5 3 -1   ( read with cin >> till -1 )

Output : 1   ( as there is loop ) , in no loop 0 : but here as the LL is constructed with loop you will get output 1.

           :  2   (  at the node ( data value 2 ) the loop starts, i,e 5th node )

* [**Draft version**](https://www.onlinegdb.com/s/as/12584)

CCC - T2 - 22 - 9 - 2020

Write a function for deleting from list LL, nodes occupying positions **indicated in list LL itself.**

For instance, if L= ( 1 3 5 7 8) , then after deletion, L= (3 7).

Explanation:  The positions of nodes at 1 , 3, 5 are nodes of  1 , 5, 8 of  given original LL are deleted. There is no node at positions 7 and 8 in the original LL.  so The original LL will now become as L = (3  7 )

You should not use another linked lists or arrays, but you have to re-adjust the existing list nodes by using few variables.

Input  : 1 3 5 7 8 -1  ( cin >> till -1 )

output : 3 7               (  print the LL )

* [**Draft version**](https://www.onlinegdb.com/s/as/12589)

### CCC - T3 - 22-9-2020

**You should not use Pointers.**

Create a Generic Queue where each element is having varying number of items.

Example Q =  {    ( 3 , 'N' , 9.8 , 6 ) ,  ( 'N' , 8, 'C' , 5, 'L', 8.1 ) , ( 9.7 , 5.4, 'B' ) , (  7, 2, 45, 4.5, 9.3,72, 81, 36 )  , ( 'A' ,6, 'C', 7.5,'D' )  }

each of                                             are elements of queue.

Implement enqueue() , Dequeue() operations on the queue.

Print the contents of the queue after creation and a dequeue operation.

Note :  **You should not use Pointers.**

### Lab-Assignment-24-9-2020

Implement Stack Operations as a Linked List  :

Write main() function to add few elements and delete few .

Implement Queue Operations as a Linked List  :

Write main() function to add few elements and delete few .

There is no test case for this. Submit the code.

### CCC - T1 - 24 -2-2020

Given a singly linked list

    L:   L0 → L1 → … → Ln-1 → Ln,

reorder it to:

    L0 → Ln → L1 → Ln-1 → L2 → Ln-2 → …

You must do this in-place without altering the nodes’ values.

For example,

 Given {1,2,3,4}, reorder it to {1,4,2,3}.

create L with cin >> ... terminated by -1

print cout <<     node values of L

### CCC-T2- 24-9-2020

Write recursive function for:

Addition of 2 numbers (  5729 + 8522 = 14251 ) whose digits are nodes of 2 LLs.

 Ex. 5à7à2à9,

8à5à2à2   output: 1à 4 à2à5à1

input : 5729       ( two cin>> )

          8522

ouput : 14251   ( one cout<< )

### CCC-T3-24-9-2020

Create two Linked Lists , L1, L2 .

Let  L1 :  5 à 3 à7  à 6 à 9  à1 à 5

       L2 : 2 à 4 à 8

And connect the last node of L2 to a node of  L1 whose data value is 6.

Write code to find the node at which the intersection of two singly linked lists begins. (i.e the meeting point)

 L1 :   5 à 3 à7  à 6 à 9  à1 à 5

                               ä

L2 :     2 à 4 à 8

input :    (cins >> )  :  5 3 7 6 9  1 5 -1

                                   2 4 8 -1

output : ( cout ) :  6

Note : ﻿﻿It is to be noted that your check function doesn't know the intersection point's (node's) data value or ptr address

﻿

CCC-T4-24-9-2020

Given a linked list L and a value x,

write a function to partition  L such that all nodes less than x come before nodes greater than or equal to x.

Now use this function to sort L in ascending order.

Example :  L =  {  7 5 4 9 6 3 2 1 8 }    . you can choose x as the first node or any other node.

output L = { 1 2 3 4 5 6 7 8 9 }

input     ( cins >> )  :        7 5 4 9 6 3 2 1 8 -1

output  ( couts <<) :         1 2 3 4 5 6 7 8 9

Note : Must have to implement the 'logic' in the question with recursive calls to partition() to get sorted,

but not not a sorting method.

* [**Draft version**](https://www.onlinegdb.com/s/as/12991)

MLL-Flatten-Q2

**Flatten the Multi level lis**t to a Singly Linked List:

Example :

Input : ML:       3 – 4 – 7 – 18 – 45 – 36

                      |     |                    |

                      8    5                 6

                      |                         |

                      9                       2

Output: LL :    3 – 8 – 9 – 4 – 5 – 7 – 18 – 45 – 6 – 2 – 36

Input ( cin >>) : 3 1 8 9 -1 4 1 5 -1 7 0 18 0 45 1 6 2 -1 36 0 -2

( 1 – indication of down link,  0 – no down link

  -1 – termination of down LL ,  -2 termination of  ML

Output (cout <<) :  3 8 9 4 5 7 18 45 6 2 36

* [**Submitted versi**](https://www.onlinegdb.com/s/as/13207)

MLL - Level - Depth - Q3

**Print Level - Depth wise of a MLL.**

Example MLL:

               10 – 5 – 12  –  7 – 18

                |                        |

               4 – 20 – 9         25 – 6

                      |      |            |

                     3     63         2 – 8 – 54

                             |            |

                           72          47 – 86

                                           |

                                          36

**Input**: 10 1 4 0 20 1 3 0 -1 9 1 63 1 72 0 -1 -1 -1 5 0 12 0 7 1 25 1 2 1 47 1 36 0 -1 86 0 -1 8 0 54 0 -1 6 0 -1 18 0 -1

( 1 for down link, 0 for no down link , -1 for end of next link )

Output1 : Level–wise : 10 5 12 7 18 4 20 9 25 6 3 63 2 8 54 72 47 86 36

Output 2 : Depth–wise: 10 4 20 3 9 63 72 5 12 7 25 2 47 36 86 8 54 6 18

 First cout << Level wise  ,               next  cour << Depth wise

(Input : 10 1 4 0 20 1 3 0 -1 9 1 63 1 72 0 -1 -1 -1 5 0 12 0 7 1 25 1 2 1 47 1 36 0 -1 86 0 -1 8 0 54 0 -1 6 0 -1 18 0 -1 )

* [**Submitte**](https://www.onlinegdb.com/s/as/13214)

SSS -26-9-2020 -MLL-Sort

**SSS**- **Saturday Sunday Specials**

**Saturday**SundaySpecials of **26 - 9 - 2020.**

**Sort the Multi level lis**t

**You should not use Linked list or arrays and no sorting method.**

**You must have to go through the MLL  ONLY ONCE**

**There should not be a Linked List code , or construction of Linked List in your code.**

**Don’t submit codes with Linked List sort or arrays.**

**You have to go through the constructed MLL ONLY ONCE and print the sorted order.**

( You should not construct a Linked List from MLLand sort it. You have to sort MLL only)

Example :

Input :MLL:       3   –    7 –       10   –  36

                      |          |             |

                      8        12         18

                      |           |            |

                      15       45       72

Output: LL :    3 –  7– 8 – 10 – 12 – 15 – 18 – 36 – 45– 72

Input ( cin >>) : 3 1 8 15 -1 7 1 12 45 -1 10 1 18 72 -1 36 0 -2

( 1 – indication of down link,  0 – no down link

  -1 – termination of down LL ,  -2 termination of  ML

Output (cout <<) :   3  7 8  10  12  15 18  36  45 72

* [**Submitted version (1 p**](https://www.onlinegdb.com/s/as/13229)

### SSS - 26-9-2020 - Target

**SSS**- **Saturday Sunday Specials**

**Saturday**SundaySpecials of **26 - 9 - 2020.**

Targets - Print all targets ( paths ) from start node 1.﻿

The nodes are numbered (with data values) from 1 to 18.

The output is needed to print the path from 1 to 16  and from 3 to 12

The test case input sequence is  node's data value number of links it has

test case input ( cin >> )  : 1 1 2 1 3 2 4 1 5 2 6 1 7 0 8 1 9 0 10 3 11 1 12 0 14 1 15 1 16 0 17 1 18 0

input ( source point s, target point t )  ( cin >> s , cin >> t )  :  1 16

print path ( cout << )

input ( source point s, target point t )  ( cin >> s, cin >> t )  :  3 12

print path ( cout << )

test case output :  1 2 3 10 14 15 16

test case output : 3 10 11 12

SSS - RPL - 27-9-2020

**Saturday Sunday**Specials of **27 - 9 - 2020**.

**Random Pointer Linked List**

Create a linked list as below:

5 à 2 à  1 à 8  à 6  à 3 à 9 à4 à7

         |\_\_\_\_|\_\_\_á     á      |\_\_\_\_\_\_\_á

                  |\_\_\_\_\_\_\_\_|

Print the RPL as below: ( You should not modify the RPL during printing)

5 2 1 8 6 3 9 4 7

5 2 8 6 3 9 4 7

5 2 8 6 3 4 7

5 2 1 6 3 9 4 7

5 2 1 6 3 4 7

5 2 1 8 6 3 4 7

Input : test case   ( Node's data value , next ptr node, random ptr node  ) three cins till 7 0 0

             5 2 0

             2 1 8

             1 8 6

             8 6 0

             6 3 0

             3 9 4

             9 4 0

             4 7 0

             7 0 0

output : test case : Print the RPL as below: cout <<  ( You should not modify the RPL during printing)

5 2 1 8 6 3 9 4 7

5 2 8 6 3 9 4 7

5 2 8 6 3 4 7

5 2 1 6 3 9 4 7

5 2 1 6 3 4 7

5 2 1 8 6 3 4 7

* [**Submitted version (0 passed of 1)**](https://www.onlinegdb.com/s/as/13320)

SSS- LL Game - 27-9-2020

**SSS**- **Saturday Sunday Specials**

**Saturday Sunday**Specials of **27 - 9 - 2020**

Consider a game played between two players A and B. The game is defined by two lists of integers L\_1 and L\_2. The game ends when both lists are empty. The two players play alternately, starting with player A, and at each move the player making the move selects either the head of L\_1 or head of L\_2    and removes it from the corresponding list. (If one of the lists is empty, the player has to select the head of the other non-empty list.)

When the game ends, the winner is the player whose sum of selected integers is greater. The game is a draw if the sums are equal.

Implement the game as a function that takes two lists as parameters, and determines whether for the game defined by these two lists, any of the players has a winning strategy. A winning strategy is a way of ensuring that no matter how the other player plays, it is possible to play in a way to ensure a win.

 For example, if the lists are  (2,1) (3,5) player B has a winning strategy.

Whatever move is made by player A first, player B selects from the same list.

The possible selections are A: 2,3  B: 1,5 or A: 3,2  B: 5,1, so B wins in all cases.

Input : L1 :  2 1 -1         ( cin >>  termination -1 )

            L2 : 3 5 -1          ( cin >> termination  -1)

output : B                       (cout <<      because B wins in all cases )

* [**Submitted ver**](https://www.onlinegdb.com/s/as/13379)

### MLL-Create-Q1

#### **Create a Multi-Level List :**

#### **{ A B { C D E } F { G { H I J } K } L M }**

#### **input :     {AB{CDE}F{G{HIJ}K}LM} #           ( cin >>   a char  till '#' )**

#### **output : { A B { C D E } F { G { H I J } K } L M }**

#### 

#### **( cout <<  observe one ' ' after each character in the output)**

### MLL-Radix-Sort-Array-Ptrs-Q4

**Radix Sort :**

Implement radix sort by taking Buckets as an Array[10] pointers to LL :  lptr R[10]

Input : n

first you have to cin >> n

Then read cin >>    n numbers .

output : you cout << the  n sorted numbers .

OTC-28-9-2020

**Flatten the Multi level lis**t to a Singly Linked List:

Example :

Input : ML:       3 – 4 – 7 – 18 – 45 – 36

                      |     |                    |

                      8    5                 6

                      |                         |

                      9                       2

Input ( cin >>) : 3 1 8 9 -1 4 1 5 -1 7 0 18 0 45 1 6 2 -1 36 0 -2

( 1 – indication of down link,  0 – no down link

  -1 – termination of down LL ,  -2 termination of  ML

Output (cout <<) :  ( 3 8 9 ) ( 4 5 ) ( 7 ) (18 ) ( 45 6 2 ) ( 36 )

* [**Submitted version (1 passed of 1)**](https://www.onlinegdb.com/s/as/13540)

### DLL- Add/Delete-Before-After-End-Sort

### Doubly Linked List

### Create a DLL and perform the following(sequence of) operations :

#### **Create DLL  ( cin >>  )   6 9 5 3 7 2 -1**

#### **print DLL ( cout << )  6 9 5 3 7 2**

#### **addfront(D, k )  : ( cin >> k )  8**

#### **print DLL (cout << ) 8 6 9 5 3 7 2**

#### **addend(D,k) : ( cin >> k )  4**

#### **print DLL (cout << ) 8 6 9 5 3 7 2 4**

#### **addbefore(D, x, y ) : ( cin >>x>>y)  1 3**

#### **print DLL (cout << ) 8 6 9 5 1 3 7 2 4**

#### **addafter(D, x, y ) : ( cin >>x>>y)  45 7**

#### **print DLL (cout << ) 8 6 9 5 1 3 7 45 2 4**

#### **delfront(D)**

#### **print DLL (cout << ) 6 9 5 1 3 7 45 2 4**

#### **delend(D)**

#### **print DLL (cout << ) 6 9 5 1 3 7 45 2**

#### **del(D,k) : ( cin >> k )  5**

#### **print DLL (cout << ) 6 9 1 3 7 45 2**

#### **sort(D) : any sorting method is allowed,** quick sort is preferable

#### **( should not use an array, have to sort DLL in place)**

#### **print DLL (cout << ) 1 2 3 6 7 9 45**

* [**Submitted version (Compile Error)**](https://www.onlinegdb.com/s/as/13641)

### DLL - sorts MLL

### Use DLLto sort the Multi level list

#### You should not use Linked list , arrays, Stacks, Queues and no sorting method. You must have to go through the MLL  ONLY ONCE  and print the sorted order.

### ( You can use one Doubly Linked List  only)

Example :

Input :MLL:       3   –    7 –       10   –  36

                      |          |             |

                      8        12         18

                      |           |            |

                      15       45       72

Output: LL :    3 –  7– 8 – 10 – 12 – 15 – 18 – 36 – 45– 72

Input ( cin >>) : 3 1 8 15 -1 7 1 12 45 -1 10 1 18 72 -1 36 0 -2

( 1 – indication of down link,  0 – no down link

  -1 – termination of down LL ,  -2 termination of  ML

Output (cout <<) :   3  7 8  10  12  15 18  36  45 72

* [**Submitted version (1 p**](https://www.onlinegdb.com/s/as/13647)

### CSR - Course-Student-Reg-MLL

### Course-Student-Reg-MLL

#### **﻿Create a CSR-MLL  with three nodes of course , student , reg as explained in the class.**

#### **You can take liberty in designing the node structures, still circularity is preferred.**

#### **Assume there are 4 courses and 5 students.**

#### **input : cin >> ( student rno  course name )   2 C 3 A ........ -1 #  ( -1 # is termination of input of registrations)**

#### **Construct the CSR - MLL**

#### **input : cin >>  (student rno  )   example 3**

#### **output : cout << print all the courses taken by student 3 with a single space**

#### **example :    A C D**

#### **input : cin>> ( student rno ) example 5**

#### **output : cout <<   print all the courses taken by student 5 with a single space**

input :cin >>  ( course name ) example B

#### **output : cout <<   print all the students who have taken course B**

example : 1 3 4

input : cin >> ( course name ) example D

#### **output : cout << print all the students who have taken course D**

#### Test case :

#### Input :

#### **1 A 2 B 4 A 3 A 5 B 1 C 2 D 4 C 3 C 5 C 4 D -1 #**

#### **4**

#### **3**

#### **A**

#### **D**

#### **output :**

#### **A C D**

#### **A C**

#### **1 3 4**

#### **2 4**

* [**Draft version**](https://www.onlinegdb.com/s/as/13837)

### DLL - Triplet

### DLL - Triplet    s = 2x + y + z

### Create a sorted doubly linked list of distinct nodes (no two nodes have the same data) with given data values and read a value s.

**Care fully observe the output and accordingly implement the logic.**

Print the triplets ( x y z )  in the list that sum up to s = 2x + y + z

Example DLL  1 ßà 2 ßà 3 ßà 4 ßà 5 ßà 6 ßà 7 ßà 8 ßà 9

s = 17

triplet  :  1 7 8   ( 2\*1 + 7 + 8 = 17 )

Input : cin >>    ( till -1 )

Input : cin >> s

Output : cout << x y z     ( print all possible triplets )

test case input :

1 2 3 4 5 6 7 8 9 -1

19

test case output :

1 8 9

2 6 9

##### **2 7 8 3 4 9 3 5 8 3 6 7 4 2 9 4 3 8 4 5 6 5 1 8 5 2 7 5 3 6 6 2 5 6 3 4 7 1 4 7 2 3 8 1 2**

CLL- Circle’s –circles

**CLL- Circle’s –circles**

Construct a Circular Linked List (CLL) with given data.

Print all the circles as shown in the test case output.

 4 à 3 à 8 à 7 à 2 à 9 à 4 à 3 à 4 à 6 à 2

 á\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ û

Test case Input : 4 3 8 7 2 9 4 3 4 6 2 -1

Output :              4 3 8 7 2 9 4

                           4 3 4

                           4 6 2 4

                           3 8 7 2 9 4 3

                           3 4 6 2 4 3

                           2 9 4 3 4 6 2

                           2 4 3 8 7 2

Note: **Either iterative or recursive (preferable)**

CLL - Josephus problem- V1

**Josephus problem- V1**

Given a group of **n** men arranged in a circle under the edict that every **m-th** man will be executed going around the circle until only one remains.

Order of arrangement :  **6, 5, 4, 3, 2, 1, 0**      given **n = 7 , m = 3**

For example, if 6, 5, 4, 3, 2, 1, 0 are arranged in circle, with m = 3,

then the order of removal is 4, 1, 5, 0, 2, 6, with 3 remaining as the survivor.

4, 1( in the first round)

5, 0  ( in the second round)  
2, 6  ( in the third round ) 

Code to construct a**Circular Linked List**  and to print the sequence of removals

for a given  n, the order of numbers and m.

**Sequence of cins, couts**:

               input :   cin >>   n

                             cin >>        ( n numbers )

                             cin >>  m

output :

 cout <<   ( sequence of removals)

test case input :

**7**

**6 5 4 3 2 1 0**

**3**

test case output :

**4 1 5 0 2 6**

CLL - JP-V2 - DGG

**JP-V2 - DGG**

A teacher plays the game “Duck-Goose-Goose” with his class. The game is played as follows: All the students stand in a circle and the teacher walks around the circle. As he passes each student, he taps the student on the head and declares her a ‘duck’ or a ‘goose’. Any student named a ‘goose’ leaves the circle immediately. Starting with the first student, the teacher tags students in the pattern: duck, goose, goose, duck, goose, goose, etc., and continues around the circle (re-tagging some former ducks as geese) until only one student remains. This remaining student is the winner.

For example, if there are 8 students, the game proceeds as follows: student 1 (duck), student 2 (goose), student 3 (goose), student 4 (duck), student 5 (goose), student 6 (goose), student 7 (duck), student 8 (goose), student 1 (goose), student 4 (duck), student 7 (goose) and student 4 is the winner.

Code to create a CLL and given the number of students n, print the winner.

cin >> n

cout << winner

test case :

input   : 8

output :  4

CLL - JP - V3- AB

**JP - AB**

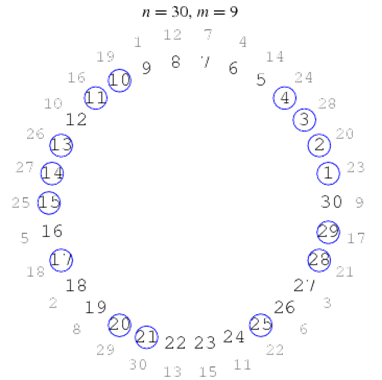
Consider a circle of two groups (say, "A" and "B") of 15 men each (giving a total of 30 men), with every ninth man cast overboard, illustrated bel. To save all the members of the "A" group, the men must be placed at positions 1, 2, 3, 4, 10, 11, 13, 14, 15, 17, 20, 21, 25, 28, 29. Written out explicitly, the order is



Code to create a CLL and print the order of As , Bs

test case : cout <<  As, Bs.

AAAABBBBBAABAAABABBAABBBABBAAB



### Multi-Index-Lists-CSR

### Multi-Index-Lists-Course-Student-Course-Reg

#### **﻿Create Multi Index Lists with three nodes of student, course , reg as explained in the class on 1-10-2020**

#### **Use the structure of stdnode as rno, slptr , structure of coursenode as cname, clptr**

#### **regnode as rrno , sfp, rcname , cfp , next.**

#### **( if you wish you can use sbkp , cbkp as backword pointers for further extensions of CSR)**

#### **Assume there are 4 courses and 5 students.**

#### **input : cin >> ( student rno  course name )**

#### **2 C 3 A ........ -1 #  ( -1 # is termination of input of registrations)**

#### Construct  stdindextable ( array of stdnode) , courseindextable ( array of coursenode) and

#### R ( linked list of all registrations of regnode and adjust pointers accordingly

#### **input : cin >>  (student rno  )   example 3**

#### **output : cout << print all the courses taken by student 3 with a single space**

#### **example :    A C D**

#### **input :cin >>  ( course name ) example B**

#### **output : cout <<   print all the students who have taken course B**

example : 1 3 4

input : cin >> ( course name1 , course name2 )

example  A C

#### **output : cout << print all the students who have taken both courses A C**

input : cin >> ( student rno1 , student rno2 )

output : cout << print all the common courses taken by rno1 and rno2

output : cout << print all the registrations

**Test case :**

#### Input :

#### **1 A 2 B 4 A 3 A 5 B 1 C 2 D 4 C 3 C 5 C 4 D 3 D 5 A -1 #**

#### **5**

#### **D**

#### **C D**

1 5

#### **output :**

#### **B C A**

#### **2 4 3**

#### **3 4**

A C

1 A 2 B 4 A 3 A 5 B 1 C 2 D 4 C 3 C 5 C 4 D 3 D 5 A

Lab-Assignment- 3-10-2020

**Implement Creation of Doubly Linked List and operations on it.**

Write main() function to add two large numbers as given blow.

 Create two DLLs for representing the two numbers ( digits of  a number as data value of  a node)

Add the two numbers and store the sum in  third DLL and print it.

In your program you should have cins , couts as below :

Test case input :  cin >>     587624319269#

                            cin >>               8978196#

output                cout <<    587633297465

### CCC- T1- 3 -10-2020

### Create a Multi-Level List  and Print it as given.

#### **{ A B { C D E } F { G { H I J } K } L M }**

#### **output  as given is :  { A B [ C D E ] F ( G < H I J > K ) L M }**

test case :

#### **input :     {AB{CDE}F{G{HIJ}K}LM}#**

#### **( cin >>   a char  till '#' )**

#### **output : { A B [ C D E ] F ( G < H I J > K ) L M }**

#### 

#### **( cout <<  observe one ' ' after each character in the output)**

CCC - T2 - 3-10-2020

**Common elements among n Linked lists of a Linked List**

Create a Linked list L, whose elements as linked lists with data value as both char and int.

Read(and create) **Three** linked lists as the elements of L

 ( L also gets created).

**Many of your logic is almost WRONG.               You should not use any sorting methods.**

**You have to use logic for even there can be 10 linked lists**

Print all the node data values of all the linked lists of L as shown.

Find and print the **common node data values**of the **three linked lists** of L.

Note : You **should not use brute force method** of taking one node of a LL and comparing it with all other nodes of other LLs.

Example :

L  = {  [ ‘A’ 3 , ‘C’ 9 , ‘N’ 5 , ‘S’  6]  , [ ‘B’  5,  ‘S’ 6 , ‘L’ 2, ‘C’ 9 ],

           [ ‘L ‘ 7 , ‘J’ 8 , ‘N’ 2, ‘A’ 3 , ‘C’ 9 , ‘B’ 9, ‘S’ 6 , ‘E’ 4 ] }

Common elements :

                  ‘C’ 9  , ‘S’ 6

Test case :

**Input**  cin >>  :   A3 C9 N5 S6 #-1

                            B5 S6 L2 C9# -1

                            L7 J8 N2 A3 C9 B9 S6 E4 #-1

**Output**:   cout <<   :  A3 C9 N5 S6 B5 S6 L2 C9 L7 J8 N2 A3 C9 B9 S6 E4

                 cout <<   :  C9 S6

CCC - T3 - 3 -10 - 2020

**Link the Links**

Create two sorted linked lists L1 and L2.

**Create** another linked list **SL** as show in the figure, which **links the nodes of L1, L2** in a sorted order.

**NOTE : Look at Email to see the figure**

**And You must have to use proper Definitions of Structures**

**Print SL to get sorted values**of data from L1 and L2.

Test case input cins >> :

2 3 9 12 25 48 54 72 -1

4 6 8 15 18 20 36 40 63 67 80 90 -1

Output : cout << :

2 3 4 6 8 9 12 15 18 20 25 36 40 48 54 63 67 72 80 90

M1- Q5 - Common Elements in n LLs

**Top three in frequency(count) of Common elements among n Linked lists**

**(final timeline till 5.25PM)**

Create  **n**   linked lists with data value as  int.

Find and print the  **Top three  frequency(count) of common node data values**of the   **n   linked lists**

**You should not use any sorting methods.**

**You have to use approriate logic of O(n)  : You should not use two loops of O(n\*n)**

Note : You **should not use brute force method** of taking one node of a LL and comparing it with all other nodes of other LLs.

Example :

L1  =   [  3 , 9 , 5 ,  6 ]

L2  =   [ 5,  6 ,  2,  9 ]

L3  =   [  7 , 8 ,  2,  3 ,  9 ,  6 ,  4 ]

L4  =   [  1, 8, 9, 3, 5, 6 ]

L5  =   [   9, 7,  1, 4 ]

Top three count(frequency) of Common elements :

9 5

6 4

5 3

3 3

Test case :

**Input**

cin >> n   :    5

cin >>  :

3 9 5 6 -1

5 6 2 9 -1

7 8 2 3 9 6 4 -1

1 8 9 3 5 6 -1

9 7 1 4 -1

**Output**:   cout <<   :

9 5

6 4

5 3

3 3

M1- Q6 - MLL - LLM

**Convert (reverse ) MLL as Required.**

**(final timeline till 5.30PM)**

Example Given MLL:

               10 – 5 – 12  –  7 – 18

                |                        |

               4 – 20 – 9         25 – 6

                      |      |            |

                     3     63         2 – 8 – 54

                             |            |

                           72          47 – 68

                                           |

                                          36

**Convert( reverse) as below :**

10 –  4

  |       |

 5     20 – 3

 |        |

12     9 – 63 – 72

 |

7 – 25 – 2 – 47 – 36

 |      |      |      |

18   6     8    68

              |

             54

**Input**: 10 1 4 0 20 1 3 0 -1 9 1 63 1 72 0 -1 -1 -1 5 0 12 0 7 1 25 1 2 1 47 1 36 0 -1 68 0 -1 8 0 54 0 -1 6 0 -1 18 0 -1

( 1 for down link, 0 for no down link , -1 for end of next link )

Output1 : Level–wise : 10 4 5 20 3 12 9 63 72 7 25 2 47 36 18 6 8 68 54

Output 2 : Depth–wise:  10 5 12 7 18 25 6 2 8 54 47 68 36 4 20 9 63 72 3

 First cout << Level wise  ,               next  cout < Depth wise

(Input : 10 1 4 0 20 1 3 0 -1 9 1 63 1 72 0 -1 -1 -1 5 0 12 0 7 1 25 1 2 1 47 1 36 0 -1 68 0 -1 8 0 54 0 -1 6 0 -1 18 0 -1 )

BST Levels - Labels of Novel Intellections

**Binary Search Tree – Levels**

Create a binary tree with given input data.

Print all the following level orders.

Input data: 5 8 3 4 1 9 6 7 2 -1

Output level orders :

**1.** Level order :   5 3 8 1 4 6 9 2 7

**2.** Reverse reversed level order:

7 2 9 6 4 1 8 3 5

**3.** Reverse level order :

2 7 1 4 6 9 3 8 5

**4.** Squirrel level order (clock-wise start):

5 8 3 1 4 6 9 7 2

**5.** Squirrel level order (anti-clock-wise start):

5 3 8 9 6 4 1 2 7

**6.** Alternate odd-level reversal level order :

5 8 3 1 4 6 9 7 2

**7.** Alternate even-level reversal level order:

5 3 8 9 6 4 1 2 7

Test case:

input ( cin >> ) : 5 8 3 4 1 9 6 7 2 -1

output ( cout >>) :

5 3 8 1 4 6 9 2 7

7 2 9 6 4 1 8 3 5

2 7 1 4 6 9 3 8 5

5 8 3 1 4 6 9 7 2

5 3 8 9 6 4 1 2 7

5 8 3 1 4 6 9 7 2

5 3 8 9 6 4 1 2 7

BST - Paths - of Success

**Binary Search Tree – Paths**

Create a binary tree with given input data.

Print all the following types of Paths.

Input data: 5 8 3 4 1 9 6 7 2 -1

Output Paths:

**1.**All paths - Root to leaf : 5 3 1 2

                                            5 3 4

                                            5 8 6 7

                                            5 8 9

**1.** Leftmost path :   5 3 1 2

**2.** Rightmost path : 5 8 9

**3.** Maximum sum path:  5 8 6 7

**4.** Minimum sum path : 5 3 1 2

**5.** Path with sum s = 22  :  5 8 9

**6.** Path with maximum even numbers : 5 8 6 7

**7.**  Path with maximum odd numbers : 5 3 1 2

Test case:

input ( cin >> ) : 5 8 3 4 1 9 6 7 2 -1

output ( cout >>) :

5 3 1 2

5 3 4

5 8 6 7

5 8 9

5 3 1 2

5 8 9

5 8 6 7

5 3 1 2

5 8 9

5 8 6 7

5 3 1 2

Lab-10-10-2020-BST post order - levels -lines

.**Binary Search Tree – Create from Postorder**

Create a binary tree with given input data of post order traversal of it.

**You must have to create with LRD . Not just reverse reading of  input.**

Then print all levels line by line.

Input data:  2 1 4 3 7 6 9 8 5 -1

Output level line by line :

5

3 8

1 4 6 9

2 7

### Lab-10-10-2020 - BT-Levels - Max, Min, Ascending , Descending, Unsorted

#### BT-Levels - Max, Min, Ascending , Descending, Unsorted

Create a Binary Tree with given input data values of integers.

Implement the logic in  a**Single function**to print the following:

Input  : 5 2 7 0 0 4 9 0 0 6 0 0 8 3 0 15 0 0 1 0 12 0 0

Output :

1. Maximum of all levels : 15

2. Minimum of all levels  : 1

3. Ascending level numbers : 1

4. Descending level numbers : 2

5. Unsorted level numbers : 3

Lab-10-10-2020-BST post order - levels -lines

.**Binary Search Tree – Create from Postorder**

Create a binary tree with given input data of post order traversal of it.

**You must have to create with LRD . Not just reverse reading of  input.**

Then print all levels line by line.

Input data:  2 1 4 3 7 6 9 8 5 -1

Output level line by line :

5

3 8

1 4 6 9

2 7

### Lab-10-10-2020 - Correct to be BST

**Adjust the misplaced two nodes to make a BST**

Create a Binary tree of integers with given data.

In that Binary tree, two nodes are misplaced, swap those two nodes to make in BST.

**You must have to traverse the tree only once ( touch each node only once)**

test case :

input : 5 3 1 0 2 0 0 6 0 0 8 4 0 7 0 0 9 0 0    ( 0 for Null node)

output( in order) : 1 2 3 4 5 6 7 8 9

### Lab-10-10-2020 - BST in BT

**Find largest BST in given BT**

Create a Binary tree with given data of integers.

Find the largest Binary Search Tree in the BT.

input :  27 18 15 9 0 0 0 22 12 8 0 0 0 45 32 0 0 72 0 0 36 25 0 0 63 54 0 0 0

output : In order of largest BST : 8 12 22 32 45 72

### BST-Two to Third Sum

**Sum of Two nodes to a Third Node :**

Create Two Binary Search Trees pointed by T1, T2.  
Find and print the node values  of either T1 of T2 such that sum of a node value  from T1 and a node value from T2 is equal to this third node. ( Don't use Brute Force, You may have to traverse T1, T2 only once or at most two times)

Example : T1 is : 5 3 8 1 7   and  T2 is : 6 4 10 2 9

output : ( 1 , 9 , 10 )  ,  ( 1, 6 , 7 ) , ( 1 , 4 , 5 ) , ( 1, 2, 3) , ( 3, 6, 9) , (3,4,7) , (3,2,5) , ( 5,4,9) , ( 5 , 2, 7), ( 7, 2, 9), (8,2, 10).

test case Input : 5 3 8 1 7 -1

                          6 4 10 2 9 -1

Test case Output : As many ways of solving, the sequence of outputs differ.

If you have not used Brute-Force method, and have used an elegant method and have got the above outputs

then send an email with the screen short of output. Submit code here in GDB.

### BT- Max node in Two Nodes Path

**Maximum value Node in the path of two Nodes :**

Create a Binary Tree with given data.

Find the maximum node value in the path of given two nodes.

Input  for Tree creation : 5 2 7 0 0 4 9 0 0 6 0 0 8 3 0 15 0 0 1 0 12 0 0

Input two Nodes : 15 12   output : 8    Input two Nodes : 7 3 output : 8  Input two nodes : 7 8 output : 5

Test case :

Input : 5 2 7 0 0 4 9 0 0 6 0 0 8 3 0 15 0 0 1 0 12 0 0

Input :  15 12

             7 3

             7 8

Output : 8 8 5

### Lab-Assignment-Q- 15 - 10 -2020

**Construct a Binary Tree** from its given  Level order serialized string.

Input string : NITC#SE#ABW$      ---  ( # for NULL , $ for end of input )

print the tree level by level.

test case : Input : NITC#SE#ABW$

out ut :

N

I T

C S E

A B W

BT - Boundary Nodes -15-10-2020

**Binary Tree Boundary Nodes**

Construct Binary Tree and print the boundary nodes of the tree.

Input : 5 2 7 0 0 4 9 0 0 6 0 0 8 3 0 15 0 0 1 0 12 0 0

Output : 5 2 7 9 6 15 12 1 8 5

### BT-Blends of bends -15-10-2020

**Path of Maximum Bends:**

Construct a Binary tree.

Find and print  the path , that having maximum number of bends.

A bend could be left to right or right to left.

Input : 5 2 7 0 0 4 9 0 0 6 0 0 8 3 0 15 0 0 1 0 12 0 0

Output  (the paths of maximum(2) bends) :

              5 2 4 9

              5 8 3 15

BT - Hidden Nodes -15-10-2020

**Binary Tree Hidden Nodes**

Construct Binary Tree and print the hidden nodes of the tree.

Input : 5 2 7 0 0 4 9 0 0 6 0 0 8 3 0 15 0 0 1 0 12 0 0

Output : 4 3

### SSS-PPP-BT-Hidden View

**Hidden View** of a Binary Tree :

Given Binary Tree input as Serialized string, construct the tree and

print the hidden nodes values as given below:

input : MJPB###TE###LKS###QF#G###

output : T K F

### SSS-PPP-BT-Path of LCA

**Path from Root to LCA of two nodes**of a Binary Tree :

Given Binary Tree input as Serialized string, construct the tree and

print the path from root to LCA of given two nodes:

input : MJPB###TE###LKS###QF#G###

two nodes : BE

output : M J

### SSS-PPP-BT-Braced Print

**BT- Braced Print**

Given a serialized input string of Binary tree,

construct the tree and print it in Braced form as shown below:

input : ABD##EH###CFI##J##G##

output : A(B(DE(H))C(F(IJ)G))

### SSS-PPP- BST level to preorder

**BST - construct from Level order**

Construct a BST with given level order .

Print it in preorder.

Level order input : 5 2 8 1 4 7 9 3 6 -1

Preorder output    : 5 2 1 4 3 8 7 6 9

Lab-6-11-2020 Segment Tree -1

**Segment Tree**

Write code to answer the following queries and update the values.

Input format:

First line contains an integer N.(number of elements in array)

Second line contains N numbers of array.

Third line contains an integer Q (number of queries)

Next q lines contains three integers each of following pattern:

0 i x : Update i’th index element to x

1 l r: print sum of range from l to r (l & r including)

2 l r: print minimum of range from l to r(l & r including)

3 l r:print average of range from l to r(l & r including).

Output format:

For queries with 1 or 2 or 3 print the required answer.

Test case :

Input:

9

9 3 7 2 5 4 3 1 2

7

1 2 7

2 2 7

3 4 8

0 7 99

1 6 7

2 6 8

3 7 8

Output:

24

2

3

103

1

50

Lab-6-11-2020 Segment Tree-2

**All-in-One Segment Tree**

Code a Segment Tree which constitutes of all min, max, sum, average in a single tree.

You should use only a single tree and it should consist of all the parameters (min, max, sum, avg) of given range. (HINT: Try storing all the parameters in a single node of segment tree).

Input format:

First line contains an integer N.(number of elements in array)

Second line contains N numbers of array.

Third line contains an integer Q (number of queries)

Next Q lines contains three integers each of following pattern:

0 i x : Update i’th index element to x and update the segment tree

1 l r: print minimum of range from l to r(l & r including)

2 l r: print maximum of range from l to r(l & r including)

3 l r: print sum of range from l to r (l & r including)

4 l r:print average(float) of range from l to r(l & r including).

Output format:

For queries with 1 or 2 or 3 or 4 print the required answer.

Test case:

 input:

10

1 2 3 4 5 6 7 8 9 10

12

1 2 7

2 3 9

3 2 6

4 0 5

0 6 13

0 2 7

0 9 12

0 0 17

1 0 4

2 5 9

3 2 7

4 6 9

output:

3

10

25

3.5

2

13

43

10.5

Segment Tree-3

**Commonality SGTs**

Print all the possible ranges [l,r]  of given two arrays A1 and A2, which satisfies the following condition.

The min of range [l,r] in array A1 should be same as min of range [l,r] in array A2.

And also l and r should not be same i.e  l != r .

Code efficient way.

Example:

Input:

5

3 5 7 8 10

5

3 11 1 12 8

Output:(Any order is allowed)

0 0    min is same and it is 3

0 1    min is same and it is 3

3 4    min is same and it is 8

Test Case:

input:

7

3 4 1 7 11 2 10

7

15 3 10 2 11 4 18

output :(any one of the following, enough to get one test case passed)

0 1

3 5

3 6

\_\_\_\_\_

0 1

3 6

3 5

\_\_\_\_\_\_

3 5

0 1

3 6

\_\_\_\_\_\_\_

 3 5

3 6

0 1

\_\_\_\_\_\_\_\_

3 6

3 5

0 1

\_\_\_\_\_\_\_\_

3 6

0 1

3 5

Segment Tree - 4

**Segment Tree - String**

Code to answer the following queries and update the values.

Input format:

First line contains a string ( terminated by #)

Second line contains Q(number of queries).

Next Q lines contain input from each of the following pattern:

1 i x : Update i’th index character of string s to x.(x is a character)

2 l r: Calculate number of distinct characters in the substring[l:r] (both l and r included).

Output format:

For queries with type 2, print the required answer.

Test case:

Input:

dfcbbcfeeedbaea#

17

1 6 e

1 4 b

2 6 14

1 7 b

1 12 c

2 6 8

2 1 6

1 7 c

1 2 f

1 10 a

2 7 9

1 10 a

1 14 b

1 1 f

2 1 11

1 14 e

2 14 15

 Output:

5

2

5

2

6

2

Interval Tree - 1 - Lab-6-11-2020

**Interval Tree**

 Every node of Interval Tree stores following information.

a) **i**: An interval which is represented as a pair *[low, high]*  
b) **max**: Maximum *high*value in subtree rooted with this node ,

The max value of a node will be given as input first time. later on it gets updated depenging on other input maxximu values.

**The Maximum high should get updated depending on nodes insertion.**

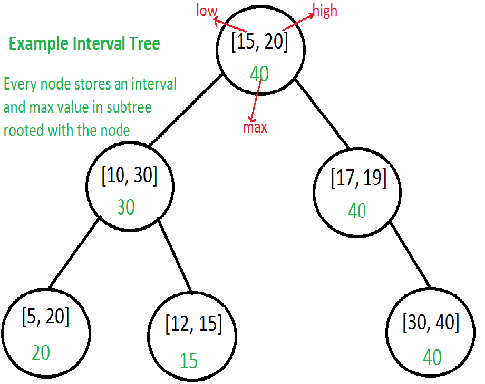
The low value of an interval is used as key to maintain order in Binary Search Tree.

The insert and delete operations are same as in BST.

Operations to be implemented are :

**1)**Add an interval  
**2)** Given an interval x, find if x overlaps with any of the existing intervals.

Example:



Interval Tree :

[5, 20]  max = 20

[10, 30]  max = 30

[12, 15] max = 15

[15, 20] max = 40

[17, 19] max = 40

[30, 40] max = 40

Searching for interval [6,7]

Output :

Overlaps with [5, 20]

test case

input :

1 15 20 20

1 10 30 30

1 12 15 15

1 5 20 20

1 17 19 40

1 30 40 40

2 6 7

2 32 35

output

5 20

30 40

### RCC-General Tree - 9-11-2020

Construct a general tree with given input and print it in level order.

input :

ABC.D.E..FG.H..JK.L.MP.Q..N....

output :

A

B F J

C D E G H K L M N

P Q

### Flow Tree

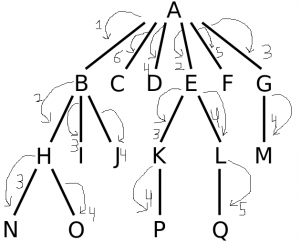
**Flow Tree**

Given a very large n-ary tree. Where the root node has some information which it wants to pass to all of its children down to the leaves with the constraint that it can only pass the information to one of its children at a time (take it as one iteration).

Now in the next iteration the child node can transfer that information to only one of its children and at the same time instance the child’s parent i.e. root can pass the info to one of its remaining children. Continuing in this way you have to find the minimum no of iterations required to pass the information to all nodes in the tree.

Minimum no of iterations for tree below is 6. The root A first passes information to B.

 In next iteration, A passes information to E and B passes information to H and so on.



input :  ABHN.O..I.J..C.D.EKP..LQ...F.GM....

output :  6

GT - BT ( General Tree to Binary Tree )

**GT to BT :**Split a give general tree into a set of binary trees.

Input1:  ABFI.J..SK..H..C.DLM.N..P.Q..E...

Output1:

ABFI..J..SK...C..

ABH...DLM..N..P..

ADQ...E..

Input2 : ABFI.J..SK..H.O..C.DLM.N..P.Q..E...

Output2:

ABFI..J..SK...C..

ABH..O..DLM..N..P..

 ADQ...E..

IDS

**Iterative Deepening Search(IDS) or Iterative Deepening Depth First Search(IDDFS)**

There are two common ways to traverse a graph, BFS and DFS. Considering a Tree (or Graph) of huge height and width, both BFS and DFS are not very efficient due to following reasons.

1. **DFS** first traverses nodes going through one adjacent of root, then next adjacent. The problem with this approach is, if there is a node close to root, but not in first few subtrees explored by DFS, then DFS reaches that node very late. Also, DFS may not find shortest path to a node (in terms of number of edges).

2. **BFS** goes level by level, but requires more space. The space required by DFS is O(d) where d is depth of tree, but space required by BFS is O(n) where n is number of nodes in tree (Why? Note that the last level of tree can have around n/2 nodes and second last level n/4 nodes and in BFS we need to have every level one by one in queue).

**IDDFS** combines depth-first search’s space-efficiency and breadth-first search’s fast search (for nodes closer to root).

How does IDDFS work?

IDDFS calls DFS for different depths starting from an initial value. In every call, DFS is restricted from going beyond given depth. So basically we do DFS in a BFS fashion.

**Algorithm:**

// Returns true if target is reachable from

// src within max\_depth

**bool** IDDFS(src, target, max\_depth)

**for** limit **from** 0 **to** max\_depth

**if** DLS(src, target, limit) == **true**

**return** true

**return** **false**

**bool** DLS(src, target, limit)

**if** (src == target)

**return** **true**;

    // If reached the maximum depth,

    // stop recursing.

**if** (limit <= 0)

**return** **false**;

**foreach** adjacent i of src

**if** DLS(i, target, limit?1)

**return** **true**

**return** **false**

An important thing to note is, we visit top level nodes multiple times. The last (or max depth) level is visited once, second last level is visited twice, and so on. It may seem expensive, but it turns out to be not so costly, since in a tree most of the nodes are in the bottom level. So it does not matter much if the upper levels are visited multiple times.

Code the implementation of above algorithm.﻿

input : // data is of  type int, 0 for null

50 49 47 33 0 43 35 0 34 0 0 42 32 0 0 0 0 46 41 31 0 30 0 0 40 29 0 28 0 0 0 48 45 39 27 0 26 0 0 38 25 0 24 0 0 0 44 37 23 0 22 0 0 21 0 36 20 0 0 0 0 0 0

3

21

35

19

output

1

1

0

### RCC - GT - 1 - 10 - 11- 2020

Create the same general tree of yesterday according to input format of.

Node data, number of children. Then print serialized order.

input : A 3 B 3 C 0 D 0 E 0 F 2 G 0 H 0 J 4 K 0 L 0 M 2 P 0 Q 0 N 0

output : ABC.D.E..FG.H..JK.L.MP.Q..N....

### RCC- General Tree - 2 - 10 -11-2020

Create the same general tree of yesterday according to input format of.

Node data, number of children level wise

Then print serialized order.

input : A 3 B 3 F 2 J 4 C 0 D 0 E 0 G 0 H 0 K 0 L 0 M 2 N 0 P 0 Q 0

output : ABC.D.E..FG.H..JK.L.MP.Q..N....

### M-way Tree - Lab

**M - Way Tree v1 , v2**

﻿Create m-way trees of version 1 and version 2.

Then print them in inorder. (sorted sequence)

input1 ( for version 1 tree)  : 4  ( m = 4)

18 36 27 9 54 45 50 63 20 47 72 0 // 0 for end of input

input2 ( for version 2 tree ) : 4 ( m = 4)

18 36 27 9 54 45 50 63 20 47 72 0 // 0 for end of input

output1 : 9 18 20 27 36 45 47 50 54 63 72

output2 : 9 18 20 27 36 45 47 50 54 63 72

test cases look like :

input :

4

18 36 27 9 54 45 50 63 20 47 72 0

4

18 36 27 9 54 45 50 63 20 47 72 0

output :

9 18 20 27 36 45 47 50 54 63 72   
9 18 20 27 36 45 47 50 54 63 72

### M-way Tree : Level Order Construction

**M - way Tree V2 construction**

﻿Construct m-way tree of version2 with given level order input.

i) Then print in level order line by line.

ii) Delete given two key values k, then again print in level order line by line.

test case

input :

3   //  m = 3

18 54 6 8 25 45 63 81 1 3 10 12 30 36 72 75 90 -1   // -1  for end of input

25  // k = 25 is to be deleted

54  // k = 54 is to be deleted

output:     // after creation print line by line

18 54

6 8 25 45 63 81

1 3 10 12 30 36 72 75 90

output:         // after deleting 25 , 54  print line by line

18 63

6 8 30 45 72 81

1 3 10 12 36 75 90

Quaternary Heap

﻿**Quaternary heap**

﻿A quaternary heap is a min heap stored with a complete quaternary tree(four child ptrs): each node has up to four children and the nodes are filled in a breadth-first traversal order( i.e. top to bottom left to right). Parent node data value is minimum of four children node data values.

An example of a complete quaternary tree is shown in Figure



  A complete ternary tree with 59 nodes.

Given input construct a quaternary min heap and print it level by level.

It is to be noted that you must use " complete tree construction...

Th tree node struct must be "

**int data ;**

**struct qtnode \*cptr[4] ;**

Others notations and arrays are not accpeted.

You even try to code for a complete tree heap for any number of children.

Test case :

input

9 5 12 3 10 6 8 14 11 4 7 1 13 15 2 -1   // -1 for end of input

output           // print level by level

1

6 3 2 10

9 8 14 11 12 7 4 13 15 5

M-way Tree : Inorder Order Construction

**M - way Tree V2   a fully complete tree construction**

﻿Construct m-way tree of version2  which is fully complete with given inorder input.

Then print in level order line by line.

input :  // inorder

3    //value of m

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 -1 //inorder sequence.

Output: //level order

9 18

3 6 12 15 21 24

1 2 4 5 7 8 10 11 13 14 16 17 19 20 22 23 25 26

### Best - First Tree

**B- Tree** :

Create a B - tree of order 3 for input 1 to 10 .

Then print in inorder.

test case :

Input : 10 9 8 7 6 5 4 3 2 1

output 1 2 3 4 5 6 7 8 9 10

Best - First Tree2

**B- Tree order 5** :

Create a B - tree of order 5 for given input

Then print in inorder.

test case :

Input :

5  // cin >> d      .... order of B-tree

22 // cin >> n     .... number of keys

 4 7 22 21 35 12 23 39 16 32 29 46 28 43 64 9 49 53 40 42 24 60

output

4 7 9 12 16 21 22 23 24 28 29 32 35 39 40 42 43 46 49 53 60 64

### Lab-13 -11- 2020 - M-way Tree - Left view - Right view

**﻿M-way Tree - Left view - Right view**

Create m-way tree of  version 1.

Then print the left view and right view of it.

test cases look like :

input : 4 ( m = 4)

18 36 27 9 54 45 50 63 20 47 72  46 48 49 0 // 0 for end of input

output1  Left view: 18 9 20 46

output  Right view: 54 72 49 46

### Lab - 13-11-2020 M-way tree bottom-up alternate reverse print

**M-way tree bottom-up alternate reverse print**

Create m-way tree of  version 2.

Then print bottom-up reverse of it.

test cases look like :

input : 4 ( m = 4)

18 36 27 9 54 45 50 63 20 47 72  46 48 49 0 // 0 for end of input

output : 48 72 63 49 47 46 9 20 45 50 54 36 27 18

B-Tree -LL - Linked Leaves

**B- Tree -LL - Linked leaves** :

Create a B - tree of order 3 for input 1 to 10 by

linking the leaf nodes. That means the right most pointer of a leaf node should link to the right side leaf.

Then print all the leaf nodes contents only as if printing a linked list.

(i.e. travers to the leftmost leaf node then follow the links to print the contents.)

test case :

Input : 10 9 8 7 6 5 4 3 2 1

output   1 2 6 8 10        // all linked leaf nodes contents

### B - Tree Left biased Right biased

**B - Tree Left biased Right biased**- **4 way.**

 Create a B-tree of 4 way .

First create as Left biased and print level order.

Next create as Right biased and print level order.

Testcases..

input : 10 9 8 7 6 5 4 3 2 1

output1 : left biased level order

2 5 8  
1 3 4 6 7 9 10

output2 : right biased level order

7  
3 5 9  
1 2 4 6 8 10

B -Tree Diwali Dhamaka

**B- Tree -DL - Doubly Linked leaves** :

Create a B - tree of order 3 (3 way) for input given by doubly linking the leaf nodes.

That means the right most pointer of a leaf node should link to the right side leaf and the leftmost pointer should link to the left side leaf node.

i) First print it in level order

Then print all the leaf nodes contents only as if printing a Doubly linked list as

ii) Print left to right

(i.e. traverse to the leftmost leaf node then follow the links to print the contents as left to right)

iii) Print right to left

(i.e. traverse to the rightmost leaf node then follow the links to print the contents as right to left)

test case :

Input : 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 16 17 18 0 // 0 for end of input

output 1 : level order

8

4 12

2 6 10 14 16

1 3 5 7 9 11 13 15 17 18

output 2   :  1 3 5 7 9 11 13 15 17 18      // all linked leaf nodes contents left to right

output 2   :  18 17 15 13 11 9 7 5 3 1      // all linked leaf nodes contents right to left

B+ Tree ^ Diwali Delight ^

**B+ Tree ^ Diwali Delight ^**

A B+ \_Tree  is a B - Tree with few changes:

1. When a leaf node is split, then the median also will be stored in the new node along with its right side keys.

2. Splitting of a non leaf nodes is same as B-Tree.

3. All the keys will be there in leaf nodes.

Construct a B+ \_ tree of  order 3 ( 3- way) and print it in level order.

test case input :

10 9 8 7 6 5 4 3 2 1 0 // 0 is end of input

output :   // level order

7

3 5 9

1 2 3 4 5 6 7 8 9 10

B -Tree Levels connected - Diwali Special

**B- Tree -LC - Levels Connected** :

Create a B - tree of order 3 for given input as below:

Include an additional pointer(d+1) at end in the node structure.

Link all the nodes at each level from left to right with this additional pointer during construction of tree.

That means the additional right most pointer of a node should link to its right side node.

i) First, print nodes contents of each level as if printing linked lists.

(i.e. travers to the leftmost node of a level then follow the links to print the contents.)

ii) Next, form a single linked list of linking all levels from top to bottom, left to right and then print that single linked list.

test case :

Input : 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 16 17 18 0 // 0 for end of input

output 1 : level orders as printing linked lists

8

4 12

2 6 10 14 16

1 3 5 7 9 11 13 15 17 18

output 2 : level orders as printing a single linked list

8 4 12 2 6 10 14 16 1 3 5 7 9 11 13 15 17 18

B –Tree Diwali Celebration of Distribution

**B –Tree Diwali Celebration of Distribution**

Create a B-Tree of order 3 ( d=3 ways) as specified below.

When a node is Overflow then before splitting, look for the below **Key Distribution** options.

i) if the overflow node’s **immediate**left sibling is having less than 2 keys ( less than d-1 keys),

 then distribute the keys as: bring down the key , Ki of the parent node and

insert it into the **immediate**left sibling node(*the position of this key will be rightmost in this node*)  of overflow node,

and put the leftmost key of the overflow node as the Ki of parent node.

ii)if the **immediate** left sibling of the overflow node is full( with 2 keys, i.e. d-1 keys) , then look for the **immediate** rightmost sibling of the overflow node for a vacant place.

If the **immediate** rightmost sibling’s count<2 ( less than d-1 keys) ,then distribute the keys as :

bring down the key Ki of the parent node and insert it into the **immediate** right sibling node (*the position of this key will be leftmost in this node*) of overflow node ,

and put the rightmost key of the overflow node as the Ki of parent node.

ii) If  Key Distribution is not possible with immediate left and right siblings , then do Node Splitting as you have coded in B-Trees so far.

Explanation: step by step

input  -  15 , 9 , 8 , 7, 6 ,

Level order tree :

**. 8 .  .**

**. 6. 7.     .9.15.**        // key distribution with right sibling

input – 18 27 45 ,

level order of tree:

**. 8 .18 .**

**. 6. 7.     .9.15.**    **.27.45.     //**key distribution with left sibling

input – 3, 5 ,4

level order of tree :

**.8.**

**.5. .                    .18. .**

**.3.4.   .6.7.      .9.15.     .27.45.**

Test case

input :

15 9 8 7 6 18 27 45 3 5 4 0     // 0 for end of input

**output:** // level order

**8**

**5 18**

**3 4 6 7 9 15 27 45**

### B-Tree Siblings Distribution

**B-Tree Siblings Distribution :**

B-Tree – 3-way creation

Implement a modified version of the B-Tree insertion algorithm so that each time we create a new node because of a split of a node v, we can redistribute keys among all of v’s siblings.

test case :

input : 5 19 21 28 16 18 26 29 13 20 8 14 15 4 6 27 30 0

Output:  // level order

18

6 14 21 28

4 5 8 13 15 16 19 20 26 27 29 30

B-Tree Delete

**B-Tree Delete**

Implement B-Tree delete of order 5  without key distribution ( for test case below)which incudes all the cases.

Example : order 3 ( 3 way tree)

You can check manually as indicated below before passing the test case

case 1: Leaf nodes

Input : 5 10 15 20 30 25 31 40 50 45 55 33 35 60 65 28 32 0

simple b-tree **without**key distribution

output : level print  
20 40  
10 30 33 50 60  
5 15 25 28 31 32 35 45 55 65

delete 32

level print:

20 40  
10 30 33 50 60  
5 15 25 28 31 35 45 55 65

delete 31 (checking left sibling first for deleting)

level print

 20 40  
10 28 33 50 60  
5 15 25 30 35 45 55 65

delete 30

20 40

10 33 50 60

5 15 25 28 35 45 55 65

case 2: internal nodes

Input : 5 10 15 20 30 25 31 40 50 45 55 33 35 60 65 28 32 0

simple b-tree **without**key distribution

level print  
20 40  
10 30 33 50 60  
5 15 25 28 31 32 35 45 55 65

delete 33

level print

20 40

10 30 32 50 60

5 15 25 31 35 45 55 60

delete 30

level print

20 40

10 32 50 60

5 15 25 31 35 45 55 65

case 3: internal node with no distribution possible

input : 5 10 15 20 30 35 70 0

simple B-Tree with no key distribution

level print

20

10 35

5 15 30 70

 delete 10

level print

20 35

5 15 30 70

and distribution will prefer left node over right and merging will follow same criteria

 TEST CASE ( check it )without key distribution

**Input for tree of order 5 ( 5 way )**

4 5 6 14 15 16 17 18 19 20 21 90 99 100 101 82 89 80 52 59 13 85 44 32 27 0

Input keys to delete

5   52  14  80  82

level order output:

16 19 32 90

4 6 13 15 17 18 20 21  27 44 59 85 89 99 100 101

RCC - 17-11-2020 Left Rotations

Create a balanced BST with given input using left rotations.

Test case

input :   // -1 for end and 0 for LR and -2 for data end

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

output : Preorder ( LDR ) : 4 2 1 3 6 5 7

RCC - 17-11-2020 Right Rotations

Create a balanced BST with given input using right rotations.

Test case

input :   // -1 for end and 0 for LR and -2 for data end

15 14 13 -1 15 0 12 11 -1 13 0 10 -1 14 0 9 -1 11 0 -2

output Preorder  : 12 10 9 11 14 13 15

### RCC-17-11-2020 Treaps

### Treaps

test case :

input1 : p 45 s 63 h 21 m 36 e 27 g 9 #

input 2 ( delete  read two keys one by one without space) :  gh

output ( after deleting g, h) : LDR :

e 27 m 36 p 45 s 63

Ternary Tree

Ternary Tree

Create a Ternary Tree by taking input of 4 strings

and then take input for two search strings and then print 1 if string is present or 0 if it is not there for each of them.

 input : //cin>> 4 strings

indiana

india

jakarta

japan

//cin>> 1st string

//cin>> 2nd string

india

jap

**Output** : 1 0

Rotation Mutation - T1 to T2

Covert a Binary Search Tree T1 into another Binary Search Tree T2. ( similar shape )

These two binary search trees T1 and T2 are built with the same set of keys, {1,..,n}.

First construct T1 and T2 with the given input Preorders .

Then covert T1 into the same shape of T2

and print T1 in preorder which you should be same as that preorder of T2.

Test case :

input : **T1 Preorder DLR**:   6 4 2 1 3 5 8 7 9 -1

input : **T2 Preorder DLR**:   7 5 4 2 1 3 6 8 9 -1

output : **T1 Preorder DLR:**7 5 4 2 1 3 6 8 9      // T1 is  same as T2's shape

### RCC - Trie

Construct a trie with the following strings and print them in sorted order.

abd5c2

ac8b9

bd367abd367a

cdd294a1

da5b4c63

input : abd5c2 ac8b9 cdd294a1 da5b4c63 bd367#

output : abd5c2 ac8b9 bd367 cdd294a1 da5b4c63